The structure of Magnus

Timothy Daly

March, 2005

Abstract

Contents

1 The global classes

1.1 global/BaseObjectOf.h

— **BaseObjectOf.h** —

```
#include "PureRep.h"
#include "ObjectOf.h"

\getchunk{template BaseObjectOf}
```

———-

1.1.1 template BaseObjectOf

— **template BaseObjectOf** —

```
template<class Rep> class BaseObjectOf : private ObjectOf<Rep> {
public:
protected:
  ObjectOf<Rep>::look;
  ObjectOf<Rep>::change;
  ObjectOf<Rep>::enhance;
  ObjectOf<Rep>::acquireRep;
  BaseObjectOf( Rep* newrep ) : ObjectOf<Rep>( newrep)
private:
  void force_derivation( )
};
```

———-

1.2 global/DerivedObjectOf.h

1.2.1 template DerivedObjectOf

— **template DerivedObjectOf** —

```
template <class Base, class Rep> class DerivedObjectOf : public Base {
public:
  const Rep* look( ) const
  Rep* enhance( ) const
  Rep* change( )
  DerivedObjectOf( Rep* newrep ) : Base( newrep )
  DerivedObjectOf( const Base& b ) : Base( b )
};
```

———-

1.3 global/ExtendedIPC.h

— **ExtendedIPC.h** —

```
#include "Integer.h"
#include "Rational.h"

inline ostream& operator < ( ostream& ostr, const Integer& n )
inline ostream& operator < ( ostream& ostr, const Rational& r )
inline istream& operator > ( istream& istr, Integer& n )
inline istream& operator > ( istream& istr, Rational& r )
```

———-

1.4 global/GenericObject.h

— **GenericObject.h** —

```
#include "PureRep.h"
#include "BaseObjectOf.h"
#include "Type.h"

\getchunk{struct GenericRep}
\getchunk{class GenericObject}
```

———-

1.4.1 struct GenericRep

— **struct GenericRep** —

```
struct GenericRep : public PureRep {
  static const Type theGenericObjectType;
  static Type type()
  virtual Type actualType() const
  PureRep* clone( ) const
};
```

———-

1.4.2 class GenericObject

— **class GenericObject** —

```
class GenericObject : protected BaseObjectOf<GenericRep> {
public:
  GenericObject( ) : BaseObjectOf<GenericRep>( new GenericRep )
  static Type type()
  Type actualType() const
protected:
  GenericObject( GenericRep* newrep ) : BaseObjectOf<GenericRep>(newrep)
};
```

———-

1.5 global/IPC.h

— **IPC.h** —

```
inline ostream& operator < ( ostream& ostr, int n )
inline ostream& operator < ( ostream& ostr, unsigned int n )
inline ostream& operator < ( ostream& ostr, short n )
inline ostream& operator < ( ostream& ostr, long n )
inline ostream& operator < ( ostream& ostr, bool b )
inline ostream& operator < ( ostream& ostr, char c )
inline ostream& operator < ( ostream& ostr, const char *s )
inline ostream& operator < ( ostream& ostr, const double d )

inline istream& operator > ( istream& istr, int& n )
inline istream& operator > ( istream& istr, unsigned int& n )
inline istream& operator > ( istream& istr, short& n )
inline istream& operator > ( istream& istr, long& n )
inline istream& operator > ( istream& istr, bool& b )
inline istream& operator > ( istream& istr, char& c )
inline istream& operator > ( istream& istr, char* s )
inline istream& operator > ( istream& istr, double d )
inline istream& operator > ( istream& istr, void* p )
```

———-

1.6 global/ObjectOf.h

— **ObjectOf.h** —

```
#include "RefCounter.h"

\getchunk{template ObjectOf}
```

———-

1.6.1 template ObjectOf

— **template ObjectOf** —

```
template<class Rep> class ObjectOf {
public:
  ObjectOf( const ObjectOf& o )
 ~ObjectOf()
  ObjectOf& operator = ( const ObjectOf& o )
protected:
  const Rep* look( ) const
  Rep* enhance( ) const
  Rep* change( )
  void acquireRep( const Rep* rep )
  ObjectOf( Rep* newrep )
private:
  Rep* theRep;
  void force_derivation( )
};
```

———-

1.7 global/PureRep.h

— **PureRep.h** —

```
#include "RefCounter.h"

\getchunk{struct PureRep}
```

———-

1.7.1 struct PureRep

— **struct PureRep** —

```
struct PureRep : public RefCounter
{
  public :
    virtual ~PureRep( )
    virtual PureRep* clone( ) const = 0;
};
```

———-

1.8 global/RefCounter.h

1.8.1 class RefCounter

— class RefCounter —

```
class RefCounter
{
  public:
    typedef unsigned int refCounterType;
    RefCounter( ) : xrefs(0)
    RefCounter( const RefCounter& rc ) : xrefs(0)
    bool lastRef( ) const
    bool sharedRef( ) const
    void addRef( ) const
    void delRef( ) const
  private:
    refCounterType xrefs;
    RefCounter& operator = ( const RefCounter& );
};
```

———-

1.9 global/Trichotomy.h

1.9.1 class Trichotomy

— class Trichotomy —

```
class Trichotomy
{
  public:
    enum TrichotomyValue { DONTKNOW = -1, NO, YES };
    class dontknow
    Trichotomy( bool b ) : theValue(convert(b))
    Trichotomy( int i ) : theValue(convert(i))
    Trichotomy( void* p ) : theValue(convert(p))
    Trichotomy( const dontknow& ) : theValue(DONTKNOW)
    operator bool ( ) const
    operator int ( ) const
    inline friend bool operator == ( const Trichotomy& t, const Trichotomy& u )
    Trichotomy operator ! ( ) const
    bool defined( ) const
    bool undefined( ) const
    friend ostream& operator << ( ostream& ostr, const Trichotomy& t )
    friend ostream& operator < ( ostream& ostr, const Trichotomy& t )
    friend istream& operator > ( istream& istr, Trichotomy& t )
  private:
```

```
    static TrichotomyValue convert( bool b )
    static TrichotomyValue convert( void* p )
    TrichotomyValue theValue;
    inline bool operator != ( const Trichotomy& t, const Trichotomy& u )
    inline Trichotomy operator && ( const Trichotomy& t, const Trichotomy& u )
    inline Trichotomy operator || ( const Trichotomy& t, const Trichotomy& u )
    extern const Trichotomy YES, NO, DONTKNOW, yes, no, dontknow;
```

———-

2 The Apps classes

2.1 Apps/include/PresentationProblems.h

— **PresentationProblems.h** —

```
#include "FPGroup.h"
#include "FPGroupRep.h"
#include "KBMachine.h"
#include "RKBPackage.h"
#include "File.h"
#include "Subgroup.h"

\getchunk{class KernelOfHom}
\getchunk{class ImageOfHom}
\getchunk{class NewPresentation}
```

———-

2.1.1 class KernelOfHom

— **class KernelOfHom** —

```
class KernelOfHom
{
 public:
   KernelOfHom( const FPGroup& d, const FPGroup& r,
                const KBMachine& k, const VectorOf<Word> v ) :
      domain( d ), range( r ), kb( k ), images( v )
   KernelOfHom( ) :
      domain(), range(), kb(), images()
   FPGroup getKernelPresentation( );
 private:
   FPGroup domain;
   FPGroup range;
   KBMachine kb;
```

```
   VectorOf<Word> images;
```

———-

2.1.2 class ImageOfHom

— class ImageOfHom —

```
class ImageOfHom
{
  public:
    ImageOfHom( const FPGroup& d, const FPGroup& r, const VectorOf<Word> v ) :
      domain( d ), range( r ), images( v )
    ImageOfHom() :
      domain(), range(), images()
    FPGroup getImageOfHomo( VectorOf<Word>& v );
 private:
   FPGroup domain;
   FPGroup range;
   VectorOf<Word> images;
};
```

———-

2.1.3 class NewPresentation

— class NewPresentation —

```
class NewPresentation
{
  public:
    NewPresentation( const FPGroup& g, const KBMachine& k,
                     const VectorOf<Word> v ) :
      G( g ), kb( k ), newBasis( v )
    NewPresentation( ) :
      G( ), kb( ), newBasis( )
    FPGroup getNewPresentation( );
  private:
    FPGroup G;
    KBMachine kb;
    VectorOf<Word> newBasis;
};
```

———-

3 The AProducts classes

3.1 AProducts/include/AmalgamatedProductParser.h

— **AmalgamatedProductParser.h** —

```
#include "PresentationParser.h"
#include "APofFreeGroups.h"
#include "APwithOneRelator.h"

\getchunk{class AmalgamatedProductParser}
```

———-

3.1.1 class AmalgamatedProductParser

— **class AmalgamatedProductParser** —

```
class AmalgamatedProductParser : protected PresentationParser
{
  public:
    AmalgamatedProductParser(istream &istr) : PresentationParser(istr)
    VectorOf<Word> parseAPRelator(const VectorOf<Chars>& names,
                                  Chars& errMesg);
    VectorOf<Word> parseAPRelatorList(const VectorOf<Chars>& names,
                                      Chars& errMesg, int axRelators);
    void parseFactorsAndRelations(FreeGroup& f1, FreeGroup& f2,
                                  VectorOf<Word>& gens1, VectorOf<Word>& gens1,
                                  int maxRelators, Chars& errMesg);
    AmalgProductOfFreeGroupsRep* parseAPofFreeGroups( Chars& errMesg );
    APwithOneRelatorRep* parseAPwithOneRelator( Chars& errMesg );
};
\subsection{AProducts/include/AP-fixups.h}
\begin{chunk}{AP-fixups.h}
#include "SGofFreeGroup.h"
#include "Associations.h"
#include "File.h"

\getchunk{struct RelatorConjugate}
\getchunk{template swap}
\getchunk{functions for SGofFreeGroup}
\getchunk{template excludeFrom}
\getchunk{template cyclicallyPermute}
\getchunk{functions for FreeGroup}
\getchunk{functions for Word}
\getchunk{class DetailedReport}
```

———-

3.1.2 struct RelatorConjugate

— **struct RelatorConjugate** —

```
struct RelatorConjugate
{
  RelatorConjugate( ) : relator(), conjugator()
  RelatorConjugate( const Word& r, const Word& c ) :
    relator(r), conjugator(c)
  bool operator==( const RelatorConjugate& t) const
  Word relator, conjugator;
};
  inline ostream& operator<<( ostream& ostr, const RelatorConjugate& rc )
  inline istream& operator>>( istream& istr, RelatorConjugate& rc )
  inline ostream& operator<( ostream& ostr, const RelatorConjugate& rc )
  inline istream& operator>( istream& istr, RelatorConjugate& rc )
  typedef VectorOf<RelatorConjugate> ProductOfRelatorConjugates;
  ProductOfRelatorConjugates conjugateBy(
    const ProductOfRelatorConjugates& product, const Word& conjugator );
```

———-

3.1.3 template swap

— **template swap** —

```
template <class T> inline void swap( T& a, T& b )
{
  T temp = a;
  a = b;
  b = temp;
}
```

———-

3.1.4 functions for SGofFreeGroup

— **functions for SGofFreeGroup** —

```
  inline ostream& operator << (ostream& o, const SGofFreeGroup& sg)
```

———-

3.1.5 template excludeFrom

— template excludeFrom —

```
template <class T> inline void excludeFrom(VectorOf<T>& v, int i)
```

———-

3.1.6 template cyclicallyPermute

— template cyclicallyPermute —

```
template< class T > VectorOf<T> cyclicallyPermute(const VectorOf<T>& v, int j)
```

———-

3.1.7 functions for FreeGroup

— functions for FreeGroup —

```
  void maximalRootInFreeGroup(const Word& w, Word& root, int& power);
  Trichotomy conjugacyProblem(const FreeGroup& G, const Word& u,
                              const Word& v, Word& conjugator );
```

———-

3.1.8 functions for Word

— functions for Word —

```
  VectorOf<int> exponentSum(const Word& w);
  Word cyclicallyReduce(const Word& w, Word& conjugator);
  Word compose(const VectorOf<Word>& v);
  inline int tailAgreementLength( const Word& u, const Word& v )
  int GCD2(int a, int b);
  Word cyclicallyShortenByRelators(
    const Word& u, const SetOf<Word>& givenRelators,
    Word& conjugator, ProductOfRelatorConjugates& productOfRelatorConjugates);
```

———-

3.1.9 class DetailedReport

— class DetailedReport —

```
class DetailedReport
{
  public:
    DetailedReport( const FPGroup& group, const VectorOf<Word>& subgroup,
                    const Chars fileName = File().getFileName() );
    DetailedReport( const FPGroup& group,
                    const Chars fileName = File().getFileName() );
    ostream& file()
    bool haveDetails( ) const;
    Chars getFileName( ) const;
    void printDehnTransformationDetails( const Word& w );
    void printTrivialWordDetails( const Word& w,
                                  const ProductOfRelatorConjugates& deco,
                                  const bool header );
    void printNonTrivialWordDetails( const Word& w,
                                     const Chars& expl,
                                     const bool header );
    void printTrivialGeneratorDetails( const Generator& g,
                                       const ProductOfRelatorConjugates& deco,
                                       const bool header );
    void printNonTrivialGeneratorDetails( const Generator& g,
                                          const Chars& explanation,
                                          const bool header );
    void printTrivialCommutatorDetails(const Chars& comm,
                                       const ProductOfRelatorConjugates& deco,
                                       const bool header );
    void printNonTrivialCommutatorDetails( const Chars& comm,
                                           const Chars& explanation,
                                           const bool header );
    void printSubgroupElement( const Word& w,
                               const ProductOfRelatorConjugates& deco,
                               const bool header );
private:
  void printTrivialWordDecomposition( const ProductOfRelatorConjugates& deco );
  void printHeader( const bool header );
  void buildRelators( );
  bool bHaveDetails;
  File theFile;
  FPGroup G;
  VectorOf<Word> H;
  bool builtRelators;
  AssociationsOf<Word, int> theRelators;
  AssociationsOf<Word, int> theSGenerators;
};
```

———-

3.2 AProducts/include/APofFreeGroups.h

— APofFreeGroups.h —

```
#include "FreeGroup.h"
#include "FPGroup.h"
#include "APofFreeGroupsRep.h"

\getchunk{class AmalgProductOfFreeGroups}
```

———-

3.2.1 class AmalgProductOfFreeGroups

— class AmalgProductOfFreeGroups —

```
class AmalgProductOfFreeGroups :
    public DerivedObjectOf<FPGroup,AmalgProductOfFreeGroupsRep>
{
  public:
    AmalgProductOfFreeGroups() :
        DerivedObjectOf<FPGroup, AmalgProductOfFreeGroupsRep>(
          new AmalgProductOfFreeGroupsRep(FreeGroup(), FreeGroup(),
          VectorOf<Word>(), VectorOf<Word>() ) )
    AmalgProductOfFreeGroups(const FreeGroup& g1, const FreeGroup& g2,
          const VectorOf<Word>& gen1,
          const VectorOf<Word>& gen2  ) :
        DerivedObjectOf<FPGroup, AmalgProductOfFreeGroupsRep>(
          new AmalgProductOfFreeGroupsRep(g1, g2, gen1, gen2) )
    AmalgProductOfFreeGroups(const SGofFreeGroup& sg1,
                             const SGofFreeGroup& sg2) :
        DerivedObjectOf<FPGroup, AmalgProductOfFreeGroupsRep>(
          new AmalgProductOfFreeGroupsRep(sg1, sg2) )
    AmalgProductOfFreeGroups( const Group& G ) :
        DerivedObjectOf<FPGroup,AmalgProductOfFreeGroupsRep>(G)
    static Type type( )
    FreeGroup factor( const NumberOfFactor& t ) const;
    SGofFreeGroup subgroup( const NumberOfFactor& t ) const;
    Trichotomy isFree( ) const
    VectorOf<Word> decompose(const Word& w) const
    VectorOf<Word> reducedDecomposition(const Word& w) const
    Word reducedFormOf(const Word& w) const
    VectorOf<Word> normalDecomposition(const Word& w) const
    Word normalFormOf(const Word& w) const
    void cyclicDecomposition(const Word& w, VectorOf<Word>& result,
```

```
                          Word& conjugator) const
    void cyclicReduction(const Word& w, Word& result, Word& conjugator) const
    int numberOfSubstitutions( const Word& w ) const
    NumberOfFactor factorOfFormalWord(const Word& w) const
    NumberOfFactor factorOfElement(const Word& w) const
    Word localToGlobal(const LocalWord& w) const
    LocalWord globalToLocal(const Word& w) const
    Trichotomy isHyperbolic() const
    void maximalRoot(const Word& w, Word& root, int& power) const
    bool isProperPower(const Word& w) const
    bool isProperPowerOfSecond(const Word& u, const Word& w,
                               int& power) const
    bool commute(const Word& u, const Word& w) const
    bool isSubgroupAbelian(const VectorOf<Word>& subgroupWords) const
    bool isSubgroupTrivial(const VectorOf<Word>& vec) const
    bool isSubgroupCyclic(const VectorOf<Word>& vec) const
    void printDecomposition(ostream& ostr, const VectorOf<Word> deco) const
protected:
  AmalgProductOfFreeGroups( AmalgProductOfFreeGroupsRep* newrep ) :
  DerivedObjectOf<FPGroup,AmalgProductOfFreeGroupsRep>(newrep)
private:
};
```

———-

3.3 AProducts/include/APofFreeGroupsRep.h

— **APofFreeGroupsRep.h** —

```
#include "FPGroupRep.h"
#include "FreeGroup.h"
#include "VectorPtr.h"
#include "SGofFreeGroup.h"
#include "AP-fixups.h"
#include "Automorphism.h"

enum NumberOfFactor { LEFT_FACTOR = 0, RIGHT_FACTOR = 1,
                      PRODUCT, INTERSECTION };

\getchunk{struct LocalWord}
\getchunk{struct AmalgProductOfFreeGroupsRep}
```

———-

3.3.1 struct LocalWord

— **struct LocalWord** —

```
struct LocalWord
{
  LocalWord() : theWord(), theFactor(INTERSECTION)
  LocalWord(const Word& w, const NumberOfFactor& fac) :
     theWord(w), theFactor(fac)
  operator Word() const
  friend LocalWord operator * (const LocalWord& w1, const LocalWord& w2);
  LocalWord& operator *= (const LocalWord& w)
  LocalWord inverse() const
  LocalWord freelyReduce() const
  Word theWord;
  NumberOfFactor theFactor;
};
```

———-

3.3.2 struct AmalgProductOfFreeGroupsRep

— struct AmalgProductOfFreeGroupsRep —

```
struct AmalgProductOfFreeGroupsRep : FPGroupRep
{
  AmalgProductOfFreeGroupsRep(const FreeGroup& g1, const FreeGroup& g2,
                              const VectorOf<Word>& gen1,
                              const VectorOf<Word>& gen2 );
  AmalgProductOfFreeGroupsRep(const SGofFreeGroup& sg1,
                              const SGofFreeGroup& sg2);
  PureRep* clone( ) const
  static const Type theAmalgProductOfFreeGroupsType;
  static Type type( )
  Type actualType( ) const
  int order( ) const;
  Trichotomy isTrivial( ) const;
  Trichotomy isFinite( ) const;
  Trichotomy isInfinite( ) const;
  Trichotomy isAbelian( ) const;
  Trichotomy isFree( ) const;
  Trichotomy isHyperbolic() const;
  void printOn(ostream&) const;
  GroupRep* readFrom(istream&, Chars&) const;
  void printRelators(ostream&) const;
  void printDecomposition(ostream& ostr, const VectorOf<Word>& deco) const;
  Elt eval( const Word& w ) const
  Trichotomy wordProblem( const Word& w ) const
  NumberOfFactor factorOfFormalWord(const Word& w) const;
  NumberOfFactor factorOfElement(const Word& w) const;
  VectorOf<Word> decompose(const Word& w) const;
  VectorOf<Word> reducedDecomposition(const Word& w) const;
```

```
  Word reducedFormOf(const Word& w) const
  VectorOf<Word> normalDecomposition(const Word& w) const;
  Word normalFormOf(const Word& w) const
  int lengthOf(const Word& w) const
  int numberOfSubstitutions( const Word& w ) const;
  Word localToGlobal(const LocalWord& w) const
  Word localToGlobal(const Word& theWord, NumberOfFactor theFactor) const;
  LocalWord globalToLocal(const Word& w) const;
  NumberOfFactor factorOfGenerator(const Generator& gen) const
  LocalWord mapFromSubgroup(const LocalWord& w) const;
  Word mapFromSubgroup(const Word& w) const
  bool isElementOfSubgroup(const LocalWord& w) const;
  LocalWord rightSchreierRepresentativeOf(const LocalWord& w) const;
  void makeSubgroupMapping(const VectorOf<Word>& gen1,
                           const VectorOf<Word>& gen2);
  void fixGeneratorsNames();
  virtual void maximalRoot(const Word& w, Word& root, int& power) const;
  bool isProperPower(const Word& w) const;
  bool isProperPowerOfSecond(const Word& u, const Word& w, int& power) const;
  bool commute(const Word& u, const Word& w) const;
  bool isSubgroupTrivial(const VectorOf<Word>& subgrp) const;
  bool isSubgroupCyclic(const VectorOf<Word>& subgrp) const;
  bool isSubgroupAbelian(const VectorOf<Word>& subgrp) const;
  void cyclicReduction(const Word& w, Word& result, Word& conjugator) const;
  void cyclicDecomposition(const Word& w, VectorOf<Word>& result,
                           Word& conjugator) const;
  VectorPtrOf<FreeGroup> factor;
  VectorPtrOf<SGofFreeGroup> assocSubgroup;
  VectorPtrOf<Map> subgroupMapping;
  VectorPtrOf<Automorphism> nielsenBasisToGensOfSubgroup;
  int rankOfSubgroups;
  int numerationShift;
};
```

———-

3.4 AProducts/include/APwithOneRelator.h

— APwithOneRelator.h —

```
#include "APofFreeGroups.h"
#include "APwithOneRelatorRep.h"

\getchunk{class APwithOneRelator}
```

———-

3.4.1 class APwithOneRelator

— **class APwithOneRelator** —

```
class APwithOneRelator :
    public DerivedObjectOf<AmalgProductOfFreeGroups,APwithOneRelatorRep>
{
public:
  APwithOneRelator() :
    DerivedObjectOf<AmalgProductOfFreeGroups,
                APwithOneRelatorRep>(new APwithOneRelatorRep( 0 ))
  APwithOneRelator(const FreeGroup& g1, const FreeGroup& g2,
                   const Word& a, const Word& b ) :
    DerivedObjectOf<AmalgProductOfFreeGroups,
                APwithOneRelatorRep>(new APwithOneRelatorRep(g1, g2, a, b))
  APwithOneRelator(const FreeGroup& g1, const FreeGroup& g2,
                    const VectorOf<Word>& gen1, const VectorOf<Word>& gen2) :
    DerivedObjectOf<AmalgProductOfFreeGroups, APwithOneRelatorRep>
                ( new APwithOneRelatorRep(g1, g2, gen1, gen2) )
  APwithOneRelator(const SGofFreeGroup& sg1, const SGofFreeGroup& sg2) :
    DerivedObjectOf<AmalgProductOfFreeGroups, APwithOneRelatorRep>
                ( new APwithOneRelatorRep(sg1, sg2) )
  APwithOneRelator( const Group& G ) :
    DerivedObjectOf<AmalgProductOfFreeGroups,APwithOneRelatorRep>(G)
  static Type type( )
  Trichotomy conjugacyProblem(const Word& u, const Word& w,
                              Word& conjugator) const
  VectorOf<Word> centralizerOf(const Word& w) const
protected:
  APwithOneRelator( APwithOneRelatorRep* newrep ) :
    DerivedObjectOf<AmalgProductOfFreeGroups,APwithOneRelatorRep>(newrep)
private:
};
```

———-

3.5 AProducts/include/APwithOneRelatorRep.h

— **APwithOneRelatorRep.h** —

```
#include "APofFreeGroups.h"

\getchunk{struct APwithOneRelatorRep}
```

———-

3.5.1 struct APwithOneRelatorRep

— **struct APwithOneRelatorRep** —

```
struct APwithOneRelatorRep : AmalgProductOfFreeGroupsRep
{
  APwithOneRelatorRep(const int dummy) :
    AmalgProductOfFreeGroupsRep( FreeGroup(), FreeGroup(),
                                 VectorOf<Word>(), VectorOf<Word>() )
    APwithOneRelatorRep(const FreeGroup& g1, const FreeGroup& g2,
                        const Word& gen1, const Word& gen2 );
    APwithOneRelatorRep(const FreeGroup& g1, const FreeGroup& g2,
                        const VectorOf<Word>& gen1,
                        const VectorOf<Word>& gen2) :
      AmalgProductOfFreeGroupsRep( g1, g2, gen1, gen2 )
    APwithOneRelatorRep(const SGofFreeGroup& sg1, const SGofFreeGroup& sg2) :
      AmalgProductOfFreeGroupsRep( sg1, sg2 )
    Word assocWord(int i) const
    PureRep* clone( ) const
    static const Type theAPwithOneRelatorType;
    static Type type( )
    Type actualType( ) const
    Trichotomy isHyperbolic() const;
    GroupRep* readFrom(istream&, Chars&) const;
    VectorOf<int> computeInvariants() const;
    void maximalRoot(const Word& w, Word& root, int& power) const;
    VectorOf<Word> centralizerOf(const Word& w) const;
    Trichotomy conjugacyProblem( const Word& u, const Word& v ) const
    Trichotomy conjugacyProblem(const Word& u, const Word& v,
                                Word& conjugator) const;
private:
  void maximalRoot_case1(const Word& w, Word& root, int& power) const;
  void maximalRoot_case2(const Word& w, Word& root, int& power) const;
  Word translateToAnotherPresentation(const Word& w, const Word& aConjInv,
                                      const Word& bConjInv) const;
  Trichotomy conjugacyProblem_reduced(const Word& u, const Word& v,
                                      Word& conjugator) const;
  Trichotomy conjugacyProblem_case1(const Word& u, const Word& v,
                                    Word& conjugator) const;
  Trichotomy conjugacyProblem_case2(VectorOf<Word>& uDec, VectorOf<Word>& vDec,
                                    Word& conjugator) const;
};
```

———-

3.6 AProducts/include/HNNExtension.h

— **HNNExtension.h** —

```
#include "FPGroup.h"

\getchunk{class HNNExtensionRep}
\getchunk{class HNNExtension}
```

———-

3.6.1 class HNNExtensionRep

— class HNNExtensionRep —

```
class HNNExtensionRep : public FGGroupRep
{
public:
  HNNExtensionRep( const FPGroup& G );
  static const Type theHNNExtensionType;
  static Type type( )
  Type actualType( ) const
  enum NumberOfSubgroup { A, B };
  enum Pinch { UP, DOWN };
  const FPGroup& getFPGroup() const
  virtual const FGGroup& getBasisGroup( ) const = 0;
  Generator stableLetter( ) const
  Trichotomy isTrivial( ) const;
  Trichotomy isFinite( ) const;
  Trichotomy isInfinite( ) const;
  Trichotomy isAbelian( ) const;
  virtual Trichotomy isFree( ) const = 0;
  bool isSubgroupTrivial( const VectorOf<Word>& vec ) const;
  bool isSubgroupAbelian( const VectorOf<Word>& vec ) const;
  Trichotomy areEqual( const Elt& e1, const Elt& e2 ) const
  Elt eval( const Word& w ) const
  int lengthOf( const Word& w ) const
  Word reducedFormOf( const Word& w ) const
  Word normalFormOf( const Word& w ) const
  Word cyclicallyReducedFormOf( const Word& w, Word conjugator ) const
  VectorOf<Word> decompositionOf( const Word& w ) const;
  VectorOf<Word> reducedDecompositionOf( const Word& w ) const;
  VectorOf<Word> normalDecompositionOf( const Word& w ) const;
  VectorOf<Word> cyclicallyReducedDecompositionOf( const Word& w,
                                                   Word& conjugator ) const;
  static Word compose( const VectorOf<Word>& V );
  Trichotomy wordProblem( const Word& w ) const;
  virtual Trichotomy  maximalRoot( const Word& w, Word& maxRoot,
                                   int& maxPower ) const = 0;
  void printOn( ostream& ) const;
  void printDecomposition( ostream& ostr, const VectorOf<Word>& deco) const
  void write( ostream& ostr ) const;
```

```
  void read( istream& istr );
protected:
  FPGroup theFPGroup;
  virtual Word mappingFromSubgroup( NumberOfSubgroup S,
                                    const Word& w ) const = 0;
  virtual Word getGeneratorOfSubgroup( const NumberOfSubgroup subgrp,
                                       int gen ) const = 0;
  virtual int getNumberOfGeneratorsInSubgroup( const NumberOfSubgroup subgrp )
                                        const = 0;
  virtual bool subgroupContains( const NumberOfSubgroup subgrp, const Word& w)
                                         const  = 0;
  virtual Word rightRepresentative( const NumberOfSubgroup subgrp,
                                    const Word& w ) const = 0;
private:
  int powerOfStableGen( int component, const VectorOf<Word>& deco ) const;
  bool suspectPinch(int component, const VectorOf<Word>& decomposition) const;
  Pinch formPinch( int component, const VectorOf<Word>& deco ) const;
};
```

———-

3.6.2 class HNNExtension

— class HNNExtension —

```
class HNNExtension : public DerivedObjectOf<FGGroup,HNNExtensionRep>
{
public:
  typedef HNNExtensionRep::NumberOfSubgroup NumberOfSubgroup;
  const FPGroup& getFPGroup() const
  const FGGroup& getBasisGroup( ) const
  Generator stableLetter( ) const
  Trichotomy isFree( ) const
  bool isSubgroupTrivial( const VectorOf<Word>& vec ) const
  bool isSubgroupAbelian( const VectorOf<Word>& vec ) const
  int lengthOf( const Word& w ) const
  Word reducedFormOf( const Word& w ) const
  Word normalFormOf( const Word& w ) const
  Word cyclicallyReducedFormOf( const Word& w, Word conjugator ) const
  VectorOf<Word> decompositionOf( const Word& w ) const
  VectorOf<Word> reducedDecompositionOf( const Word& w ) const
  VectorOf<Word> normalDecompositionOf( const Word& w ) const
  VectorOf<Word> cyclicallyReducedDecompositionOf( const Word& w,
                                                   Word& conjugator) const
  static Word compose( const VectorOf<Word>& V ) ;
  virtual Trichotomy  maximalRoot( const Word& w, Word& maxRoot,
                                   int& maxPower ) const
  void printDecomposition( ostream& ostr, const VectorOf<Word> deco ) const
```

```
  friend ostream& operator < ( ostream& ostr, const HNNExtension& G )
  friend istream& operator > ( istream& istr, HNNExtension& G )
protected:
  HNNExtension( HNNExtensionRep *newrep ) :
    DerivedObjectOf<FGGroup,HNNExtensionRep> ( newrep )
private:
  HNNExtension( );
};
```

———-

3.7 AProducts/include/HNNExtOfFreeGroup.h

— HNNExtOfFreeGroup.h —

```
#include "HNNExtension.h"
#include "FreeGroup.h"
#include "SGofFreeGroup.h"
#include "VectorPtr.h"
#include "Automorphism.h"
#include "AP-fixups.h"

\getchunk{class HNNExtOfFreeGroupRep}
\getchunk{class HNNExtOfFreeGroup}
```

———-

3.7.1 class HNNExtOfFreeGroupRep

— class HNNExtOfFreeGroupRep —

```
class HNNExtOfFreeGroupRep: public HNNExtensionRep
{
public:
  HNNExtOfFreeGroupRep( const FreeGroup& F, const Chars& nameOfStableLetter,
                        const SGofFreeGroup& subgroupA,
                        const SGofFreeGroup& subgroupB );
  PureRep* clone( ) const
  static const Type theHNNExtOfFreeGroupType;
  static Type type( )
  Type actualType( ) const
  const FGGroup& getBasisGroup( ) const
  virtual int order( ) const
  Trichotomy isFree( ) const;
  Trichotomy isProperPower( const Word& w ) const;
  Trichotomy maximalRoot( const Word& w, Word& maxRoot, int& maxPower ) const;
```

```
  Trichotomy conjugacyProblem( const Word& w, const Word& u ) const
  Trichotomy conjugacyProblem( const Word& w, const Word& u,
                               Word& conjugator ) const;
  bool isProperPowerOfSecond( const Word& w, const Word& u, int& k ) const;
  GroupRep* readFrom( istream& istr, Chars& errMesg ) const;
  void write( ostream& ostr ) const;
  void read( istream& istr );
protected:
  void makeSubgroupsMappings();
  void makeReducedCyclicPresentation( ) const;
  Word mappingFromSubgroup( const NumberOfSubgroup S, const Word& w ) const;
  Trichotomy conjugacyProblem_reduced( const Word& u, const Word& v,
                                       Word& conjugator ) const;
  Trichotomy conjugateInSubgroups( NumberOfSubgroup S, const Word& u,
                                   const Word& v, Word& conjugator,
                                   bool oneIteration ) const;
  Trichotomy conjugacyProblem_case1( const Word& u, const Word& v,
                                     Word& conjugator ) const;
  Trichotomy conjugacyProblem_case2(VectorOf<Word>& uDec,
                                    VectorOf<Word>& vDec,
                                    Word& conjugator ) const;
  Word getGeneratorOfSubgroup(const NumberOfSubgroup S, const int gen) const
  int getNumberOfGeneratorsInSubgroup( const NumberOfSubgroup S ) const
  bool subgroupContains( const NumberOfSubgroup S, const Word& w ) const
  Word rightRepresentative( const NumberOfSubgroup S, const Word& w ) const

\getchunk{struct SpecialHNNExtOfFreeGroup}
\getchunk{class MaximalRootProblem}

  friend class MaximalRootProblem;
  FreeGroup theBasisFreeGroup;
  VectorPtrOf<SGofFreeGroup> theSubgroups;
  VectorOf< VectorOf<Word> > mapNielsenGensToSubgroupGens;
  SpecialHNNExtOfFreeGroup reducedPresentation;
};

\getchunk{class HNNExtOfFreeGroup}
```

———-

3.7.2 struct SpecialHNNExtOfFreeGroup

— struct SpecialHNNExtOfFreeGroup —

```
struct SpecialHNNExtOfFreeGroup
{
public:
  SpecialHNNExtOfFreeGroup() :
```

```
    H(0), toOriginalPresentation( FreeGroup(0), VectorOf<Word>(0) )
  SpecialHNNExtOfFreeGroup( const Automorphism& mapToOriginalPresentation,
                            const FreeGroup& basisFreeGroup,
                            const Chars& nameOfStableLetter,
                            const SGofFreeGroup& A,
                            const SGofFreeGroup& B,
                            bool  areAmalgSubgroupsConjugateSeparated,
                            const VectorOf<Word>& rootsOfSubgroupsGens,
                            const VectorOf<int>&  powersOfSubgroupsGens );
  SpecialHNNExtOfFreeGroup( const SpecialHNNExtOfFreeGroup& S );
  ~SpecialHNNExtOfFreeGroup()
  SpecialHNNExtOfFreeGroup& operator = ( const SpecialHNNExtOfFreeGroup& S );
  bool doneInit( ) const
  const HNNExtOfFreeGroupRep& presentation() const;
  Automorphism         toOriginalPresentation;
  bool                 areAmalgSubgroupsConjugateSeparated;
  VectorOf<Word>       theRootsOfSubgroupsGens;
  VectorOf<int>        thePowersOfSubgroupsGens;
private:
  HNNExtOfFreeGroupRep *H;
};
```

———-

3.7.3 class MaximalRootProblem

— class MaximalRootProblem¿¿= —

```
class MaximalRootProblem
{
public:
  MaximalRootProblem( const Word& w );
  Trichotomy answer() const
  Word root() const
  int power() const
  void solve( const SpecialHNNExtOfFreeGroup& group );
private:
    void length0( const Word& w );
    void lengthN( const VectorOf<Word> wDeco );
    void setAnswer( const Word& root, int power );
    SpecialHNNExtOfFreeGroup theGroup;
    Word theWord;
    Word theRoot;
    int  thePower;
    Trichotomy theAnswer;
    bool isSolved;
    Word stableLetter;
};
```

———-

3.7.4 class HNNExtOfFreeGroup

— class HNNExtOfFreeGroup —

```
class HNNExtOfFreeGroup
  : public DerivedObjectOf< HNNExtension, HNNExtOfFreeGroupRep>
{
public:
  HNNExtOfFreeGroup( const FreeGroup& F,
                     const Chars& nameOfStableLetter,
                     const SGofFreeGroup& subgroupA,
                     const SGofFreeGroup& subgroupB ) :
    DerivedObjectOf< HNNExtension,
                     HNNExtOfFreeGroupRep>(
                       new HNNExtOfFreeGroupRep(F, nameOfStableLetter,
                                                subgroupA, subgroupB ))
  Trichotomy isProperPower( const Word& w ) const
  Trichotomy conjugacyProblem( const Word& w, const Word& u,
                               Word& conjugator ) const
  bool isProperPowerOfSecond( const Word& w, const Word& u, int& k ) const
  friend ostream& operator < ( ostream& ostr, const HNNExtOfFreeGroup& G )
  friend istream& operator > ( istream& istr, HNNExtOfFreeGroup& G )
protected:
  HNNExtOfFreeGroup( HNNExtOfFreeGroupRep *newrep ) :
    DerivedObjectOf<HNNExtension, HNNExtOfFreeGroupRep> ( newrep )
};
```

———-

3.8 AProducts/include/HNNExtOfORGroup.h

— HNNExtOfORGroup.h —

```
#include "Set.h"
#include "Vector.h"
#include "VectorPtr.h"
#include "OneRelatorGroup.h"
#include "SubgroupOfOneRelatorGroup.h"
#include "OneRelatorGroupWithTorsion.h"

\getchunk{class HNNExtOfORGroupGeneric}
\getchunk{class HNNExtOfORGroup}
\getchunk{class HNNExtOfORGroupWithTorsion}
\getchunk{class HNNDoubleCoset}
```

———-

3.8.1 class HNNExtOfORGroupGeneric

— class HNNExtOfORGroupGeneric —

```
class HNNExtOfORGroupGeneric
{
public:
  enum NumberOfSubgroup { A = 0, B = 1 };
  enum Pinch { UP, DOWN };
  HNNExtOfORGroupGeneric( const OneRelatorGroup& G, const Chars& stableGenName,
                          const ORGSubgroup& A, const ORGSubgroup& B);
  HNNExtOfORGroupGeneric( const HNNExtOfORGroupGeneric& H );
  virtual ~HNNExtOfORGroupGeneric( );
  HNNExtOfORGroupGeneric& operator=( const HNNExtOfORGroupGeneric& H );
  virtual const ORGSubgroup& subgroup(NumberOfSubgroup i) const
  const OneRelatorGroup& basisGroup( ) const
  FPGroup getPresentation( ) const;
  Chars nameOfStableGenerator( ) const
  virtual Trichotomy wordProblem( const Word& w ) const = 0;
  int lengthOf( const Word& w ) const;
  int lengthOf( const VectorOf<Word>& deco ) const;
  VectorOf<Word> decompositionOf( const Word& w ) const;
  VectorOf<Word> reducedDecompositionOf( const Word& w ) const;
  VectorOf<Word> normalDecompositionOf( const Word& w ) const;
  VectorOf<Word> cyclicallyReducedDecompositionOf( const Word& w,
                                                   Word& conj ) const;
  Word mappingFromSubgroup( NumberOfSubgroup subgrp, const Word& w ) const;
  ProductOfRelatorConjugates mappingDecompositionOf(
    const NumberOfSubgroup& S, const Word& w, const Word& wInSBasis,
    const Word& tail ) const;
  Generator stableLetter( ) const
  virtual bool operator == ( const HNNExtOfORGroupGeneric& G ) const;
  friend inline ostream& operator << ( ostream& ostr,
                                       const HNNExtOfORGroupGeneric& H )
protected:
\getchunk{struct PinchStruct}
  void printOn( ostream& ostr ) const;
  void printDecomposition( ostream& ostr, const VectorOf<Word>& deco ) const;
  virtual void debugPrint( ostream& ostr ) const;
  virtual void write( ostream& ostr ) const;
  virtual void read( istream& istr );
  int powerOfStableGen(int component, const VectorOf<Word>& deco ) const;
  bool suspectPinch(int component, const VectorOf<Word>& deco ) const;
  bool abelianizationTest( const PinchStruct& pinch,
                           const VectorOf<int>& powersOfGens,
                           const VectorOf<bool>& subgroupGens ) const;
  PinchStruct formPinch(int component, const VectorOf<Word>& deco) const;
  int unusedGenerators( const Word& test, VectorOf<Word>& toNewGens,
                        VectorOf<Word>& toOldGens ) const;
```

```
  void init( const HNNExtOfORGroupGeneric& H );
  OneRelatorGroup *theBasisGroup;
  Chars theNameOfStableGenerator;
  ORGSubgroup *subgroups[2];
};
```

———-

3.8.2 struct PinchStruct

— **struct PinchStruct** —

```
struct PinchStruct
{
    Pinch type;
    int number;
    Word word;
};
```

———-

3.8.3 class HNNExtOfORGroup

— **class HNNExtOfORGroup** —

```
class HNNExtOfORGroup : public HNNExtOfORGroupGeneric
{
public:
  HNNExtOfORGroup( const OneRelatorGroup& G,
                   const Chars& stableGenName,
                   const SubgroupOfOneRelatorGroup& A,
                   const SubgroupOfOneRelatorGroup& B);
  Trichotomy wordProblem( const Word& test ) const;
  Trichotomy wordProblem( const Word& test, bool keepDetails,
    ProductOfRelatorConjugates& productOfRelatorConjugates ) const;
  VectorOf<Word> reducedDecompositionOf( const Word& w, bool keepDetails,
    ProductOfRelatorConjugates& productOfRelatorConjugates ) const;
private:
protected:
  int unusedGenerators( const Word& test, VectorOf<Word>& toNewGens,
                        VectorOf<Word>& toOldGens ) const;
};
```

———-

3.8.4 class HNNExtOfORGroupWithTorsion

— class HNNExtOfORGroupWithTorsion —

```
class HNNExtOfORGroupWithTorsion : public HNNExtOfORGroupGeneric
{
public:
  HNNExtOfORGroupWithTorsion( const OneRelatorGroupWithTorsion& G,
                              const Chars& stableGenName,
                              const SubgroupOfORGroupWithTorsion& A,
                              const SubgroupOfORGroupWithTorsion& B );
  Trichotomy wordProblem( const Word& test ) const;
  Trichotomy conjugacyProblem( const Word& u, const Word& v,
                               Word& conjugator ) const;
  Trichotomy maximalRoot( const Word& w, Word& root, int& power ) const;
private:
protected:
  Trichotomy conjugacyProblem_cyclicallyReduced( const VectorOf<Word>& uDeco,
    const VectorOf<Word>& vDeco, Word& conjugator ) const;

\getchunk{class MaximalRootProblem}

};
```

———-

3.8.5 class MaximalRootProblem

— class MaximalRootProblem —

```
class MaximalRootProblem
{
public:
    MaximalRootProblem( const Word& w );
    void solve( const HNNExtOfORGroupWithTorsion& group );
    Word root( ) const
    int power( ) const
    Trichotomy answer( ) const
  private:
    bool lexCheckOfStableLetters( const VectorOf<Word>& wDeco,
                                  const int rootLen ) const;
    void lengthN( const HNNExtOfORGroupWithTorsion& H,
                  const VectorOf<Word>& wDeco );
    void setAnswer( const Word& maxRoot, const int maxPower );
    void adjustRoot( );
    const Word theWord;
    Trichotomy theAnswer;
    Word theRoot;
```

```
    int thePower;
    bool isSolved;
};
```

———-

3.8.6 class HNNDoubleCoset

— class HNNDoubleCoset —

```
class HNNDoubleCoset
{
public:
  HNNDoubleCoset( const SubgroupOfORGroupWithTorsion& Sa,
                  const SubgroupOfORGroupWithTorsion& Sbw );
  Trichotomy solve( const Word& f, Word& a, const Word& g, Word& b ) const;
private:
  Trichotomy oneRelatorWithTorsionSolution(
    const Word& f, Word& a, const Word& g, Word& b ) const;
  Word minimalSpecialForm( const Word& w,
                           const SubgroupOfORGroupWithTorsion S ) const;
  SubgroupOfORGroupWithTorsion A;
  SubgroupOfORGroupWithTorsion B;
};
```

———-

3.9 AProducts/include/HNNParser.h

— HNNParser.h —

```
#include "PresentationParser.h"
#include "HNNExtOfFreeGroup.h"

\getchunk{class HNNExtensionParser}
```

———-

3.9.1 class HNNExtensionParser

— class HNNExtensionParser —

```
class HNNExtensionParser : public PresentationParser
{
public:
```

```
  HNNExtensionParser( istream& istr ) : PresentationParser(istr)
  HNNExtOfFreeGroupRep* parseHNNExtensionOfFreeGroup( Chars& );
protected:
  void parseHNNRelator( const VectorOf<Chars>& names, Word& left, Word& right,
                        Chars& errMesg);
  VectorOf<Word> parseHNNRelatorsList( const VectorOf<Chars>& names,
                                       Chars& errMesg, int maxRelators );
  void parseBasisFreeGroupAndRelations( FreeGroup& F, Chars& stableGenName,
                                        VectorOf<Word>& A, VectorOf<Word>& B,
                                        int maxRelators, Chars& errMesg );

};
```

———-

3.10 AProducts/include/MagnusBreakdown.h

— **MagnusBreakdown.h** —

```
#include "OneRelatorGroup.h"
#include "HNNExtOfORGroup.h"
#include "AbelianWord.h"
#include "Map.h"
#include "SuperGen.h"
#include "Range.h"
#include "Automorphism.h"

\getchunk{class MagnusBreakdown}
```

———-

3.10.1 class MagnusBreakdown

— **class MagnusBreakdown** —

```
class MagnusBreakdown
{
public:
  MagnusBreakdown( const OneRelatorGroup& G );
  OneRelatorGroup getORGroup() const
  HNNExtOfORGroup getHNNPresentation() const;
  int numberOfUsedGenerators() const
  Generator stableGenerator() const
  bool hasAccompGenerator() const
  Generator accompGenerator() const;
  ListOf<Generator> getGeneratorsWithZeroExpSum() const
```

```
  ListOf<Generator> getGeneratorsWithNonzeroExpSum() const
  ListOf<Generator> getDefaultBreakdownGenerators() const;
  int getExponentSumOf( const Generator& g ) const
  Word rewriteWordInOldGenerators( const Word& w ) const;
  Word rewriteWordInNewGenerators( const Word& w ) const;
  Automorphism embeddingOfORGroups( ) const
  Map toHNNPresentation( ) const
  Map toORGroup( ) const
  int numberOfOldGenerators( ) const
  int numberOfNewGenerators( ) const
  VectorOf<Range> getSubscriptsTable() const
  Generator adjoinSubscript(const Generator& oldGen, int subscript) const;
  Generator extractSubscript(const Generator& newGen, int& subscript) const;
  Trichotomy hasHNNPresentation() const
  bool findHNNPresentation();
  bool findHNNPresentation( const Generator& stableGen );
  bool findHNNPresentation( const Generator& stableGen,
                            const Generator& accompGen );
  void printOn( ostream& ostr ) const;
  MagnusBreakdown *readFrom( istream& istr ) const;
  void debugPrint( ostream& ostr ) const;
protected:
  void makeHNNPresentation( );
  bool chooseStableGenerator( );
  void makeSubscriptsTable( );
  void makeEmbedding();
  void makeTranslationTables();
  OneRelatorGroup       theORGroup;
  int                   theNumberOfOldGenerators;
  ListOf<Generator>     theGeneratorsWithZeroExpSum;
  ListOf<Generator>     theGeneratorsWithNonzeroExpSum;
  VectorOf<int>         exponentsOfOldGenerators;
  OneRelatorGroup       theLargerORGroup;
  HNNExtOfORGroup       theHNNPresentation;
  bool                  haveHNNPresentation;
  Automorphism          theEmbeddingOfORGroups;
  Map                   mapToHNN;
  Map                   mapToORGroup;
  SuperGen              stable, accomp;
  Generator             stableNewGen;
  int                   theNumberOfNewGenerators;
  VectorOf<Chars>       theNamesOfNewGenerators;
  VectorOf<Range>       subscriptsOfGenerator;
  VectorOf<int>         toNewGens;
  VectorOf<int>         toOldGens;

};

ProductOfRelatorConjugates liftUpProduct(
  const ProductOfRelatorConjugates& mProduct,
```

```
  const MagnusBreakdown& M, const OneRelatorGroup& G
);
```

———-

3.11 AProducts/include/OneRelatorGroup.h

— **OneRelatorGroup.h** —

```
#include "Word.h"
#include "Chars.h"
#include "Vector.h"
#include "FPGroup.h"
#include "AP-fixups.h"

\getchunk{class OneRelatorGroupRep}
\getchunk{class OneRelatorGroup}
\getchunk{class EnumeratorOfConsequences}
```

———-

3.11.1 class OneRelatorGroupRep

— **class OneRelatorGroupRep** —

```
class OneRelatorGroupRep : public FGGroupRep
{
public:
  OneRelatorGroupRep( int ngens, const Word& relator );
  OneRelatorGroupRep( const VectorOf<Chars>& gennames, const Word& relator );
  OneRelatorGroupRep( const FPGroup& G );
  PureRep* clone( ) const
  static const Type theOneRelatorGroupType;
  static Type type( )
  Type actualType( ) const
  int order( ) const;
  Trichotomy isTrivial( ) const;
  Trichotomy isFinite( ) const;
  Trichotomy isInfinite( ) const;
  Trichotomy isAbelian( ) const;
  Trichotomy isFree( ) const;
  Trichotomy wordProblem( const Word& w, bool keepDetails,
    ProductOfRelatorConjugates& productOfRelatorConjugates ) const;
  Trichotomy wordProblem( const Word& w ) const;
  Trichotomy areEqual( const Elt& e1, const Elt& e2 ) const
  Elt eval( const Word& w ) const
```

```
  Trichotomy conjugacyProblem( const Word& u, const Word& w ) const
  GroupRep* readFrom( istream& istr, Chars& errMesg ) const;
  void printOn( ostream& ostr ) const;
  bool operator == ( const OneRelatorGroupRep& G ) const
  bool operator != ( const OneRelatorGroupRep& G ) const
  void write( ostream& ostr ) const;
  void read( istream& istr );
  Word theRelator;
};
```

———-

3.11.2 class OneRelatorGroup

— class OneRelatorGroup —

```
class OneRelatorGroup
  : public DerivedObjectOf< FGGroup, OneRelatorGroupRep >
{
public:
  OneRelatorGroup( int ngens, const Word& relator ) :
    DerivedObjectOf<FGGroup, OneRelatorGroupRep>(
        new OneRelatorGroupRep( ngens, relator ) )
  OneRelatorGroup( const VectorOf<Chars>& gennames, const Word& relator ) :
    DerivedObjectOf<FGGroup, OneRelatorGroupRep>(
        new OneRelatorGroupRep( gennames, relator ) )
  OneRelatorGroup( const FPGroup& G ) :
    DerivedObjectOf<FGGroup, OneRelatorGroupRep>(
        new OneRelatorGroupRep( G ) )
  Word relator( ) const
  Trichotomy wordProblem( const Word& w ) const
  Trichotomy wordProblem( const Word& w, bool keepDetails,
    ProductOfRelatorConjugates& productOfRelatorConjugates ) const
  friend ostream& operator < ( ostream& ostr, const OneRelatorGroup& G )
  friend istream& operator > ( istream& istr, OneRelatorGroup& G )
protected:
  OneRelatorGroup( OneRelatorGroupRep *newrep ) :
    DerivedObjectOf<FGGroup, OneRelatorGroupRep> ( newrep )
};
```

———-

3.11.3 class EnumeratorOfConsequences

— class EnumeratorOfConsequences —

```
class EnumeratorOfConsequences
{
public:
  EnumeratorOfConsequences( const OneRelatorGroup& G );
  Word word() const;
  ProductOfRelatorConjugates tuple() const;
  void reset();
  bool done() const;
  void advance();
  friend ostream& operator < ( ostream& ostr,
                               const EnumeratorOfConsequences& ce )
  friend istream& operator > ( istream& istr, EnumeratorOfConsequences& ce )
private:
  void generate( ) const;
  Integer theCurrentWordNumber;
  Integer theLastComputedWordNumber;
  Word theCurrentWord;
  ProductOfRelatorConjugates theCurrentProduct;
  Word theRelator;
  FreeGroup theGroup;
};
```

———-

3.12 AProducts/include/OneRelatorGroupWithTorsion.h

— OneRelatorGroupWithTorsion.h —

```
#include "OneRelatorGroup.h"

\getchunk{class OneRelatorGroupWithTorsionRep}
\getchunk{class OneRelatorGroupWithTorsion}
```

———-

3.12.1 class OneRelatorGroupWithTorsionRep

— class OneRelatorGroupWithTorsionRep —

```
class OneRelatorGroupWithTorsionRep : public OneRelatorGroupRep
{
public:
  OneRelatorGroupWithTorsionRep( int ngens, const Word& relator );
  OneRelatorGroupWithTorsionRep( const VectorOf<Chars>& gennames,
                                 const Word& relator );
  OneRelatorGroupWithTorsionRep( const FPGroup& G );
```

```
  PureRep* clone( ) const
  static const Type theOneRelatorGroupWithTorsionType;
  static Type type( )
  Type actualType( ) const
  Trichotomy isFreeProduct( ) const;
  bool isProperMagnusSubgroup( const VectorOf<Generator>& gens ) const;
  bool doesMagnusSubgroupContainElt( const VectorOf<Generator>& subgroup,
    const Word& w, Word& wInSubgroupBasis ) const;
  bool isPowerOfEltInMagnusSubgroup( const VectorOf<Generator>& subgroup,
    const Word& w, Word& subgroupElt, int& powerOfElt ) const;
  Trichotomy wordProblem( const Word& w,
    ProductOfRelatorConjugates& productOfRelatorConjugates ) const;
  Trichotomy wordProblem( const Word& w ) const;
  Trichotomy conjugacyProblem( const Word& u, const Word& v, Word& conjugator )
    const;
  Trichotomy conjugacyProblem( const Word& u, const Word& w ) const
  Trichotomy maximalRoot( const Word& w, Word& maxRoot, int& maxPower ) const;
  int powerOfElt(const Word& w, Word& st, Word& x,
                 ProductOfRelatorConjugates& productOfRelatorConjugates) const;
  VectorOf<Word> centralizerOfElt( const Word& w ) const;
  GroupRep* readFrom( istream& istr, Chars& errMesg ) const;

};
```

———-

3.12.2 class OneRelatorGroupWithTorsion

— class OneRelatorGroupWithTorsion —

```
class OneRelatorGroupWithTorsion
  : public DerivedObjectOf<OneRelatorGroup,OneRelatorGroupWithTorsionRep>
{
public:
  OneRelatorGroupWithTorsion( int ngens, const Word& relator )
    : DerivedObjectOf<OneRelatorGroup, OneRelatorGroupWithTorsionRep>(
        new OneRelatorGroupWithTorsionRep( ngens, relator ) )
  OneRelatorGroupWithTorsion( const VectorOf<Chars>& gennames,
                              const Word& relator )
    : DerivedObjectOf<OneRelatorGroup, OneRelatorGroupWithTorsionRep>(
        new OneRelatorGroupWithTorsionRep( gennames, relator ) )
  OneRelatorGroupWithTorsion( const FPGroup& G )
    : DerivedObjectOf<OneRelatorGroup, OneRelatorGroupWithTorsionRep>(
        new OneRelatorGroupWithTorsionRep( G ) )
  bool isProperMagnusSubgroup( const VectorOf<Generator>& subgroup ) const
  bool doesMagnusSubgroupContainElt( const VectorOf<Generator>& subgroup,
    const Word& w, Word& wInSubgroupBasis ) const
  bool isPowerOfEltInMagnusSubgroup( const VectorOf<Generator>& subgroup,
```

```
    const Word& w, Word& subgroupElt, int& powerOfElt ) const
  Trichotomy wordProblem( const Word& w,
    ProductOfRelatorConjugates& productOfRelatorConjugates ) const
  Trichotomy conjugacyProblem( const Word& u, const Word& v, Word& conjugator)
    const
  Trichotomy maximalRoot( const Word& w, Word& maxRoot, int& maxPower ) const
  int powerOfElt(const Word& w, Word& st, Word& x,
                 ProductOfRelatorConjugates& productOfRelatorConjugates ) const
  VectorOf<Word> centralizerOfElt( const Word& w ) const
protected:
  OneRelatorGroupWithTorsion( OneRelatorGroupWithTorsionRep *newrep )
    : DerivedObjectOf<OneRelatorGroup, OneRelatorGroupWithTorsionRep>(newrep)

};
```

3.13 AProducts/include/ORProblems.h

— ORProblems.h —

```
#include "FPGroup.h"

\getchunk{class ORProblems}
```

3.13.1 class ORProblems

— class ORProblems —

```
class ORProblems
{
public:
  ORProblems( int numOfGens, const Word& relator )
    : numberOfGenerators( numOfGens ),
      theRelator( relator )
  bool isTrivial( ) const;
  bool isAbelian( ) const;
  bool isFinite( ) const;
  bool isFree( ) const;
  int order( ) const;
private:
  int numberOfGenerators;
  Word theRelator;
};
```

———-

3.14 AProducts/include/Range.h

3.14.1 struct Range

— **struct Range** —

```
struct Range
{
  int low, high;
  bool Default;
  Range() : low(0), high(0), Default(true)
  Range( int lo, int hi ) : low(lo), high(hi), Default(false)
  bool operator == ( Range r ) const
};
  void operator < ( ostream& ostr, const Range& g );
  void operator > ( istream& istr, Range& g );
  ostream& operator << ( ostream& ostr, const Range& g );
  istream& operator >> ( istream& istr, Range& g );
```

———-

3.15 AProducts/include/ShortenByRelators2.h

— **ShortenByRelators2.h** —

```
#include "SymmetricRelators.h"
#include "QuickAssociations.h"
#include "AP-fixups.h"

\getchunk{class ShortenByRelators2}
```

———-

3.15.1 class ShortenByRelators2

— **class ShortenByRelators2** —

```
class ShortenByRelators2
{
public:
  ShortenByRelators2 ( const SetOf<Word>& relators );
  Word getShortenWord( const Word& w ) const;
  Word expressWordInConjugatesOfRelators( const Word& w,
    ProductOfRelatorConjugates& productOfRelatorConjugates ) const;
```

```
private:
  virtual int compare( const Word& w1, const Word& w2 ) const;
  QuickAssociationsOf< Word, int > relatorsPieces;
  VectorOf<Word> theRelators;
  VectorOf<int> relatorsLengths;
  int base;
};
```

———-

3.16 AProducts/include/SubgroupOfOneRelatorGroup.h

— **SubgroupOfOneRelatorGroup.h** —

```
#include "OneRelatorGroup.h"
#include "OneRelatorGroupWithTorsion.h"

\getchunk{class ORGSubgroup}
\getchunk{class SubgroupOfOneRelatorGroup}
\getchunk{class SubgroupOfORGroupWithTorsion}
```

———-

3.16.1 class ORGSubgroup

— **class ORGSubgroup** —

```
class ORGSubgroup
{
public:
  ORGSubgroup( const VectorOf<Word>& gens );
  virtual ~ORGSubgroup( )
  virtual ORGSubgroup *copy( ) const = 0;
  virtual const OneRelatorGroup& parentGroup( ) const = 0;
  const VectorOf<Word>& generators() const
  bool isMagnusSubgroup( ) const;
  virtual bool contains( const Word& w ) const = 0;
  virtual Word rewriteFromGroupToSubgroupGenerators(const Word& w) const = 0 ;
  Word rewriteFromSubgroupToGroupGenerators( const Word& w ) const;
  bool lexicallyContains( const Word& w ) const;
  bool lexicallyContains( const Word& w, Word& wInBasis ) const;
  virtual bool operator == ( const ORGSubgroup& S ) const
  friend ostream& operator << ( ostream& ostr, const ORGSubgroup& S )
  friend ostream& operator < ( ostream& ostr, const ORGSubgroup& S )
  friend istream& operator > ( istream& istr, ORGSubgroup& S )
protected:
```

```
  virtual void write( ostream& ostr ) const;
  virtual void read( istream& istr );
private:
  virtual void printOn( ostream& ostr ) const;
  VectorOf<Word> theGenerators;
};
```

———-

3.16.2 class SubgroupOfOneRelatorGroup

— class SubgroupOfOneRelatorGroup —

```
class SubgroupOfOneRelatorGroup : public ORGSubgroup
{
public:
  SubgroupOfOneRelatorGroup( const OneRelatorGroup& G,
                             const VectorOf<Word>& gens );
  ORGSubgroup *copy( ) const
  const OneRelatorGroup& parentGroup() const
  bool contains( const Word& w ) const;
  bool contains( const Word& w, bool keepDetails ) const;
  Word rewriteFromGroupToSubgroupGenerators( const Word& w ) const;
  ProductOfRelatorConjugates getIdentityProductDecomposition( const Word& w )
    const;
  SubgroupOfOneRelatorGroup join( const SubgroupOfOneRelatorGroup& sg ) const;
  bool operator == ( const SubgroupOfOneRelatorGroup& S ) const
private:
  void write( ostream& ostr ) const;
  void read( istream& istr );
private:
  OneRelatorGroup theParentGroup;

\getchunk{struct ContainmentProblemData}

  ContainmentProblemData containsResult;
  void setContainmentProblemData( const Word& w, bool in_subgroup,
          const Word& wInBasis, bool keepDetails,
          const ProductOfRelatorConjugates& product ) const;
  bool subgroupsDecomposition(const Word& w,
          const SubgroupOfOneRelatorGroup& T,
          const SubgroupOfOneRelatorGroup& A,
          Word& t_part, Word& a_part,
          const bool keepDetails,
          ProductOfRelatorConjugates& prodDeco ) const;
  bool pureContainmentProblem( const Word& u, bool keepDetails = false ) const;
  class MagnusBreakdown buildMagnusBreakdown( const Word& w,
    SetOf<Generator>& subGens ) const;
```

```
};
```

3.16.3 struct ContainmentProblemData

— struct ContainmentProblemData —

```
struct ContainmentProblemData
{
  Word wordInParentGroupGenerators;
  bool inSubgroup;
  Word wordInSubgroupGenerators;
  ProductOfRelatorConjugates idProductDeco;
  bool keepDetails;
  ContainmentProblemData()
    : wordInParentGroupGenerators( ), inSubgroup( true ),
      wordInSubgroupGenerators( ), idProductDeco( ),
      keepDetails( false )
  ContainmentProblemData( const Word& w, bool f, const Word& u,
      bool keep_details, const ProductOfRelatorConjugates& prodDeco )
      : wordInParentGroupGenerators(w), inSubgroup(f),
        wordInSubgroupGenerators(u), idProductDeco( prodDeco ),
       keepDetails( keep_details )
  };
```

3.16.4 class SubgroupOfORGroupWithTorsion

— class SubgroupOfORGroupWithTorsion —

```
class SubgroupOfORGroupWithTorsion : public ORGSubgroup
{
public:
  SubgroupOfORGroupWithTorsion( const OneRelatorGroupWithTorsion& G,
                                const VectorOf<Word>& gens );
  ORGSubgroup *copy( ) const
  const OneRelatorGroup& parentGroup() const
  bool contains( const Word& w ) const;
  Word rewriteFromGroupToSubgroupGenerators( const Word& w ) const;
  bool operator == ( const SubgroupOfORGroupWithTorsion& S ) const
private:
  void write( ostream& ostr ) const;
  void read( istream& istr );
  bool contains( const Word& w, Word& wInBasis ) const;
  OneRelatorGroupWithTorsion theParentGroup;
```

```
};
```

———-

3.17 AProducts/include/SuperGen.h

— **SuperGen.h** —

```
#include "Generator.h"

\getchunk{class SuperGen}
```

———-

3.17.1 class SuperGen

— **class SuperGen** —

```
class SuperGen
{
public:
  SuperGen() : gen(1), exp(0)
  SuperGen( const Generator& g, int exponent = 1 ) : gen(g), exp(exponent)
  Generator generator() const
  int exponent() const
  bool operator == ( const SuperGen& g ) const
private:
  Generator gen;
  int exp;
};

  void operator < ( ostream& ostr, const SuperGen& g );
  void operator > ( istream& istr, SuperGen& g );
  ostream& operator << ( ostream& ostr, const SuperGen& g );
  istream& operator >> ( istream& istr, SuperGen& g );
```

———-

3.18 AProducts/include/Whitehead.h

— **Whitehead.h** —

```
#include "Set.h"
#include "List.h"
#include "Word.h"
```

```
#include "Automorphism.h"

typedef enum { WA_INVERSE, WA_PERMUTATION, WA_LEFT_MULT, WA_RIGHT_MULT }
        WhiteheadAutoType;

\getchunk{class ElementaryWhiteheadAuto}
\getchunk{class WhiteheadAuto}

  inline bool operator==(const WhiteheadAuto& u, const WhiteheadAuto& w)
  inline Word operator | (Word& w, ElementaryWhiteheadAuto& ewa)
  inline Word operator | (const Word& w, const WhiteheadAuto& wa)
  inline WhiteheadAuto operator | (const WhiteheadAuto& wa1,
                                   const WhiteheadAuto& wa2)
  inline WhiteheadAuto& operator |= ( WhiteheadAuto& wa1,
                                      const WhiteheadAuto& wa2)
  inline ostream& operator << ( ostream& ostr, const WhiteheadAuto& wa )
  WhiteheadAuto whiteheadDecomposition( const VectorOf<Word>& vec );
```

———-

3.18.1 class ElementaryWhiteheadAuto

— class ElementaryWhiteheadAuto —

```
class ElementaryWhiteheadAuto
{
public:
  ElementaryWhiteheadAuto( const Generator& x );
  ElementaryWhiteheadAuto( const Generator& x, const Generator& y,
                           const WhiteheadAutoType& wat );
  Generator x( ) const;
  Generator y( ) const;
  Word imageOf( const Word& w ) const;
  ElementaryWhiteheadAuto inverse( ) const;
  WhiteheadAutoType type( ) const
  void printOn( ostream& ostr ) const;
  void printOn( ostream& ostr, const VectorOf<Chars>& names ) const;
private:
  void printGenerator( ostream& ostr, const Generator& g ) const;
  void printGenerator( ostream& ostr, const Generator& g,
                       const VectorOf<Chars>& names ) const;
  WhiteheadAutoType theType;
  Generator g1, g2;
};

inline ostream& operator << ( ostream& ostr,
                              const ElementaryWhiteheadAuto& aut )
inline bool operator == ( const ElementaryWhiteheadAuto& u,
```

```
                          const ElementaryWhiteheadAuto& w )
inline bool operator != ( const ElementaryWhiteheadAuto& u,
                          const ElementaryWhiteheadAuto& w )
```

———-

3.18.2 class WhiteheadAuto

— class WhiteheadAuto —

```
class WhiteheadAuto
{
public:
  WhiteheadAuto()
  WhiteheadAuto(const ElementaryWhiteheadAuto& a) : autoList(a)
  WhiteheadAuto(const ListOf<ElementaryWhiteheadAuto>& list)
    : autoList(list)
  ListOf<ElementaryWhiteheadAuto> getAutoList() const
  WhiteheadAuto inverse() const;
  Word imageOf(const Word& w) const;
  Automorphism makeAutomorphism(const FGGroup& G) const;
  void printOn( ostream& ostr ) const;
private:
  ListOf<ElementaryWhiteheadAuto> autoList;
};
```

———-

4 The Elt classes

4.1 Elt/include/AbelianWord.h

— AbelianWord.h —

```
#include "Integer.h"
#include "Vector.h"
#include "PureRep.h"
#include "Word.h"

\getchunk{class AbelianWordRep}
\getchunk{class AbelianWord}
```

———-

4.1.1 class AbelianWordRep

— class AbelianWordRep —

```
class AbelianWordRep : public PureRep
{
public:
  AbelianWordRep( )
  AbelianWordRep( int numOfGens, const Word& w );
  AbelianWordRep( const VectorOf<Integer>& v ) : thePowers( v )
  bool operator == ( const AbelianWordRep& w ) const;
  Integer operator [] ( int i ) const;
  Integer& operator [] ( int i );
  AbelianWordRep* operator * ( const AbelianWordRep& aw ) const;
  int numberOfGenerators( ) const
  VectorOf<Integer> getPowers( ) const
  Word getWord( ) const;
  Integer fullLength( ) const;
  AbelianWordRep inverse( ) const;
  bool isTrivial( ) const;
  void printOn( ostream& ) const;
  void write( ostream& ostr ) const;
  void read( istream& istr );
  AbelianWordRep* clone( ) const
private:
  VectorOf<Integer> thePowers;
};
```

———-

4.1.2 class AbelianWord

— class AbelianWord —

```
class AbelianWord : public ObjectOf<AbelianWordRep>
{
public:
  AbelianWord( )
    : ObjectOf<AbelianWordRep>( new AbelianWordRep( ) )
  AbelianWord( int numOfGens, const Word& w )
    : ObjectOf<AbelianWordRep>( new AbelianWordRep(numOfGens,w) )
  AbelianWord( const VectorOf<Integer>& v )
    : ObjectOf<AbelianWordRep>( new AbelianWordRep(v) )
  bool operator == ( const AbelianWord& w ) const
  int operator != ( const AbelianWord& w ) const
  Integer operator [] ( int i ) const
  Integer& operator [] ( int i )
  AbelianWord operator * ( const AbelianWord& w ) const
```

```
  int numberOfGenerators( ) const
  VectorOf<Integer> getPowers( ) const
  Word getWord( ) const
  Integer fullLength( ) const
  AbelianWord inverse( ) const
  bool isTrivial( ) const
  friend ostream& operator << ( ostream& ostr, const AbelianWord& w )
  friend ostream& operator < ( ostream& ostr, const AbelianWord& w )
  friend istream& operator > ( istream& istr, AbelianWord& w )
protected:
  AbelianWord( AbelianWordRep* newrep )
    : ObjectOf<AbelianWordRep>(newrep)
};
```

———-

4.2 Elt/include/Elt.h

— **Elt.h** —

```
#include "EltRep.h"
#include "Integer.h"
#include "Rational.h"

\getchunk{class Elt}
```

———-

4.2.1 class Elt

— **class Elt** —

```
class Elt : public GenericObject
{
public:
  Elt( ) : GenericObject( new EltIdentityRep )
  int operator == ( const Elt& e ) const
  int operator != ( const Elt& e ) const
  static Type type( )
  Type actualType( ) const
  int hash( ) const
  Elt operator * ( const Elt& e ) const
  Elt operator *= ( const Elt& e )
  Elt raiseToPower( Integer power ) const;
  Elt inverse() const
  friend Elt inverse( const Elt& e )
```

```
  Elt conjugateBy( const Elt& e ) const
  friend Elt commutator( const Elt&, const Elt& );
  friend ostream& operator << ( ostream&, const Elt& );
  void debugPrint( ostream& o ) const;
  friend ostream& operator < ( ostream& ostr, const Elt& e )
  friend istream& operator > ( istream& istr, Elt& e )
protected:
  const EltRep* look( ) const
  EltRep* enhance( ) const
  EltRep* change( )
  Elt( EltRep* p ) : GenericObject(p)

};
```

———-

4.3 Elt/include/EltRep.h

— **EltRep.h** —

```
#include "GenericObject.h"
#include "Type.h"

\getchunk{struct EltRep}
\getchunk{struct EltIdentityRep}
```

———-

4.3.1 struct EltRep

— **struct EltRep** —

```
struct EltRep : GenericRep
{
  virtual ~EltRep( )
  static const Type theEltType;
  static Type type( )
  virtual Type actualType( ) const
  PureRep* clone( ) const = 0;
  virtual Bool operator == ( const EltRep& ) const = 0;
  virtual int hash( ) const = 0;
  virtual EltRep* operator * ( const EltRep& ) const = 0;
  virtual EltRep* inverse( ) const = 0;
  virtual EltRep* conjugateBy( const EltRep* ep ) const
  virtual EltRep* commutatorWith( const EltRep* ep ) const
  virtual void printOn(ostream&) const;
```

```
  virtual void debugPrint(ostream&) const;
  virtual void write( ostream& ostr ) const
  virtual void  read( istream& istr )
private:
  EltRep& operator = ( const EltRep& );
};
```

———-

4.3.2 struct EltIdentityRep

— **struct EltIdentityRep** —

```
struct EltIdentityRep : EltRep
{
  PureRep* clone( ) const
  static const Type theEltIdentityType;
  static Type type( )
  Type actualType( ) const
  virtual Bool operator == ( const EltRep& ) const
  virtual int hash( ) const
  virtual EltRep* operator * ( const EltRep& e ) const
  virtual EltRep* inverse( ) const

};
```

———-

4.4 Elt/include/Generator.h

— **Generator.h** —

```
\getchunk{class Generator}


inline Generator inv( Generator g )
inline int operator == (Generator g, Generator h)
inline int operator != (Generator g, Generator h)
inline int operator < (Generator g, Generator h)
inline int operator <= (Generator g, Generator h)
inline int operator > (Generator g, Generator h)
inline int operator >= (Generator g, Generator h)
inline ostream& operator << ( ostream& ostr, Generator g )
inline int power(Generator g)
inline int abs(Generator g)
```

```
inline int shortLexIndex(Generator g)
\subsubsection{class Generator}
\begin{chunk}{class Generator}
class Generator
{
public:
  Generator( ) : genrep(0)
  Generator( int i ) : genrep(i)
  inline friend int ord( Generator g )
  int hash() const
  friend ostream& operator < ( ostream& ostr, const Generator& g )
  friend istream& operator > ( istream& istr, Generator& g )
private:
  int genrep;
};
```

———-

4.5 Elt/include/NormalRandomWord.h

— **NormalRandomWord.h** —

```
#include "Word.h"
#include "RandomNumbers.h"

\getchunk{class NormalRandomWord}
```

———-

4.5.1 class NormalRandomWord

— **class NormalRandomWord** —

```
class NormalRandomWord
{
public:
  NormalRandomWord(int numberOfGenerators, int meanLength, int lengthStddev,
                   int lengthSeed, int generatorSeed)
    : numGens(numberOfGenerators), mean(meanLength), stddev(lengthStddev),
              G(generatorSeed), L(lengthSeed)
  Word word( );
  friend ostream& operator < ( ostream& ostr, const NormalRandomWord& nRW )
  friend istream& operator > ( istream& istr, NormalRandomWord& nRW )
private:
  int numGens, mean, stddev;
  UniformRandom G;
```

```
  NormalRandom L;
};
```

———-

4.6 Elt/include/WordData.h

— **WordData.h** —

```
#include "Generator.h"
#include "List.h"
#include "Vector.h"

\getchunk{class WordData}
```

———-

4.6.1 class WordData

— **class WordData** —

```
class WordData
{
public:
  typedef int GeneratorType;
  typedef GeneratorType* GeneratorPtrType;
  typedef const GeneratorType* cGeneratorPtrType;
  WordData( const WordData& wd ) :
    len(wd.len), wrd(new GeneratorType[wd.len])
  ~WordData( )
  WordData( int le )
  WordData( const Generator& x ) : len(1), wrd(new GeneratorType[1])
  WordData( const Generator& x, const Generator& y ) :
    len(2), wrd(new GeneratorType[2])
  WordData( const VectorOf<Generator>& v )
  WordData( const ListOf<Generator>& l )
  WordData( const GeneratorType *p, int le )
  int length( ) const
  GeneratorType& ref(int i)
  GeneratorType val(int i) const
  GeneratorPtrType first( )
  cGeneratorPtrType cFirst( ) const
  GeneratorPtrType last( )
  cGeneratorPtrType cLast( ) const
private:
  int len;
```

```
  static const int MAXLENGTH = 2000000000;
  GeneratorType* wrd;
  friend class WordRep;
};
```

———-

4.7 Elt/include/Word.h

— **Word.h** —

```
#include "Elt.h"
#include "WordRep.h"
#include "Generator.h"
#include "Vector.h"
#include "List.h"
#include "Chars.h"
#include "DerivedObjectOf.h"
#include "Set.h"

\getchunk{class Genref}
\getchunk{class Word}

inline Generator Word::operator [] ( int i ) const
inline Genref Word::operator [] ( int i )
istream& operator>>( istream& istr, Word& );
int maximalRoot(const Word& w);
SetOf<Word>& closeUnderInverses(SetOf<Word>& S);
SetOf<Word>& closeUnderCyclicPermutations(SetOf<Word>& S);
inline SetOf<Word>& symmetrise(SetOf<Word>& S)
int cancellationLambda( const SetOf<Word>& ss );
Trichotomy hasMetricSmallCancellation(const SetOf<Word>& S, const int lambda);
```

———-

4.7.1 class Word

— **class Word** —

```
class Word : public DerivedObjectOf<Elt,WordRep>
{
public:
  Word( ) : DerivedObjectOf<Elt,WordRep>(new WordRep(0))
  Word( const VectorOf<Generator>& v ) :
    DerivedObjectOf<Elt,WordRep>(new WordRep(v))
  Word( const ListOf<Generator>& l ) :
```

```
    DerivedObjectOf<Elt,WordRep>(new WordRep(l))
  Word( const Generator& g ) :
    DerivedObjectOf<Elt,WordRep>( new WordRep(g) )
  Word( const Elt& e ) : DerivedObjectOf<Elt,WordRep>( e )
  static Type type( )
  int length( ) const
  Generator operator [] ( int i ) const;
  Genref operator [] ( int i );
  Bool operator < ( const Word& ) const;
  Word nextInShortLex(int numberOfGens) const;
  Word nextFreelyReduced(int numberOfGens) const;
  Word nextCyclicallyReduced(int numberOfGens) const;
  Word subword(const int i, const int j) const;
  Word initialSegment(const int i) const
  Word terminalSegment(const int i) const
  Word findAgreement(const Word&) const;
  int agreementLength( const Word& ) const;
  Word shortenByRelator(const Word&) const;
  int numberOfOccurrences(const Generator& g) const;
  int exponentSum(const Generator& g) const;
  bool allExponentSumsZero( ) const;
  bool isProperPower( ) const;
  Generator maxOccurringGenerator( ) const;
  Word replaceGeneratorWithWord(const Generator&, const Word&) const;
  Elt replaceGenerators( const VectorOf<Elt>& eltvec ) const;
  Word replaceGenerators( const VectorOf<Word>& eltvec ) const;
  Word replaceSubword(const int i, const int j, const Word& w) const;
  Word freelyReduce( ) const;
  Word cyclicallyReduce(void) const;
  Word cyclicallyPermute(const int j) const;
  Word inverse( ) const
  Word operator * ( const Word& w ) const
  Word operator * ( const Generator& x )
  inline friend Word operator * ( const Generator& x, const Word& w )
  inline friend Word operator * ( const Generator& x, const Generator& y )
  Word operator *= ( const Word& w )
  Word operator *= ( const Generator& x )
  static Word wordByLexRank( int numGens, int lexRank );
  static Word wordByLexRank( VectorOf<int> vi )

\getchunk{class EmptyWord}
\getchunk{class ProductJunctor}

private:
  typedef WordData::GeneratorPtrType GeneratorPtrType;
  typedef WordData::cGeneratorPtrType cGeneratorPtrType;
  typedef WordData::GeneratorType GeneratorType;
  friend class Genref;
  Word( int len ) : DerivedObjectOf<Elt,WordRep>( new WordRep(len) )
  Word( const GeneratorType* p, int len ) :
```

```
    DerivedObjectOf<Elt,WordRep>( new WordRep(p, len) )
  Word( EltRep* rep ) : DerivedObjectOf<Elt,WordRep>((WordRep*)rep)

};
```

———-

4.7.2 class EmptyWord

— class EmptyWord —

```
class EmptyWord
{
  Chars emptyWord;
public:
  EmptyWord( const Chars& ew ) : emptyWord(ew)
  operator Chars( ) const
};
```

———-

4.7.3 class ProductJunctor

— class ProductJunctor —

```
class ProductJunctor
{
  Chars junctor;
public:
  ProductJunctor( const Chars& j ) : junctor(j)
  operator Chars( ) const
};
```

———-

4.7.4 class Genref

— class Genref —

```
class Genref
{
public:
  Generator operator = ( const Generator& g )
  int operator == ( const Generator& g )
  operator Generator( )
```

```
private:
  friend class Word;
  Genref( Word& w, int i ) : wref(w), index(i)
  Word& wref;
  int index;
  Genref( const Genref& g ) : wref(g.wref), index(g.index)
  Genref operator = ( const Genref& g )
};
```

———-

4.8 Elt/include/WordRep.h

— **WordRep.h** —

```
#include "EltRep.h"
#include "WordData.h"

\getchunk{struct WordRep}
```

———-

4.8.1 struct WordRep

— **struct WordRep** —

```
struct WordRep : EltRep, WordData
{
  WordRep( int len ) : WordData(len)
  WordRep( const Generator& x ) : WordData(x)
  WordRep( const Generator& x, const Generator& y ) : WordData(x, y)
  WordRep( const VectorOf<Generator>& v ) : WordData(v)
  WordRep( const ListOf<Generator>& l ) : WordData(l)
  WordRep( const GeneratorType *p, int len ) : WordData(p, len)
  PureRep* clone( ) const
  static const Type theWordType;
  static Type type( )
  Type actualType( ) const
  Bool operator == ( const EltRep& ) const;
  int hash() const;
  EltRep* operator * ( const EltRep& a ) const;
  EltRep* inverse() const;
  EltRep* rightMultBy( const Generator& x ) const;
  EltRep* leftMultBy( const Generator& x ) const;
  EltRep* conjugateBy( const EltRep* ep ) const;
  EltRep* commutatorWith( const EltRep* ep ) const;
```

```
  void printOn(ostream&) const;
  void debugPrint(ostream&) const;
  void write( ostream& ostr ) const
  void read( istream& istr )
};
```

———-

5 The Enumerators classes

5.1 Enumerators/include/AutoEnumerator.h

— **AutoEnumerator.h** —

```
#include "Supervisor.h"
#include "SMEnumerator.h"
#include "SMList.h"
#include "SMListIterator.h"
#include "RandomAutoInFree.h"
#include "File.h"
#include "Word.h"
#include "SMFPGroup.h"
#include "fastProblems.h"
#include "OutMessages.h"

\getchunk{class AutEnumeratorARCer2}
\getchunk{class AutoEnumeratorProblem}
\getchunk{class FiniteAutoEnumeratorProblem}
\getchunk{class IsMapInListARCer}
\getchunk{class IsMapInList}
```

———-

5.1.1 class AutEnumeratorARCer2

— **class AutEnumeratorARCer2** —

```
class AutEnumeratorARCer2 : public EnumeratorARCer
{
public:
  AutEnumeratorARCer2( ComputationManager& boss, SMListData& d )
    : EnumeratorARCer( boss, d ), randAuto( 0 ),  current( 0 )
  ~AutEnumeratorARCer2( )
  void setArguments( FreeGroup group, int avgNumbers, int n);
  void enumerate( );
  void writeResults( ostream& );
```

```
  void readResults( istream& );
  bool isAbelian, isFinite;
private:
  int current;
  RandomAutoInFree* randAuto;
  int number;
};
```

———-

5.1.2 class AutoEnumeratorProblem

— class AutoEnumeratorProblem —

```
class AutoEnumeratorProblem : public EnumeratorProblem< Map >
{
public:
  AutoEnumeratorProblem(SMFPGroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  Chars getDataFileName() const
  void start( )
  void terminate( )
  void resume( )
private:
  SMFPGroup& theGroup;
  AutEnumeratorARCer2 arcer;
};
```

———-

5.1.3 class FiniteAutoEnumeratorProblem

— class FiniteAutoEnumeratorProblem —

```
class FiniteAutoEnumeratorProblem : public EnumeratorProblem< Map >
{
public:
  FiniteAutoEnumeratorProblem(SMFPGroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  Chars getDataFileName() const
  void start( )
  void terminate( )
  void resume( )
private:
  SMFPGroup& theGroup;
```

```
  AutEnumeratorARCer2 arcer;
};
```

———-

5.1.4 class IsMapInListARCer

— class IsMapInListARCer —

```
class IsMapInListARCer : public ARCer
{
public:
  IsMapInListARCer( ComputationManager& boss )
    : ARCer( boss ), result(0)
  ~IsMapInListARCer( )
  void setArguments( const SMList<Map>&, const Map&);
  bool isInList();
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SMListIterator<Map>* theIterator;
  Map theMap;
  bool result;
};
```

———-

5.1.5 class IsMapInList

— class IsMapInList —

```
class IsMapInList  : public Supervisor
{
public:
 IsMapInList( SMList<Map>&, const class SMMap& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class SMMap& theMap;
  SMList<Map>& theList;
  IsMapInListARCer arcer;
};
```

———-

5.2 Enumerators/include/FreeListProblems.h

— **FreeListProblems.h** —

```
#include "Supervisor.h"
#include "SMList.h"
#include "File.h"
#include "FPGroup.h"
#include "OutMessages.h"
#include "SMListIterator.h"
#include "ListProblems.h"

\getchunk{class SMListProperPowerInFreeARCer}
\getchunk{class SMListExtractProperPowerInFree}
\getchunk{class SMListCommutatorsInFreeARCer}
\getchunk{class SMListExtractCommutatorsInFree}
\getchunk{class SGListExtractOfRankARCer}
\getchunk{class SGListExtractOfRank}
\getchunk{class SGListExtractNormalARCer}
\getchunk{class SGListExtractNormal}
\getchunk{class SGListExtractMalnormalARCer}
\getchunk{class SGListExtractMalnormal}
\getchunk{class MapListExtractIAautoARCer}
\getchunk{class MapListExtractIAauto}
\getchunk{class MapListExtractInnerARCer}
\getchunk{class MapListExtractInner}
```

———-

5.2.1 class SMListProperPowerInFreeARCer

— **class SMListProperPowerInFreeARCer** —

```
class SMListProperPowerInFreeARCer : public ARCer
{
public:
  SMListProperPowerInFreeARCer( ComputationManager& boss )
    : ARCer( boss ),
      theIterator( NULL )
  void setArguments( const SMList<Word>& );
  const SMListData& getData() const
  void runComputation( );
  void writeResults( ostream& o )
  void readResults( istream& i)
private:
  SMListIterator<Word>* theIterator;
  FPGroup theGroup;
  SMListData theData;
```

```
};
```

———-

5.2.2 class SMListExtractProperPowerInFree

— class SMListExtractProperPowerInFree —

```
class SMListExtractProperPowerInFree  : public SMListSupervisor
{
public:
  SMListExtractProperPowerInFree( SMList<Word>& );
  ~SMListExtractProperPowerInFree()
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( )
private:
  SMList<Word>& theList;
  SMListProperPowerInFreeARCer arcer;
};
```

———-

5.2.3 class SMListCommutatorsInFreeARCer

— class SMListCommutatorsInFreeARCer —

```
class SMListCommutatorsInFreeARCer : public ARCer
{
public:
  SMListCommutatorsInFreeARCer( ComputationManager& boss )
    : ARCer( boss ),
      theIterator( NULL )
  void setArguments( const SMList<Word>& );
  const SMListData& getData() const
  void runComputation( );
  void writeResults( ostream& o )
  void readResults( istream& i)
private:
  SMListIterator<Word>* theIterator;
  FreeGroup theGroup;
  SMListData theData;
};
```

———-

5.2.4 class SMListExtractCommutatorsInFree

— class SMListExtractCommutatorsInFree —

```
class SMListExtractCommutatorsInFree  : public SMListSupervisor
{
public:
  SMListExtractCommutatorsInFree( SMList<Word>& );
  ~SMListExtractCommutatorsInFree()
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( )
private:
  SMList<Word>& theList;
  SMListCommutatorsInFreeARCer arcer;
};
```

———-

5.2.5 class SGListExtractOfRankARCer

— class SGListExtractOfRankARCer —

```
class SGListExtractOfRankARCer : public ARCer
{
public:
  SGListExtractOfRankARCer( ComputationManager& boss )
    : ARCer( boss ),
    minRank( 1 ),
    maxRank( 0 )
  ~SGListExtractOfRankARCer( )
  void setArguments( const SMList<Subgroup>&);
  const SMListData& getData() const
  void setMinimalRank(int  min )
  void setMaximalRank(int  max )
  int indexInFreeGroup( const VectorOf<Word>& s )const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SMListIterator<Subgroup>* theIterator;
  SMListData theData;
  const SMFPGroup* theGroup;
  int minRank;
  int maxRank;
};
```

———-

5.2.6 class SGListExtractOfRank

— class SGListExtractOfRank —

```
class SGListExtractOfRank  : public Supervisor
{
public:
 SGListExtractOfRank( SMList<Subgroup>& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( )
private:
  SMList<Subgroup>& theList;
  bool started;
  SGListExtractOfRankARCer arcer;
};
```

———-

5.2.7 class SGListExtractNormalARCer

— class SGListExtractNormalARCer —

```
class SGListExtractNormalARCer : public ARCer
{
public:
  SGListExtractNormalARCer( ComputationManager& boss )
    : ARCer( boss )
  ~SGListExtractNormalARCer( )
  void setArguments( const SMList<Subgroup>&);
  const SMListData& getData() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SMListIterator<Subgroup>* theIterator;
  SMListData theData;
  FreeGroup theGroup;
};
```

———-

5.2.8 class SGListExtractNormal

— class SGListExtractNormal —

```
class SGListExtractNormal  : public Supervisor
{
public:
  SGListExtractNormal( SMList<Subgroup>& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( )
private:
  SMList<Subgroup>& theList;
  bool started;
  SGListExtractNormalARCer arcer;
};
```

———-

5.2.9 class SGListExtractMalnormalARCer

— class SGListExtractMalnormalARCer —

```
class SGListExtractMalnormalARCer : public ARCer
{
public:
  SGListExtractMalnormalARCer( ComputationManager& boss )
    : ARCer( boss )
  ~SGListExtractMalnormalARCer( )
  void setArguments( const SMList<Subgroup>&);
  const SMListData& getData() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SMListIterator<Subgroup>* theIterator;
  SMListData theData;
  FreeGroup theGroup;
};
```

———-

5.2.10 class SGListExtractMalnormal

— class SGListExtractMalnormal —

```
class SGListExtractMalnormal  : public Supervisor
{
public:
  SGListExtractMalnormal( SMList<Subgroup>& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( )
private:
  SMList<Subgroup>& theList;
  bool started;
  SGListExtractMalnormalARCer arcer;
};
```

———-

5.2.11 class MapListExtractIAautoARCer

— class MapListExtractIAautoARCer —

```
class MapListExtractIAautoARCer : public ARCer
{
public:
  MapListExtractIAautoARCer( ComputationManager& boss )
    : ARCer( boss )
  ~MapListExtractIAautoARCer( )
  void setArguments( const SMList<Map>&);
  const SMListData& getData() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SMListIterator<Map>* theIterator;
  SMListData theData;
  FreeGroup theGroup;
};
```

———-

5.2.12 class MapListExtractIAauto

— class MapListExtractIAauto —

```
class MapListExtractIAauto  : public Supervisor
{
public:
  MapListExtractIAauto( SMList<Map>& );
```

```
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( )
private:
  SMList<Map>& theList;
  bool started;
  MapListExtractIAautoARCer arcer;
};
```

———-

5.2.13 class MapListExtractInnerARCer

— class MapListExtractInnerARCer —

```
class MapListExtractInnerARCer : public ARCer
{
public:
  MapListExtractInnerARCer( ComputationManager& boss )
    : ARCer( boss )
  ~MapListExtractInnerARCer( )
  void setArguments( const SMList<Map>&);
  const SMListData& getData() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SMListIterator<Map>* theIterator;
  SMListData theData;
  FreeGroup theGroup;
};
```

———-

5.2.14 class MapListExtractInner

— class MapListExtractInner —

```
class MapListExtractInner  : public Supervisor
{
public:
  MapListExtractInner( SMList<Map>& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( )
```

```
private:
  SMList<Map>& theList;
  bool started;
  MapListExtractInnerARCer arcer;
};
```

———-

5.3 Enumerators/include/HomEnumerators.h

— **HomEnumerators.h** —

```
#include "Supervisor.h"
#include "SMEnumerator.h"
#include "WordEnumerator.h"
#include "File.h"
#include "Word.h"
#include "SMFPGroup.h"
#include "RandomNumbers.h"
#include "OutMessages.h"

\getchunk{class HomEnumeratorARCer1}
\getchunk{class RandHomEnumeratorProblem}
\getchunk{class HomEnumeratorARCer2}
\getchunk{class TotalHomEnumeratorProblem}
```

———-

5.3.1 class HomEnumeratorARCer1

— **class HomEnumeratorARCer1** —

```
class HomEnumeratorARCer1 : public EnumeratorARCer
{
public:
  HomEnumeratorARCer1( ComputationManager& boss , SMListData& d , GIC& gic )
    : EnumeratorARCer( boss , d ), rangeGIC( gic ), current( 0 ), number( 0 ),
      avgLength( 0 ), lengthPicker(2112), genPicker(1812)
  void setArguments( FPGroup g , FPGroup h , int avgNumbers , int n );
  void setParams( int k , int n )
  Map getMap();
  bool extendToHom( Map& );
  void enumerate( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
```

```
  int  current;
  int avgLength;
  int number;
  NormalRandom lengthPicker;
  UniformRandom genPicker;
  GIC& rangeGIC;
  FPGroup G;
  FPGroup H;
};
```

———-

5.3.2 class RandHomEnumeratorProblem

— class RandHomEnumeratorProblem —

```
class RandHomEnumeratorProblem : public EnumeratorProblem< Map >
{
public:
  RandHomEnumeratorProblem( SMFPGroup& , SMFPGroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  Chars getDataFileName() const
  void start( );
  void terminate( )
  void resume( )
private:
  SMFPGroup& G;
  SMFPGroup& H;
  HomEnumeratorARCer1 arcer;
  bool init;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
};
```

———-

5.3.3 class HomEnumeratorARCer2

— class HomEnumeratorARCer2 —

```
class HomEnumeratorARCer2 : public EnumeratorARCer
{
public:
  HomEnumeratorARCer2( ComputationManager& boss , SMListData& d , GIC& gic )
    : EnumeratorARCer( boss , d ), rangeGIC( gic ), current( 0 ),
      number( 0 ), we( FreeGroup() )
```

```
  void setArguments( FPGroup G , FPGroup H , int n );
  void setParam( int k )
  Map nextMap();
  bool extendToHom( Map& );
  void enumerate( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  WordEnumerator we;
  int number;
  int current;
  FPGroup G;
  FPGroup H;
  class GIC& rangeGIC;
};
```

———-

5.3.4 class TotalHomEnumeratorProblem

— class TotalHomEnumeratorProblem —

```
class TotalHomEnumeratorProblem : public EnumeratorProblem< Map >
{
public:
  TotalHomEnumeratorProblem( SMFPGroup& , SMFPGroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  Chars getDataFileName() const
  void start( );
  void terminate( )
  void resume( )
private:
  SMFPGroup& G;
  SMFPGroup& H;
  HomEnumeratorARCer2 arcer;
  bool init;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
};
```

———-

5.4 Enumerators/include/ListProblems.h

— ListProblems.h —

```
#include "Supervisor.h"
#include "fastProblems.h"
#include "SMList.h"
#include "SMListIterator.h"

\getchunk{class SMListSupervisor}
\getchunk{template <class T> class SMListJoinARCer}
\getchunk{template <class T> class SMListJoin}
\getchunk{template <class T> class MakeSMListOf}
```

———-

5.4.1 class SMListSupervisor

— class SMListSupervisor —

```
class SMListSupervisor :  public Supervisor
{
 public:
  SMListSupervisor( SMList<Word>& l ) : Supervisor( true ), theList( l )
  SMList<Word>& getList()
 protected:
  SMList<Word>& theList;
};
```

———-

5.4.2 template ¡class T¿ class SMListJoinARCer

— template ¡class T¿ class SMListJoinARCer —

```
template <class T> class SMListJoinARCer : public ARCer
{
public:
  SMListJoinARCer( ComputationManager& boss )
    : ARCer( boss ), l2Data( NULL ), l1Data( NULL )
  ~SMListJoinARCer( )
  void setArguments( const SMList<T>&, const SMList<T>& );
  const SMListData getJoinData() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SMListData* l1Data;
  SMListData* l2Data;
  SMListData joinData;
};
```

———-

5.4.3 template ¡class T¿ class SMListJoin

— template ¡class T¿ class SMListJoin —

```
template <class T> class SMListJoin  : public Supervisor
{
public:
  SMListJoin( SMList<T>&, SMList<T>& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMList<T>& list1;
  SMList<T>& list2;
  bool started;
  SMListJoinARCer<T> arcer;
};
```

———-

5.4.4 template ¡class T¿ class MakeSMListOf

— template ¡class T¿ class MakeSMListOf —

```
template <class T> class MakeSMListOf : public FastComputation
{
public:
  MakeSMListOf( EnumeratorProblem<T>& e): enumerator( e )
  void takeControl( );
protected:
  EnumeratorProblem<T>& enumerator;
};
```

———-

5.5 Enumerators/include/ORConsequencesEnumerator.h

— ORConsequencesEnumerator.h —

```
#include "OneRelatorGroup.h"
#include "Supervisor.h"
#include "SMSubgroup.h"
#include "SMEnumerator.h"
```

```
#include "File.h"

\getchunk{class ORConsequencesEnumeratorARCer}
\getchunk{class ORConsequencesEnumerator}
```

5.5.1 class ORConsequencesEnumeratorARCer

— class ORConsequencesEnumeratorARCer —

```
class ORConsequencesEnumeratorARCer : public EnumeratorARCer
{
public:
  ORConsequencesEnumeratorARCer( ComputationManager& boss,SMListData& d,
         const OneRelatorGroup& g )
    : EnumeratorARCer( boss, d ), theEnumerator( g ), theGroup( g ),
      counter( 1 ),  numberOfAll( 100 )
  Chars getFileName() const
  void setNumberOfAll( int n )
  void writeResults( ostream& );
  void readResults( istream& );
protected:
  void enumerate();
private:
  EnumeratorOfConsequences theEnumerator;
  OneRelatorGroup theGroup;
  File file;
  int counter;
  int numberOfAll;
};
```

5.5.2 class ORConsequencesEnumerator

— class ORConsequencesEnumerator —

```
class ORConsequencesEnumerator : public  EnumeratorProblem< Word >
{
public:
  ORConsequencesEnumerator( SMFPGroup&);
  void viewStructure(ostream& ostr) const;
  class SMFPGroup& getGroup() const
  void takeControl( );
  void start( )
  void terminate( )
```

```
  void resume()
private:
  SMFPGroup& theSMFPGroup;
  ORConsequencesEnumeratorARCer arcer;
  bool started;
};
```

———-

5.6 Enumerators/include/REnumerator.h

— **REnumerator.h** —

```
#include "Supervisor.h"
#include "SMEnumerator.h"
#include "File.h"
#include "Word.h"
#include "SMFPGroup.h"
#include "OutMessages.h"
#include "RandomNumbers.h"

\getchunk{class NCEnumerateTotalARCer}
\getchunk{class NCEnumerateRandomARCer}
\getchunk{class NCRelatorEnumerator}
```

———-

5.6.1 class NCEnumerateTotalARCer

— **class NCEnumerateTotalARCer** —

```
class NCEnumerateTotalARCer : public EnumeratorARCer
{
public:
  NCEnumerateTotalARCer( ComputationManager& boss, SMListData& d,
       const FPGroup& group )
    : EnumeratorARCer( boss, d ), theGroup(group), numberOfAll(100),
      counter(0)
  void setTotalNumber(int n)
  void writeResults( ostream& ostr )
  void readResults( istream& istr)
protected:
  void enumerate();
private:
  FPGroup theGroup;
  int numberOfAll;
```

```
  Word currentWord;
  int counter;
};
```

———-

5.6.2 class NCEnumerateRandomARCer

— class NCEnumerateRandomARCer —

```
class NCEnumerateRandomARCer : public EnumeratorARCer
{
public:
  NCEnumerateRandomARCer( ComputationManager& boss, SMListData& d,
        const FPGroup& group )
    : EnumeratorARCer( boss, d ), theGroup(group), numberOfAll(100),
    maxConjLength(100), typeIsUniform( true ),  numberOfFactors( 10 ),
    counter(0)
  void setTotalNumber(int n)
  void setConjL(int n)
  void setNumOfFactors(int n)
  void setTypeUniform()
  void setTypeNormal()
  void writeResults( ostream& o )
  void readResults( istream& i )
protected:
  Word getRelConj();
  void enumerate();
private:
  FPGroup theGroup;
  int numberOfAll;
  int maxConjLength;
  bool typeIsUniform;
  int counter;
  int numberOfFactors;
  UniformRandom ur;
  NormalRandom nr;
};
```

———-

5.6.3 class NCRelatorEnumerator

— class NCRelatorEnumerator —

```
class NCRelatorEnumerator : public EnumeratorProblem< Word >
{
```

```
public:
  NCRelatorEnumerator(SMFPGroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( ) ;
  void terminate( )
  void resume( );
private:
  SMFPGroup& theGroup;
  EnumeratorARCer* arcer;
};
```

———-

5.7 Enumerators/include/SGREnumerator.h

— **SGREnumerator.h** —

```
#include "Supervisor.h"
#include "SMSubgroup.h"
#include "SMEnumerator.h"
#include "File.h"

\getchunk{class SGRelatorEnumeratorARCer}
\getchunk{class SGRelatorEnumerator}
```

———-

5.7.1 class SGRelatorEnumeratorARCer

— **class SGRelatorEnumeratorARCer** —

```
class SGRelatorEnumeratorARCer : public EnumeratorARCer
{
public:
  SGRelatorEnumeratorARCer( ComputationManager& boss, SMListData& d )
    : EnumeratorARCer( boss, d ), firstStart(true),  counter( 1 ),
      numberOfAll( 100 )
  void setArguments( const class SMFPGroup* group,
         const class SMSubgroup* subgroup );
  void setNumberOfAll( int n )
  bool ORisTrivial(const Word&  theTestWord);
  void writeResults( ostream& );
  void readResults( istream& );
  Chars getFileName() const
protected:
```

```
  void enumerate();
private:
  const class SMSubgroup* theSMSubgroup;
  const class SMFPGroup* theGroup;
  Word possibleRelator;
  bool firstStart;
  int counter;
  int numberOfAll;
  File file;
};
```

———-

5.7.2 class SGRelatorEnumerator

— class SGRelatorEnumerator —

```
class SGRelatorEnumerator : public  EnumeratorProblem< NoType >
{
public:
  SGRelatorEnumerator( SMSubgroup&);
  void viewStructure(ostream& ostr) const;
  class SMSubgroup& getSubgroup() const
  void takeControl( );
  void start( )
  void terminate( )
  void resume()
private:
  SMSubgroup& theSMSubgroup;
  SGRelatorEnumeratorARCer arcer;
  class GIC& theGIC;
  bool started;
  bool firstStart;
  bool resumed;
  bool useORwordProblem;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
};
```

———-

5.8 Enumerators/include/SMListIterator.h

— SMListIterator.h —

```
#include "SMList.h"
```

```
#include "Word.h"
#include "SMEnumerator.h"

\getchunk{template <class T> class SMListIterator}
\getchunk{class WriteEnumeratorElement}
\getchunk{class EnumWriteWord}
\getchunk{class EnumWriteMap}
\getchunk{class EnumWriteVectorOfWords}
\getchunk{class EnumWriteSetOfWords}
\getchunk{class EnumWriteChars}
```

———-

5.8.1 template ¡class T¿ class SMListIterator

— template ¡class T¿ class SMListIterator —

```
template <class T> class SMListIterator
{
public:
  SMListIterator(const SMList<T>& );
  ~SMListIterator()
  bool EOL() const
  void reset();
  bool nextCell();
  const T& getData();
  int getNumberOfElements()const
private:
  void deleteCurrentData();
  void parseData();
  Chars currentLine;
  T* currentData;
  FPGroup theGroup;
  int numberOfCurrent;
  SMListData theData;
  bool eol_reached;
};
```

———-

5.8.2 class WriteEnumeratorElement

— class WriteEnumeratorElement —

```
class WriteEnumeratorElement
{
 public:
```

```
  WriteEnumeratorElement()
  virtual void print(ostream& ostr) const = 0;
  inline friend ostream& operator << ( ostream& o,
                                       const WriteEnumeratorElement& we )
protected:
private:
  WriteEnumeratorElement(const WriteEnumeratorElement&);
};
```

———-

5.8.3 class EnumWriteWord

— class EnumWriteWord —

```
class EnumWriteWord : public WriteEnumeratorElement
{
public:
  EnumWriteWord( const Word& w, const FPGroup& g) :  theW( w ), theG( g )
  void print( ostream& o ) const;
private:
  Word theW;
  FPGroup theG;
};
```

———-

5.8.4 class EnumWriteMap

— class EnumWriteMap —

```
class EnumWriteMap : public WriteEnumeratorElement
{
public:
  EnumWriteMap( const Map& m ) : theM( m )
  void print( ostream& o ) const;
private:
  Map theM;
};
```

———-

5.8.5 class EnumWriteVectorOfWords

— class EnumWriteVectorOfWords —

```
class EnumWriteVectorOfWords : public WriteEnumeratorElement
{
public:
  EnumWriteVectorOfWords( const VectorOf<Word>& v, const FPGroup& g )
    : theV( v ), theG( g )
  void print( ostream& o ) const;
private:
  VectorOf<Word> theV;
  FPGroup theG;
};
```

———-

5.8.6 class EnumWriteSetOfWords

— class EnumWriteSetOfWords —

```
class EnumWriteSetOfWords : public WriteEnumeratorElement
{
public:
  EnumWriteSetOfWords( const SetOf<Word>& s, const FPGroup& g )
    : theS( s ), theG( g )
  void print( ostream& o ) const;
private:
  SetOf<Word> theS;
  FPGroup theG;
};
```

———-

5.8.7 class EnumWriteChars

— class EnumWriteChars —

```
class EnumWriteChars : public WriteEnumeratorElement
{
public:
  EnumWriteChars( const Chars& c ) : theC( c )
  void print( ostream& o ) const;
private:
  Chars theC;
};
```

———-

5.9 Enumerators/include/SMListSubgroupProblems.h

— **SMListSubgroupProblems.h** —

```
#include "Supervisor.h"
#include "SMEnumerator.h"
#include "SMList.h"
#include "File.h"
#include "Word.h"
#include "SMFPGroup.h"
#include "OutMessages.h"
#include "RandomNumbers.h"
#include "ListProblems.h"
#include "SMListIterator.h"

\getchunk{class SMListSGTrivialARCer}
\getchunk{class SMListSGTrivialComputation}
\getchunk{class SMListExtractTrivialSubgroups}
\getchunk{class SMListSGAbelianARCer}
\getchunk{class SMListSGAbelianComputation}
\getchunk{class SMListExtractAbelianSubgroups}
\getchunk{class SMListSGCentralARCer}
\getchunk{class SMListSGCentralComputation}
\getchunk{class SMListExtractCentralSubgroups}
```

———-

5.9.1 class SMListSGTrivialARCer

— **class SMListSGTrivialARCer** —

```
class SMListSGTrivialARCer : public ARCer
{
public:
  SMListSGTrivialARCer( ComputationManager& boss );
  void setArguments( SMList<Subgroup>*);
  void runComputation( );
  void writeResults( ostream& o)
  void readResults( istream& i)
  Chars getFileOfNonTrivial() const
  const SMListData& getTrivialData() const
private:
  SMListIterator<Subgroup>* theIterator;
  const class SMFPGroup* theGroup;
  class SMList<Subgroup>* theList;
  int tCounter;
  int ntCounter;
  File fileOfNonTrivial;
```

```
  SMListData trivialData;
};
```

———-

5.9.2 class SMListSGTrivialComputation

— class SMListSGTrivialComputation —

```
class SMListSGTrivialComputation  : public ComputationManager
{
public:
  SMListSGTrivialComputation( class SMListExtractTrivialSubgroups& );
  ~SMListSGTrivialComputation();
  Chars getFileOfNonTrivial() const
  const SMListData& getTrivialData() const
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMList<Subgroup>& theList;
  bool started;
  SMListSGTrivialARCer arcer;
};
```

———-

5.9.3 class SMListExtractTrivialSubgroups

— class SMListExtractTrivialSubgroups —

```
class SMListExtractTrivialSubgroups  : public Supervisor
{
public:
  SMListExtractTrivialSubgroups( SMList<Subgroup>& );
  ~SMListExtractTrivialSubgroups()
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  SMList<Subgroup>& getList()
  void start( );
  void terminate( )
private:
  SMList<Subgroup>& theList;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate normalClosure;
```

```
  Subordinate<SMListExtractTrivialSubgroups,
              SMListSGTrivialComputation> checker;
};
```

———-

5.9.4 class SMListSGAbelianARCer

— class SMListSGAbelianARCer —

```
class SMListSGAbelianARCer : public ARCer
{
public:
  SMListSGAbelianARCer( ComputationManager& boss );
  void setArguments( SMList<Subgroup>*);
  void runComputation( );
  void writeResults( ostream& o )
  void readResults( istream& i )
  Chars getFileOfNonAbelian() const
  const SMListData& getAbelianData() const
private:
  SMListIterator<Subgroup>* theIterator;
  const class SMFPGroup* theGroup;
  class SMList<Subgroup>* theList;
  File fileOfNonAbelian;
  SMListData abelianData;
  int tCounter;
  int ntCounter;
};
```

———-

5.9.5 class SMListSGAbelianComputation

— class SMListSGAbelianComputation —

```
class SMListSGAbelianComputation  : public ComputationManager
{
public:
  SMListSGAbelianComputation( class SMListExtractAbelianSubgroups& );
  ~SMListSGAbelianComputation();
  Chars getFileOfNonAbelian() const
  const SMListData& getAbelianData() const
  void takeControl( );
  void start( )
  void terminate( )
private:
```

```
  SMList<Subgroup>& theList;
  bool started;
  SMListSGAbelianARCer arcer;
};
```

———-

5.9.6 class SMListExtractAbelianSubgroups

— class SMListExtractAbelianSubgroups —

```
class SMListExtractAbelianSubgroups  : public Supervisor
{
public:
  SMListExtractAbelianSubgroups( SMList<Subgroup>& );
  ~SMListExtractAbelianSubgroups()
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  SMList<Subgroup>& getList()
  void start( );
  void terminate( )
private:
  SMList<Subgroup>& theList;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate normalClosure;
  Subordinate<SMListExtractAbelianSubgroups,
              SMListSGAbelianComputation> checker;
};
```

———-

5.9.7 class SMListSGCentralARCer

— class SMListSGCentralARCer —

```
class SMListSGCentralARCer : public ARCer
{
public:
  SMListSGCentralARCer( ComputationManager& boss );
  void setArguments( SMList<Subgroup>*);
  void runComputation( );
  void writeResults( ostream& o )
  void readResults( istream& i )
  Chars getFileOfNonCentral() const
  const SMListData& getCentralData() const
```

```
private:
  SMListIterator<Subgroup>* theIterator;
  const class SMFPGroup* theGroup;
  class SMList<Subgroup>* theList;
  File fileOfNonCentral;
  SMListData centralData;
  int tCounter;
  int ntCounter;
};
```

———-

5.9.8 class SMListSGCentralComputation

— class SMListSGCentralComputation —

```
class SMListSGCentralComputation  : public ComputationManager
{
public:
  SMListSGCentralComputation( class SMListExtractCentralSubgroups& );
  ~SMListSGCentralComputation();
  Chars getFileOfNonCentral() const
  const SMListData& getCentralData() const
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMList<Subgroup>& theList;
  bool started;
  SMListSGCentralARCer arcer;
};
```

———-

5.9.9 class SMListExtractCentralSubgroups

— class SMListExtractCentralSubgroups —

```
class SMListExtractCentralSubgroups  : public Supervisor
{
public:
  SMListExtractCentralSubgroups( SMList<Subgroup>& );
  ~SMListExtractCentralSubgroups()
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  SMList<Subgroup>& getList()
  void start( );
```

```
  void terminate( )
private:
  SMList<Subgroup>& theList;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate normalClosure;
  Subordinate<SMListExtractCentralSubgroups,
              SMListSGCentralComputation> checker;
};
```

———-

5.10 Enumerators/include/SMListWordProblem.h

— **SMListWordProblem.h** —

```
#include "Supervisor.h"
#include "SMEnumerator.h"
#include "SMList.h"
#include "File.h"
#include "Word.h"
#include "SMFPGroup.h"
#include "OutMessages.h"
#include "RandomNumbers.h"
#include "ListProblems.h"
#include "SMListIterator.h"

\getchunk{class SMListGeneticWPArcer}
\getchunk{class SMListGeneticWPCM}
\getchunk{class SMListWPCheckARCer}
\getchunk{class SMListWPCheckComputation}
\getchunk{class SMListWPPrinResultArcer}
\getchunk{class SMListExtractTrivialWords}
\getchunk{class SMListWordIsCentralARCer}
\getchunk{class SMListWordIsCentralComputation}
\getchunk{class SMListExtractCentralWords}
```

———-

5.10.1 class SMListGeneticWPArcer

— **class SMListGeneticWPArcer** —

```
class SMListGeneticWPArcer : public ARCer
{
```

```
public:
  SMListGeneticWPArcer( ComputationManager& );
  ~SMListGeneticWPArcer( )
  void setArguments( SMList<Word>* );
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SMListIterator<Word>* theIterator;
  SMList<Word>* theList;
};
```

———-

5.10.2 class SMListGeneticWPCM

— class SMListGeneticWPCM —

```
class SMListGeneticWPCM : public ComputationManager
{
public:
  SMListGeneticWPCM( class SMListSupervisor& sms );
  void takeControl( );
  void start( );
  void terminate( );
private:
  SMListGeneticWPArcer arcer;
};
```

———-

5.10.3 class SMListWPCheckARCer

— class SMListWPCheckARCer —

```
class SMListWPCheckARCer : public ARCer
{
public:
  SMListWPCheckARCer( ComputationManager& boss )
    : ARCer( boss ), theIterator( NULL ), theGroup( NULL )
  void setArguments( SMList<Word>*, const SMFPGroup*  );
  void runComputation( );
  void writeResults( ostream& )
  void readResults( istream& )
private:
  SMListIterator<Word>* theIterator;
  const class SMFPGroup* theGroup;
```

```
  class SMList<Word>* theList;
};
```

———-

5.10.4 class SMListWPCheckComputation

— class SMListWPCheckComputation —

```
class SMListWPCheckComputation  : public ComputationManager
{
public:
  SMListWPCheckComputation( SMListSupervisor& );
  ~SMListWPCheckComputation();
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMList<Word>& theList;
  bool started;
  SMListWPCheckARCer arcer;
};
```

———-

5.10.5 class SMListWPPrinResultArcer

— class SMListWPPrinResultArcer —

```
class SMListWPPrinResultArcer : public ARCer
{
public:
  SMListWPPrinResultArcer( ComputationManager& b) :
    ARCer( b ), theList( NULL )
  ~SMListWPPrinResultArcer( )
  void setArguments( SMList<Word>* );
  Chars getFileOfNonTrivial() const
  const SMListData& getTrivialData() const
  void runComputation( );
  void writeResults( ostream& o )
  void readResults( istream& i )
private:
  SMList<Word>* theList;
  File fileOfNonTrivial;
  SMListData trivialData;
};
```

———-

5.10.6 class SMListExtractTrivialWords

— class SMListExtractTrivialWords —

```
class SMListExtractTrivialWords  : public SMListSupervisor
{
public:
  SMListExtractTrivialWords( SMList<Word>& );
  ~SMListExtractTrivialWords()
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  bool finished( );
  void start( );
  void terminate( )
private:
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate normalClosure;
  Subordinate<SMListExtractTrivialWords, SMListGeneticWPCM> genetic;
  Subordinate<SMListExtractTrivialWords, SMListWPCheckComputation> checker;
  SMListWPPrinResultArcer arcer;
};
```

———-

5.10.7 class SMListWordIsCentralARCer

— class SMListWordIsCentralARCer —

```
class SMListWordIsCentralARCer : public ARCer
{
public:
  SMListWordIsCentralARCer( ComputationManager& boss );
  void setArguments( SMList<Word>*);
  void runComputation( );
  void writeResults( ostream& o )
  void readResults( istream& i )
  Chars getFileOfNonCentral() const
  const SMListData& getCentralData() const
private:
  SMListIterator<Word>* theIterator;
  const class SMFPGroup* theGroup;
  class SMList<Word>* theList;
  File fileOfNonCentral;
```

```
  SMListData centralData;
  int tCounter;
  int ntCounter;
};
```

———-

5.10.8 class SMListWordIsCentralComputation

— class SMListWordIsCentralComputation —

```
class SMListWordIsCentralComputation  : public ComputationManager
{
public:
  SMListWordIsCentralComputation( class SMListExtractCentralWords& );
  ~SMListWordIsCentralComputation();
  Chars getFileOfNonCentral() const
  const SMListData& getCentralData() const
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMList<Word>& theList;
  bool started;
  SMListWordIsCentralARCer arcer;
};
```

———-

5.10.9 class SMListExtractCentralWords

— class SMListExtractCentralWords —

```
class SMListExtractCentralWords  : public Supervisor
{
public:
  SMListExtractCentralWords( SMList<Word>& );
  ~SMListExtractCentralWords()
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  SMList<Word>& getList()
  void start( );
  void terminate( )
private:
  SMList<Word>& theList;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
```

```
  MirrorSubordinate agSupervisor;
  MirrorSubordinate normalClosure;
  Subordinate<SMListExtractCentralWords,
              SMListWordIsCentralComputation> checker;
};
```

———-

5.11 Enumerators/include/SubgroupEnumerator.h

— **SubgroupEnumerator.h** —

```
#include "Supervisor.h"
#include "SMEnumerator.h"
#include "SMList.h"
#include "Subgroup.h"
#include "SMFPGroup.h"
#include "OutMessages.h"
#include "RandomNumbers.h"
#include "SMListIterator.h"

\getchunk{class SGEnumeratorARCer}
\getchunk{class SGEnumeratorProblem}
\getchunk{class IsSubgroupInListARCer}
\getchunk{class IsSubgroupInList}
\getchunk{class SGListExtractOfIndexARCer}
\getchunk{class SGListExtractOfIndex}
```

———-

5.11.1 class SGEnumeratorARCer

— **class SGEnumeratorARCer** —

```
class SGEnumeratorARCer : public EnumeratorARCer
{
public:
  SGEnumeratorARCer( ComputationManager& boss,SMListData& d, SMFPGroup& g )
    : EnumeratorARCer( boss, d ), group(g), numberOfAll(100),
    enumerateAll( true ), isNormalRandom( false ), numberOfGenerators(1),
    maxLength(100),  counter(0)
  void setEnumerateAll()
  void setEnumerateRandom()
  void setNormalRandom()
  void setUniformRandom()
  void setTotalNumber(int n)
```

```
  void setMaxLength(int n)
  void setNumberOfGens( int n)
  void writeResults( ostream& ostr )
  void readResults( istream& istr)
protected:
  void enumerate();
private:
  void enumerateAllSubgroups( );
  Word getRandomWord() ;
  SMFPGroup& group;
  FreeGroup theGroup;
  bool enumerateAll;
  bool isNormalRandom;
  int numberOfGenerators;
  int numberOfAll;
  int maxLength;
  int counter;
  UniformRandom ur;
  NormalRandom nr;
};
```

———-

5.11.2 class SGEnumeratorProblem

— class SGEnumeratorProblem —

```
class SGEnumeratorProblem : public EnumeratorProblem< Subgroup >
{
public:
  SGEnumeratorProblem(SMFPGroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( ) ;
  void terminate( )
  void resume( );
private:
  SMFPGroup& theGroup;
  SGEnumeratorARCer arcer;
};
```

———-

5.11.3 class IsSubgroupInListARCer

— class IsSubgroupInListARCer —

```
class IsSubgroupInListARCer : public ARCer
{
public:
  IsSubgroupInListARCer( ComputationManager& boss )
  : ARCer( boss ), result(0)
  ~IsSubgroupInListARCer( )
  void setArguments( const SMList<Subgroup>&, const VectorOf<Word>&);
  bool isInList();
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SMListIterator<Subgroup>* theIterator;
  VectorOf<Word> theSubgroup;
  bool result;
};
```

———-

5.11.4 class IsSubgroupInList

— class IsSubgroupInList —

```
class IsSubgroupInList  : public Supervisor
{
public:
  IsSubgroupInList( SMList<Subgroup>&, const class SMSubgroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class SMSubgroup& theSubgroup;
  SMList<Subgroup>& theList;
  IsSubgroupInListARCer arcer;
};
```

———-

5.11.5 class SGListExtractOfIndexARCer

— class SGListExtractOfIndexARCer —

```
class SGListExtractOfIndexARCer : public ARCer
{
public:
  SGListExtractOfIndexARCer( ComputationManager& boss )
```

```
    : ARCer( boss ), minIndex( 1 ),  maxIndex( 0 )
  ~SGListExtractOfIndexARCer( )
  void setArguments( const SMList<Subgroup>&);
  const SMListData& getData() const
  void setMinimalIndex(int  min )
  void setMaximalIndex(int  max )
  int indexInFreeGroup( const VectorOf<Word>& s )const;
  int indexInFPGroup( const VectorOf<Word>& s )const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SMListIterator<Subgroup>* theIterator;
  SMListData theData;
  const SMFPGroup* theGroup;
  int minIndex;
  int maxIndex;
};
```

———-

5.11.6 class SGListExtractOfIndex

— class SGListExtractOfIndex —

```
class SGListExtractOfIndex  : public Supervisor
{
public:
  SGListExtractOfIndex( SMList<Subgroup>& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( )
private:
  SMList<Subgroup>& theList;
  bool started;
  SGListExtractOfIndexARCer arcer;
};
```

———-

5.12 Enumerators/include/WEnumerator.h

— WEnumerator.h —

```
#include "Supervisor.h"
```

```
#include "SMEnumerator.h"
#include "SMList.h"
#include "File.h"
#include "Word.h"
#include "SMFPGroup.h"
#include "OutMessages.h"
#include "RandomNumbers.h"
#include "ListProblems.h"
#include "SMListIterator.h"

\getchunk{class WordEnumeratorARCer}
\getchunk{class WordEnumeratorProblem}
\getchunk{class IsWordInListARCer}
\getchunk{class IsWordInList}
\getchunk{class WordsOfLengthARCer}
\getchunk{class WordsOfLength}
```

———-

5.12.1 class WordEnumeratorARCer

— class WordEnumeratorARCer —

```
class WordEnumeratorARCer : public EnumeratorARCer
{
public:
  WordEnumeratorARCer( ComputationManager& boss, SMListData& d, SMFPGroup& g )
    : EnumeratorARCer( boss, d ), group(g), numberOfAll(100),
    enumerateAll( true ), isNormalRandom( false ), numberOfsampleGenerators(1),
    maxLength(100), counter(0)
  void setEnumerateAll()
  void setEnumerateRandom()
  void setNormalRandom()
  void setUniformRandom()
  void setTotalNumber(int n)
  void setMaxLength(int n)
  void setSample(Word w)
  void writeResults( ostream& ostr )
  void readResults( istream& istr)
protected:
  void enumerate();
private:
  void enumerateAllWords( );
  Word getRandomWord() ;
  SMFPGroup& group;
  FreeGroup theGroup;
  bool enumerateAll;
  bool isNormalRandom;
```

```
  int numberOfsampleGenerators;
  int numberOfAll;
  int maxLength;
  int counter;
  Word theSample;
  UniformRandom ur;
  NormalRandom nr;
};
```

———-

5.12.2 class WordEnumeratorProblem

— class WordEnumeratorProblem —

```
class WordEnumeratorProblem : public EnumeratorProblem< Word >
{
public:
  WordEnumeratorProblem(SMFPGroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( ) ;
  void terminate( )
  void resume( );
private:
  SMFPGroup& theGroup;
  WordEnumeratorARCer arcer;
};
```

———-

5.12.3 class IsWordInListARCer

— class IsWordInListARCer —

```
class IsWordInListARCer : public ARCer
{
public:
  IsWordInListARCer( ComputationManager& boss ) : ARCer( boss ), result(0)
  ~IsWordInListARCer( )
  void setArguments( const SMList<Word>&, const Word&);
  bool isInList();
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SMListIterator<Word>* theIterator;
```

```
  Word theWord;
  bool result;
};
```

———-

5.12.4 class IsWordInList

— class IsWordInList —

```
class IsWordInList  : public Supervisor
{
public:
  IsWordInList( SMList<Word>&, const class SMWord& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class SMWord& theWord;
  SMList<Word>& theList;
  IsWordInListARCer arcer;
};
```

———-

5.12.5 class WordsOfLengthARCer

— class WordsOfLengthARCer —

```
class WordsOfLengthARCer : public ARCer
{
public:
  WordsOfLengthARCer( ComputationManager& boss )
    : ARCer( boss ),  minLength( 1 ), maxLength( 30000 )
  ~WordsOfLengthARCer( )
  void setArguments( const SMList<Word>&);
  const SMListData& getData() const
  void setMinimalLength(int  min )
  void setMaximalLength(int  max )
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SMListIterator<Word>* theIterator;
  int minLength;
  int maxLength;
```

```
  SMListData theData;
  FPGroup theGroup;
};
```

———-

5.12.6 class WordsOfLength

— class WordsOfLength —

```
class WordsOfLength  : public Supervisor
{
public:
  WordsOfLength( SMList<Word>& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( )
private:
  SMList<Word>& theList;
  bool started;
  WordsOfLengthARCer arcer;
};
```

———-

6 The Equations classes

6.1 Equations/include/DGESolver.h

— DGESolver.h —

```
#include "FreeGroup.h"

\getchunk{class DGESolver}
```

———-

6.1.1 class DGESolver

— class DGESolver —

```
class DGESolver
{
public:
  DGESolver( FreeGroup F, VectorOf<Chars> vNames, Word eq );
  ~DGESolver( );
  bool getSolution( const Word& u, Map& solution );
private:
  void defineRange( int curVar );
  bool pickNextValue( int curVar );
  bool findInverseOf( Word u, int p, int q, int& first, int& last );
  struct VarInfo
  FreeGroup theGroup;
  VectorOf<Chars> varNames;
  int numOfVars, numOfConsts;
  Word equation, w;
  int wLen, eqLen;
  VarInfo *varInfo;
  VarInfo *invVarInfo;
  VectorOf<int> buffer;
  VectorOf<int> invVarPos;
};
```

———-

6.2 Equations/include/NielsenTransformations.h

— **NielsenTransformations.h** —

```
#include "Set.h"
#include "List.h"
#include "Word.h"
#include "Automorphism.h"

\getchunk{class ElementaryRegularAuto}
\getchunk{class RegularAuto}
\getchunk{class ElementarySingularEndo}
\getchunk{class SingularEndo}

inline bool operator!=(const ElementaryRegularAuto& u,
                       const ElementaryRegularAuto& w)
inline Word operator | (Word& w, ElementaryRegularAuto& era)
inline Word operator | (const Word& w, const RegularAuto& ra)
inline RegularAuto operator | (const RegularAuto& ra1, const RegularAuto& ra2)
inline Word operator | (const Word& w, const ElementarySingularEndo& ese)
inline Word operator | (const Word& w, const SingularEndo& se)
inline ostream& operator <<(ostream& o, const ElementaryRegularAuto& era)
inline ostream& operator <<(ostream& o, const RegularAuto& ra)
inline ostream& operator <<(ostream& o, const ElementarySingularEndo& era)
```

```
inline ostream& operator <<(ostream& o, const SingularEndo& ra)
```

———-

6.2.1 class ElementaryRegularAuto

— class ElementaryRegularAuto —

```
class ElementaryRegularAuto
{
  Generator g, g1, g2;
public:
  ElementaryRegularAuto(const Generator& x, const Generator& y)
    : g(x), g1(x), g2(y)
  Generator x() const
  Generator y() const
  inline friend bool operator==(const ElementaryRegularAuto& u,
        const ElementaryRegularAuto& w)
  ElementaryRegularAuto inv() const
  Word imageOf(const Word& w) const
  Word operator()(const Word& w) const
};
```

———-

6.2.2 class RegularAuto

— class RegularAuto —

```
class RegularAuto
{
  ListOf<ElementaryRegularAuto> autoList;
public:
  RegularAuto()
  RegularAuto(const ElementaryRegularAuto& a) : autoList(a)
  RegularAuto(const ListOf<ElementaryRegularAuto>& list) : autoList(list)
  ListOf<ElementaryRegularAuto> getAutoList() const
  RegularAuto inv() const
  Word imageOf(const Word& w) const
  inline friend bool operator==(const RegularAuto& u, const RegularAuto& w)
  Automorphism makeAutomorphism(const FGGroup& G) const
};
```

———-

6.2.3 class ElementarySingularEndo

— class ElementarySingularEndo —

```
class ElementarySingularEndo
{
public:
  Generator g;
  ElementarySingularEndo(const Generator& G) : g(abs(ord(G)))
  Generator gen() const
  Word imageOf(const Word& w) const
  inline friend bool operator==(const ElementarySingularEndo& u,
        const ElementarySingularEndo& w)
};
```

———-

6.2.4 class SingularEndo

— class SingularEndo —

```
class SingularEndo
{
  SetOf<Generator> genSet;
public:
  SingularEndo()
  SingularEndo(const ElementarySingularEndo& ese) : genSet(ese.gen())
  SingularEndo(const SetOf<Generator>& set) : genSet(set)
  SetOf<Generator> generators() const
  bool isSpecializationOf(const SingularEndo& second) const
  friend SingularEndo operator | (const SingularEndo& se1,
          const SingularEndo& se2)
  Word imageOf(const Word& w) const
  Endomorphism makeEndomorphism(const FGGroup& G) const
  inline friend bool operator==(const SingularEndo& u, const SingularEndo& w)
};
```

———-

6.3 Equations/include/QEqnSolutions.h

— QEqnSolutions.h —

```
#include "Word.h"
#include "Vector.h"
#include "Queue.h"
```

```
#include "VectorPtr.h"
#include "QuickAssociations.h"
#include "NielsenTransformations.h"

\getchunk{class QEqnSolutionsInFreeGroup>
\getchunk{template <class T> VectorPtrOf<T> makeVectorPtrOf}

Chars  checkEquation(const FreeGroup& G, const Word& equation, int numOfVar);
```

———-

6.3.1 class QEqnSolutionsInFreeGroup

— class QEqnSolutionsInFreeGroup —

```
class QEqnSolutionsInFreeGroup
{
public:
  QEqnSolutionsInFreeGroup(const FreeGroup& G, const Word& equation,
                           int numOfVar);
  void startComputation();
  void continueComputation();
  void disableBuildingRegStab()
  void enableBuildingRegStab()
  void disableBuildingSolutions()
  void enableBuildingSolutions()
  bool isComputationDone() const
  bool isComputationStarted() const
  bool haveNewSolution() const
  bool nextSolution()
  bool haveNewStabGenerator() const
  bool nextStabGenerator()
  bool isBuildingSolutionsEnabled() const
  bool isBuildingRegStabEnabled() const
  Endomorphism getSolution()
  Automorphism getStabGenerator()
  Word surfaceForm()
  Automorphism toSurfaceForm()
  FreeGroup group() const
  Word equation() const
  int numberOfVariables() const
  int numberOfGenerators() const
  int numberOfConstants() const
  int rank() const
  VectorPtrOf<Endomorphism> basicSolutions() const
  VectorPtrOf<Automorphism> regStabGenerators() const
  Automorphism prefixMap() const
  struct EquationStatus;
```

```
  EquationStatus getProcessStatus() const
  Endomorphism eliminator() const
  friend ostream& operator << ( ostream&, const QEqnSolutionsInFreeGroup& );

\getchunk{struct EquationStatus}

private:
  Bool isVariable(const Generator& g) const
  bool isConstant(const Generator& g) const
  void buildNewVerticesAndLoopsFrom(const Word& vertex);
  RegularAuto buildPath(const Word& from, const Word& to);
  RegularAuto buildPathTo(const Word& to);
  void registerEdge(const Word& source, const ElementaryRegularAuto& edge,
        const Word& target);
  void buildSolutionsOfOneVertex(const Word& w);
  VectorOf<int> checkProperties(const Word& w);
  ListOf<SingularEndo> buildSingulars(const Word& w,
                                      const VectorOf<int>& varProperties);
  void appendNewSolution(ListOf<SingularEndo>& solutions,
       const SingularEndo& newSolution);
  int rankOfBasicSolution(const Endomorphism& solution) const;
  void processOneLoop(const Edge& edge);
  void computeSurfaceForm();
  Word extractSquare(Word& w, int firstLocation, int secondLocation);
  Word extractPinch(Word& w, int firstLocation,  int secondLocation);
  Word extractCommutator(Word& w, int xFirstLocation,  int xSecondLocation,
       int yFirstLocation);
  int getLocationOfArbitraryVariable(const Word& w, int start, int stop) const;
  int getPairOfVariable(Word& w, int& firstLocation, int& secondLocation,
      bool oppositeOrder) const;
  Word bringToCanonicForm(Word& squares, Word& commutators, Word constants,
        Word& pinches);
  Word theEquation;
  int theNumberOfGenerators;
  int theNumberOfVariables;
  FreeGroup theGroup;
  VectorPtrOf<Automorphism> theRegStabGenerators;
  VectorPtrOf<Endomorphism> theBasicSolutions;
  int theRank;
  Word theSurfaceForm;
  Automorphism toSurface;
  bool computationIsStarted;
  bool computationIsDone;
  int solution;
  int generator;
  Endomorphism theEliminator;

\getchunk{struct Edge}

  Automorphism startPath;
```

```
  Automorphism invStartPath;
  Word root;
  QuickAssociationsOf<Word, ElementaryRegularAuto> maxSubTree;
  QueueOf<Word> newVertices;
  QueueOf<Edge> loops;
  Endomorphism removeVarEndo;
  SetOf<Endomorphism> setOfBasicSolutions;
  enum { MUST_BE_MAPPED_TO_ITSELF,
   MUST_BE_MAPPED_TO_1,
   CAN_BE_MAPPED_TO_1
  };
  int tempRank;
  SetOf<Automorphism> regStabSet;
  bool surfaceFormIsComputed;
  EquationStatus status;
};
```

———-

6.3.2 struct EquationStatus

— **struct EquationStatus** —

```
struct EquationStatus
{
  EquationStatus() : enableBuildingSolutions(true),
    enableBuildingRegStab(true), verticesToPass(0), verticesPassed(0),
    loopsToPass(0), loopsPassed(0), totalLengthOfLoops(0)
    bool enableBuildingSolutions;
    bool enableBuildingRegStab;
    int verticesToPass;
    int verticesPassed;
    int loopsToPass;
    int loopsPassed;
    long totalLengthOfLoops;
};

\subsubsection{struct Edge}
\begin{chunk}{struct Edge}
struct Edge
{
  Word vertex;
  ElementaryRegularAuto edge;
  Edge() : edge(1,2)
  Edge(const Word& w, const ElementaryRegularAuto& e) : vertex(w), edge(e)
  friend inline bool operator ==(const Edge& x, const Edge& y)
  friend ostream& operator <<( ostream& o, const Edge& E);
  int hash() const
```

```
};
```

———-

6.3.3 template ¡class T¿ VectorPtrOf¡T¿ makeVectorPtrOf

— template ¡class T¿ VectorPtrOf¡T¿ makeVectorPtrOf —

```
template <class T> VectorPtrOf<T> makeVectorPtrOf(const SetOf<T>& S)
```

———-

6.4 Equations/include/Queue.h

— Queue.h —

```
#include "ObjectOf.h"
#include "List.h"

\getchunk{template < class T > class QueueOf}
```

———-

6.4.1 template ¡ class T ¿ class QueueOf

— template ¡ class T ¿ class QueueOf —

```
template < class T > class QueueOf : public ListOf<T>
{
public:
  QueueOf( ) : ListOf<T>()
  QueueOf( const T& t ) : ListOf<T>(t)
  void push( const T& t );
  T pop( )
  void popAll( )
  Bool isEmpty( ) const
  Bool isntEmpty( ) const
};
```

———-

6.5 Equations/include/SolutionsEnum.h

— **SolutionsEnum.h** —

```
#include "QEqnSolutions.h"
#include "TupleEnumerator.h"
#include "RandomNumbers.h"

\getchunk{class QEqnSolutionsEnumerator}
const int randomSolutionsThreshold = 500;
\getchunk{class GeneratorOfRandomSolutions}
```

———-

6.5.1 class QEqnSolutionsEnumerator

— **class QEqnSolutionsEnumerator** —

```
class QEqnSolutionsEnumerator
{
public:
  QEqnSolutionsEnumerator(const QEqnSolutionsInFreeGroup& equation);
  Endomorphism getSolution() const
  bool nextSolution();
  bool done() const
  void reset();
private:
  bool areBasicSolutionsParametrized();
  Word theEquation;
  int theNumberOfGenerators;
  int theNumberOfVariables;
  FreeGroup theGroup;
  VectorPtrOf<Automorphism> theRegStabGenerators;
  VectorPtrOf<Automorphism> theRegStabInvGenerators;
  VectorPtrOf<Endomorphism> theBasicSolutions;
  Automorphism prefixAuto;
  bool allSolutionsAreEnumerated;
  Trichotomy numberOfSolutionsIsFinite;
  Endomorphism solution;
  Automorphism stabAuto;
  int currentBasicSolution;
  Endomorphism specEndo;
  EnumeratorOfWordTuples tuplesEnumerator;
  VectorOf<bool> invGensComputed;
};
```

———-

6.5.2 class GeneratorOfRandomSolutions

— class GeneratorOfRandomSolutions —

```
class GeneratorOfRandomSolutions
{
public:
  GeneratorOfRandomSolutions( const FreeGroup& group, const Word& equation,
            int numberOfVariables );
  void setBasicSolutions( const VectorPtrOf<Endomorphism>& newBasicSolutions )
  void setRegStabGenerators( const VectorPtrOf<Automorphism>& newGenerators )
  void setThreshold( int newThreshold )
  bool hasSolutions() const;
  bool generateSolution();
  Endomorphism getSolution() const
  int getThreshold( ) const
  int getCurrentThreshold( ) const
  const VectorPtrOf<Endomorphism>& getBasicSolutions( ) const
  const VectorPtrOf<Automorphism>& getRegStabGenerators(  ) const
protected:
  void generateSomeSolution();
private:
  FreeGroup theGroup;
  int numOfVar;
  int numOfGen;
  VectorPtrOf<Endomorphism> basicSolutions;
  VectorPtrOf<Automorphism> regStabGenerators;
  VectorPtrOf<Automorphism> regStabGeneratorsInv;
  SetOf<Endomorphism> solutionSet;
  Endomorphism solution;
  int threshold;
  UniformRandom rnd;
  Endomorphism variablesEliminator;
};
```

———-

6.6 Equations/include/TupleEnumerator.h

— TupleEnumerator.h —

```
#include "Vector.h"
#include "Word.h"

\getchunk{class EnumeratorOfWordTuples}
\getchunk{class CantorEnumeration}
```

———-

6.6.1 class EnumeratorOfWordTuples

— class EnumeratorOfWordTuples —

```
class EnumeratorOfWordTuples
{
public:
  EnumeratorOfWordTuples(VectorOf<int> numOfGens) :
    tupleSize(numOfGens.length()), tuple(numOfGens.length()),
    maxWord(numOfGens.length()), numberOfGens(numOfGens),
    fixedWord(0)
  EnumeratorOfWordTuples(int size, int numOfGens) :
    tupleSize(size), tuple(size), maxWord(size),
    numberOfGens(size), fixedWord(0)
  void reset()
  void next();
  VectorOf<Word> value() const
private:
  int tupleSize;
  VectorOf<Word> tuple;
  VectorOf<Word> maxWord;
  VectorOf<int>  numberOfGens;
  int fixedWord;
};
```

———-

6.6.2 class CantorEnumeration

— class CantorEnumeration —

```
class CantorEnumeration
{
public:
  static Integer encodePair( const Integer& x, const Integer& y);
  static void decodePair( const Integer& n, Integer& x, Integer& y );
  static Integer encodeTuple( const VectorOf<Integer>& tuple );
  static VectorOf<Integer> decodeTuple( const Integer& number );
};
```

———-

6.7 Equations/include/VectorPtr.h

— VectorPtr.h —

```
#include "RefCounter.h"
#include "ObjectOf.h"

\getchunk{template < class T > class VectorItemRef}
\getchunk{template <class T> struct VectorPtrRep}
\getchunk{template <class T> class VectorPtrOf}
\getchunk{template < class T > class VectorItemRef}
```

———-

6.7.1 template ¡ class T ¿ class VectorItemRef

— template ¡ class T ¿ class VectorItemRef —

```
template < class T > class VectorItemRef;
```

———-

6.7.2 template ¡class T¿ struct VectorPtrRep

— template ¡class T¿ struct VectorPtrRep —

```
template <class T> struct VectorPtrRep : public RefCounter
{
public :
  VectorPtrRep( const VectorPtrRep& vr )
  VectorPtrRep( int l )
  VectorPtrRep( int l, bool e )
  ~VectorPtrRep( )
  VectorPtrRep* clone( )
  int length() const
  bool isValid(int i) const
  void set(int i, const T& t)
  T& ref(int i)
  T val(int i) const
  VectorItemRef<T> operator [] ( int i )
  void append( const T& t )
  void prepend( const T& t )
  void shrink( int start, int newlen )
private :
  VectorPtrRep& operator = ( const VectorPtrRep& );
  bool fastExpansion;
  unsigned int first;
  unsigned int last;
  unsigned int len;
  T** vec;
};
```

———-

6.7.3 template ¡class T¿ class VectorPtrOf

— template ¡class T¿ class VectorPtrOf —

```
template <class T> class VectorPtrOf : public ObjectOf< VectorPtrRep<T> >
{
  typedef VectorPtrRep< T > Rep;
  typedef ObjectOf< Rep > Base;
public:
  VectorPtrOf( int len = 0 ) : Base( new Rep(len) )
  VectorPtrOf( int len, bool e ) : Base( new Rep(len,e) )
  VectorPtrOf( int len, const VectorPtrOf& v ) : Base( new Rep(len) )
  VectorPtrOf( int len, bool e, const VectorPtrOf& v ) : Base( new Rep(len,e))
  bool operator == ( const VectorPtrOf& v ) const
  bool operator != ( const VectorPtrOf& v ) const
  T operator [] ( int i ) const
  VectorItemRef<T> operator [] ( int i )
  T val( int i ) const
  T& ref( int i )
  bool isValid( int i ) const
  int length( ) const
  int hash() const
  int indexOf( const T& t ) const
  void append( const T& t )
  void prepend( const T& t )
  void shrink( int newLength )
  void shrink( int start, int newLength )
  inline friend ostream& operator << ( ostream& o, const VectorPtrOf& v )
private:
};
```

———-

6.7.4 template ¡ class T ¿ class VectorItemRef

— template ¡ class T ¿ class VectorItemRef —

```
template < class T > class VectorItemRef
{
public:
  T& operator = ( const T& t )
  bool operator == ( const T& t )
  operator T( )
private:
  friend class VectorPtrRep<T>;
```

```
  VectorItemRef( T* p) : elPoint(p)
  VectorItemRef( T** addr, T* p) : elPoint(0), elAddr(addr)
  T* elPoint;
  T** elAddr;
private:
  VectorItemRef& operator = ( const VectorItemRef& ref );
public:
  VectorItemRef( const VectorItemRef& ref )
};
```

———-

6.8 Equations/include/VertexInfo.h

— **VertexInfo.h** —

```
#include "Word.h"
#include "NielsenTransformations.h"

\getchunk{struct VertexInfo}
```

———-

6.8.1 struct VertexInfo

— **struct VertexInfo** —

```
struct VertexInfo
{
  VertexInfo(const Word& w, int i, const SingularEndo& se) :
    word(w), lastVar(i), eliminator(se)
  friend inline bool operator==(const VertexInfo& x, const VertexInfo& y)
  friend inline ostream& operator<<(ostream& o, const VertexInfo& x)
  Word word;
  int lastVar;
  SingularEndo eliminator;
};
```

———-

7 The FSA classes

7.1 FSA/include/DFSA.h

— **DFSA.h** —

```
#include "FSA.h"
#include "DFSARep.h"

\getchunk{class DFSA}
\getchunk{class GroupDFSA}
```

———-

7.1.1 class DFSA

— class DFSA —

```
class DFSA : public FSA {
typedef DFSARep::State State;
public:
  DFSA( ) : FSA( new DFSARep() )

  DFSA(const VectorOf<Chars> & genNames) : FSA( new DFSARep("",genNames) )

  DFSA(Chars name,const VectorOf<Chars> & genNames)
     : FSA( new DFSARep(name,genNames) )

  DFSA( const VectorOf<Chars> & genNames, int numOfStates)
     : FSA( new DFSARep("",genNames, numOfStates,1) )

  DFSA( const Chars & Name, const VectorOf<Chars> & genNames, int numOfStates)
     : FSA( new DFSARep(Name, genNames, numOfStates,1) )

  DFSA( const VectorOf<Chars> & genNames, int numOfStates, int numOfStrings)
     : FSA( new DFSARep("", genNames, numOfStates, numOfStrings ) )

  DFSA( const Chars & Name, const VectorOf<Chars> & genNames, int numOfStates, int numOfStrings)
     : FSA( new DFSARep(Name, genNames, numOfStates, numOfStrings ) )

  State target(State s,Generator g) const
  int getNumStates() const

protected:
  DFSA( DFSARep * rep ) : FSA((FSARep *)rep)
  const DFSARep *look() const
  DFSARep *change()
};
```

———-

7.1.2 class GroupDFSA

— **class GroupDFSA** —

```
class GroupDFSA : public FSA {
friend class DiffMachineRep;
typedef DFSARep::State State;
public:
  GroupDFSA( ) : FSA( new GroupDFSARep() )
  GroupDFSA(const VectorOf<Chars> & genNames)
     : FSA( new GroupDFSARep("",genNames) )
  GroupDFSA(const VectorOf<Chars> & genNames, const WordOrder & word_order)
     : FSA( new GroupDFSARep("",genNames, word_order) )
  GroupDFSA(Chars name,const VectorOf<Chars> & genNames)
     : FSA( new GroupDFSARep(name,genNames) )
  GroupDFSA(Chars name,const VectorOf<Chars> & genNames,
            const WordOrder & word_order)
     : FSA( new GroupDFSARep(name,genNames,word_order) )
  GroupDFSA(const Chars & Name, const VectorOf<Chars> & genNames,
            int numOfStates)
     : FSA( new GroupDFSARep(Name, genNames, numOfStates,1) )
  GroupDFSA(const VectorOf<Chars> & genNames, const WordOrder & word_order,
            int numOfStates)
     : FSA( new GroupDFSARep("",genNames, word_order,numOfStates,1) )
  GroupDFSA(const VectorOf<Chars> & genNames, int numOfStates,
            int numOfStrings)
     : FSA( new GroupDFSARep("", genNames, numOfStates, numOfStrings ) )
  GroupDFSA(const VectorOf<Chars> & genNames, const WordOrder & word_order,
            int numOfStates, int numOfStrings)
     : FSA( new GroupDFSARep("", genNames, word_order,
                             numOfStates, numOfStrings ) )
  GroupDFSA(const Chars & Name, const VectorOf<Chars> & genNames,
            int numOfStates, int numOfStrings)
     : FSA( new GroupDFSARep(Name, genNames, numOfStates, numOfStrings ) )
  State target(State s,Generator g) const
  int getNumStates() const
protected:
  GroupDFSA( GroupDFSARep * rep ) : FSA((FSARep *)rep)
  const GroupDFSARep *look() const
  GroupDFSARep *change()
};
```

———-

7.2 FSA/include/DFSAParser.h

— **DFSAParser.h** —

```
#include "global.h"
#include "DFSARep.h"
```

```
enum { SIMPLE=0, WORDS=1, LABELED=2};

\getchunk{class DFSAParser}
```

———-

7.2.1 class DFSAParser

— class DFSAParser —

```
class DFSAParser {
public:
  DFSAParser(istream  &istr,int stype,Bool useInv) : name(""), str(istr),
   parses(1), table(0), alphabet(0), baseAlphabet(0), stateAlphabet(0),
   symbolOrder(1), symbolForColumn(1), stateWordLabels(0), stateCharsLabels(0),
   numStates(0), numSymbols(0), numTransits(0), numStrings(0), minimized(NO),
   stateType(stype), useInverses(useInv)
  ~DFSAParser()
  void parseError(char * errormessage);
  void setDFSA(DFSARep * M);
  void setDFSA(GroupDFSARep * M);
  ListOf<int> getStateLabelsIndex() const
  VectorOf<Chars> getStateCharsLabels() const
  VectorOf<Word> getStateWordLabels() const
  void setAlphabet(DFSARep * M);
  void setAlphabet(GroupDFSARep * M);
  void setFlags(DFSARep * M);
  void setData(DFSARep * M);
  Bool parseDFSA(DFSARep * M);
  void parseAlphabetRec();
  void matchAlphabet(DFSARep * M);
  void matchAlphabet(GroupDFSARep * M);
  void parseStatesRec();
  void parseFlags();
  void parseAccepting();
  void parseTableRec();
  void setStateAlphabet(DFSARep * M);
  void parseStateLabelsRec();
  void parseStateWordLabels();
private:
  typedef DFSARep::State State;
  typedef DFSARep::GeneratorIndex GeneratorIndex;
  int ** table;
  Chars name;
  VectorOf<Chars> alphabet;
  VectorOf<Chars> baseAlphabet;
  VectorOf<Chars> stateAlphabet;
```

```
  VectorOf<int> symbolOrder;
  VectorOf<int> symbolForColumn;
  Chars paddingSymbol;
  int stateType;
  VectorOf<Word> stateWordLabels;
  VectorOf<Chars> stateCharsLabels;
  ListOf<int> stateLabelsIndex;
  int numStates;
  int numSymbols;
  int numTransits;
  int numStrings;
  Bool minimized;
  Bool useInverses;
  istream& str;
  Bool getInt(int &);
  Bool readToToken();
  Bool getToken(int &);
  Bool getToken(Chars &);
  Chars getToken();
  Bool checkToken(const char * token);
  Bool checkToken(const char ch);
  void parseTable(Bool compressed);
  Bool parses;
};
```

———-

7.3 FSA/include/DFSARep.h

— DFSARep.h —

```
#include <Integer.h>
#include "global.h"
#include "Vector.h"
#include "Generator.h"
#include "Word.h"
#include "Vector.h"
#include "Chars.h"
#include "FSARep.h"
#include "WordOrder.h"

\getchunk{class DFSARep}
\getchunk{class GroupDFSARep}
```

———-

7.3.1 class DFSARep

— class DFSARep —

```
class DFSARep : public FSARep {
public:
typedef int State;
typedef int Category;
typedef int GeneratorIndex;
  DFSARep() : name(""), paddingSymbol("_"), generatorNames(0), numSymbols(0),
              numTransits(0), numStates(0), numStrings(1), transitTable(1,YES),
              categoryList(1,YES), minimized(NO)
  DFSARep (const Chars & Name,const VectorOf<Chars> & genNames) :
    name(Name), paddingSymbol("_"), generatorNames(genNames),
    numSymbols(genNames.length()), numStates(0), numStrings(1),
    numTransits(genNames.length()), transitTable(1,YES),
    categoryList(1,YES), minimized(NO)
  DFSARep (const Chars & Name, const VectorOf<Chars> & genNames,
                               int numOfStates, int numOfStrings) :
    name(name), paddingSymbol("_"), generatorNames(genNames),
    numSymbols(genNames.length()), numStates(numOfStates),
    numStrings(numOfStrings),
    numTransits(numOfStrings==1 ?
                genNames.length() :
                genNames.length()*(genNames.length()+2)),
    transitTable(numOfStates+1,YES), categoryList(numOfStates+1,YES),
    minimized(NO)
  DFSARep( const DFSARep& D ) :  name(D.name), paddingSymbol(D.paddingSymbol),
    generatorNames(D.generatorNames), numStates(D.numStates),
    numStrings(D.numStrings), numSymbols(D.numSymbols),
    numTransits(D.numTransits), categoryList(D.categoryList),
    minimized(D.minimized)
  ~DFSARep()
  DFSARep & operator = ( const DFSARep & D )
  inline friend ostream& operator <<
        ( ostream& ostr, const DFSARep& D )
  FSARep *clone() const
  Bool equalDFSA(const FSARep & D) const
  Bool operator == ( const FSARep& D ) const
  void minimize()
  Integer sizeLanguage() const ;
  Bool finiteLanguage() const
  virtual void printAccepting(ostream &str=cout) const ;
  void printFlags(ostream &str=cout) const ;
  virtual void printStates(ostream &str=cout) const ;
  void printTableDensely(ostream &str=cout) const ;
  void printTableSparsely(ostream &str=cout) const ;
  Bool accepts(Word w) const
  Bool rejectsInState(Word w, int& state) const
```

```
Bool nextAcceptedWord(Word& w) const
void readFrom(istream &str = cin);
void printOn(ostream &str = cout) const;
void GAPprintWord(ostream &str,const Word & w) const;
void printAlphabet(ostream &str=cout) const ;
void printWord(ostream &str,const Word & w) const ;
Chars getName() const
State startState() const
State failure() const
Category category(State s) const
Bool getMinimized() const
State getNumStates() const
int getNumStrings() const
int getNumSymbols() const
int getNumTransits() const
Chars getPaddingSymbol() const
State targetInt( State s,int i) const
State targetInt( State s,int i, int j) const
State targetIntOn1stString(State s,int i) const
State targetIntOn2ndString(State s,int j) const
VectorOf<Chars> getGeneratorNames() const
int getNumGenerators() const
virtual Chars getSymbolName(Generator g) const
virtual Chars getSymbolName(int i) const
virtual Generator getSymbol(int i) const
virtual int getPosition(Generator a) const
virtual State target( State s,Generator a) const
virtual State target( State s,Word w) const
virtual State target( State s,Generator a, Generator b) const
virtual State targetOn1stString(State s,Generator a) const
virtual State targetOn2ndString(State s,Generator b) const
bool noTransits(State s) const
void setName(const Chars & Name)
void setNumStrings(int numOfStrings)
virtual void setNumStates(int numOfStates);
State newState();
void setTargetInt( State source,int i, State target )
void setTargetInt( State source,int i, int j, State target )
void setTargetIntOn1stString ( State source,int i, State target )
void setTargetIntOn2ndString ( State source, int j, State target )
void setCategory(State s, Category cat)
void setPaddingSymbol(const Chars & s)
void setMinimized(Bool m)
virtual void setGeneratorNames(const VectorOf<Chars> & names)
virtual void setTarget( State source,Generator a, State target )
virtual void setTarget(State source,Generator a, Generator b, State target )
virtual void setTargetOn1stString( State source,Generator a,State target )
virtual void setTargetOn2ndString(State source, Generator b, State target )
void resetTransitTable()
void clearTransits(State s);
```

```
  void removeDeadState(State s);
  void removeDeadStates(const VectorOf<State> & deadStates);
  void write( ostream& ostr ) const;
  void read( istream& istr );
protected:
  VectorOf<Chars> generatorNames;
  int numSymbols;
  int numTransits;
  int numStrings;
private:
  Chars name;
  Chars paddingSymbol;
  VectorOf<State *> transitTable;
  VectorOf<Category> categoryList;
  int  numStates;
  Bool minimized;
  State * newRow() const
  VectorOf<State *> copyTransitTable() const;
  void clearTransitTable();
};
```

———-

7.3.2 class GroupDFSARep

— class GroupDFSARep —

```
class GroupDFSARep  : public DFSARep {
public:
  GroupDFSARep() : DFSARep(),
    symbolForColumn(1), columnForSymbol(1), columnForInvSymbol(1)
  GroupDFSARep (const Chars & Name, const VectorOf<Chars>& genNames) :
    DFSARep(Name,0),
    symbolForColumn(1), columnForSymbol(1), columnForInvSymbol(1)
  GroupDFSARep (const Chars & Name, const VectorOf<Chars>& genNames,
                const WordOrder & word_order) :
    DFSARep(Name,0),
    symbolForColumn(1), columnForSymbol(1), columnForInvSymbol(1)
  GroupDFSARep (const Chars & Name, const VectorOf<Chars>& genNames,
                              int numOfStates, int numOfStrings) :
    DFSARep(Name,0,numOfStates,numOfStrings),
    symbolForColumn(1), columnForSymbol(1), columnForInvSymbol(1)
  GroupDFSARep (const Chars & Name, const VectorOf<Chars>& genNames,
     const WordOrder & word_order, int numOfStates, int numOfStrings) :
    DFSARep(Name,0,numOfStates,numOfStrings),
    symbolForColumn(1), columnForSymbol(1), columnForInvSymbol(1)
  GroupDFSARep & operator = (const GroupDFSARep & D)
  FSARep *clone() const
```

```
  Bool operator == ( const GroupDFSARep& D ) const
  State target( State s,Generator a) const
  State target( State s,Word w) const
  State target( State s,Generator a, Generator b) const
  State targetOn1stString(State s,Generator a) const
  State targetOn2ndString(State s,Generator b) const
  Chars getSymbolName(Generator g) const
  Chars getSymbolName(int i) const
  Generator getSymbol(int i) const
  int getPosition(Generator a) const
  Bool selfInverse(Generator a) const
  void setTarget( State source,Generator a, State target )
  void setTarget ( State source,Generator a, Generator b, State target )
  void setTargetOn1stString( State source,Generator a,State target )
  void setTargetOn2ndString ( State source, Generator b, State target )
  void setGeneratorNames(const VectorOf<Chars> & names)
  void setGeneratorNames(const VectorOf<Chars> & names,
                         const VectorOf<int> & symForCol)
  Bool accepts(Word w) const
  Bool rejectsInState(Word w, int& state) const
  Bool nextAcceptedWord(Word& w) const
  void readFrom(istream &str = cin);
  void write( ostream& ostr ) const
  void read( istream& istr )
private:
  VectorOf<int> symbolForColumn;
  VectorOf<int> columnForSymbol;
  VectorOf<int> columnForInvSymbol;
};
```

———-

7.4 FSA/include/FSA.h

— FSA.h —

```
#include <Integer.h>
#include "global.h"
#include "ObjectOf.h"
#include "FSARep.h"

\getchunk{class FSA}
```

———-

7.4.1 class FSA

— **class FSA** —

```
class FSA : public ObjectOf<FSARep> {
public:
  Bool accepts(Word w) const
  Bool rejectsInState(Word w, int& state) const
  Bool nextAcceptedWord(Word& w) const
  void minimize()
  Integer sizeLanguage() const
  Bool finiteLanguage() const
  void readFrom(istream &str = cin)
  void printOn(ostream &str = cout) const
  void setName(const Chars & name)
  int getNumStates() const
  int operator == ( const FSA & F ) const
  int operator != ( const FSA & F ) const
  friend ostream& operator < ( ostream& ostr, const FSA& fsa )
  friend istream& operator > ( istream& istr, FSA& fsa )
protected:
  typedef ObjectOf<FSARep> R;
  FSA( FSARep *p ) : R(p)
};
```

———-

7.5 FSA/include/FSARep.h

— **FSARep.h** —

```
#include <Integer.h>
#include "global.h"
#include "RefCounter.h"
#include "Word.h"

\getchunk{class FSARep}
```

———-

7.5.1 class FSARep

— **class FSARep** —

```
class FSARep : public RefCounter {
public:
  virtual ~FSARep()
  virtual FSARep *clone() const = 0;
  virtual Bool operator == ( const FSARep& ) const = 0;
```

```
  virtual Bool accepts(Word w) const = 0;
  virtual Bool rejectsInState(Word w, int& state) const = 0;
  virtual Bool nextAcceptedWord(Word& w) const = 0;
  virtual void minimize() = 0;
  virtual Integer sizeLanguage() const = 0;
  virtual Bool finiteLanguage() const = 0;
  virtual void readFrom(istream &str = cin) = 0;
  virtual void printOn(ostream &str = cout) const = 0;
  virtual void setName(const Chars & name)=0;
  virtual int getNumStates() const =0;
  virtual void write( ostream& ostr ) const = 0;
  virtual void read( istream& istr ) = 0;

};
```

———-

7.6 FSA/include/StatePair.h

— **StatePair.h** —

```
typedef int State ;

\getchunk{class StatePair}
```

———-

7.6.1 class StatePair

— **class StatePair** —

```
class StatePair {
public:
  StatePair(): first(0),second(0)
  StatePair( const StatePair& s )
  ~StatePair()
  State getFirstElt() const
  State getSecondElt() const
  void setFirstElt(State f)
  void setSecondElt(State s)
  int hash() const
  int operator== ( const StatePair& s ) const
  StatePair&  operator= ( const StatePair& s )
private:
  State first;
  State second;
```

```
};
```

———-

8 The GAP classes

8.1 GAP/include/GAP.h

— **GAP.h** —

```
#include "global.h"

\getchunk{class GAP}
```

———-

8.1.1 class GAP

— **class GAP** —

```
class GAP {
public:
  GAP( )
  bool available( ) const;
private:
};
```

———-

8.2 GAP/include/GAPManager.h

— **GAPManager.h** —

```
#include "GAP.h"

\getchunk{class GAPManager}
```

———-

8.2.1 class GAPManager

— class GAPManager —

```
class GAPManager {
public:
private:
  const int maxNumOfGaps;
  int theNumOfGaps;
  GAP* gap;
};
```

———-

8.3 GAP/include/Permutation.h

— Permutation.h —

```
#include "IStreamPoll.h"
#include "Vector.h"

\getchunk{class Permutation}
```

———-

8.3.1 class Permutation

— class Permutation —

```
class Permutation {
public:
  struct GAPPermutation;
  friend struct GAPPermutation;
  Permutation::Permutation( VectorOf<int> v = VectorOf<int>() );
  Permutation inverse( ) const;
  friend ostream& operator << ( ostream&, const Permutation& );
  friend IStreamPoll operator >> ( istream&, Permutation& );
  GAPPermutation inGAP( )
private:
  struct GAPPermutation
  friend ostream& operator << ( ostream& o, const GAPPermutation& gp );
  friend IStreamPoll operator >> ( istream& i, const GAPPermutation& gp );
  void printInGAP( ostream& ) const;
  void readInGAP( istream&, Chars& );
  VectorOf<int> perm;
};
```

———-

8.4 GAP/include/PermutationParser.h

— PermutationParser.h —

```
#include "Permutation.h"

\getchunk{class PermutationParser}
```

———-

8.4.1 class PermutationParser

— class PermutationParser —

```
class PermutationParser {
public:
  PermutationParser(istream &str) : istr(str)
  Permutation parsePermutation( Chars& errMesg );
  int cursorPos( ) const
private:
  istream& istr;
  int pos;
  bool isDigit( char c );
  bool isSpace( char c );
  void eatWhite( );
};
```

———-

9 The general classes

9.1 general/include/Associations.h

— Associations.h —

```
#include "RefCounter.h"
#include "ObjectOf.h"
#include "List.h"
#include "Cell.h"

\getchunk{template Association}
\getchunk{template AssociationsIteratorRep}
\getchunk{template AssociationsRep}
```

```
\getchunk{template AssocRef}
\getchunk{template AssociationsOf}
\getchunk{template AssocRef}
\getchunk{AssociationsOf<Key,Val>::operator [ ]}
\getchunk{template AssociationsIteratorRep}
\getchunk{template AssociationsIterator}
\getchunk{AssociationsOf<Key,Val>::keys( )}
```

———-

9.1.1 template Association

— **template Association** —

```
template <class Key, class Val> class Association {
public:
  Association(Key k, Val v) : key(k), val(v)
  Key key;
  Val val;
  friend ostream& operator < ( ostream& ostr, const Association& A )
  friend istream& operator > ( istream& istr, Association& A )
};
```

———-

9.1.2 template AssociationsIteratorRep

— **template AssociationsIteratorRep** —

```
template <class Key, class Val> class AssociationsIteratorRep;
```

———-

9.1.3 template AssociationsRep

— **template AssociationsRep** —

```
template <class Key, class Val> class AssociationsRep : public RefCounter {
public:
  AssociationsRep( ) : theList(NULL), theCardinality(0)
  AssociationsRep( const AssociationsRep& ar )
  ~AssociationsRep( )
  AssociationsRep* clone( ) const
  void unbind( const Key& k )
  void bind( const Key& k, const Val& v )
```

```
  Val val( const Key& k ) const
  Bool bound( const Key& k ) const
  int cardinality() const
  void  write( ostream& ostr ) const;
  void  read( istream& istr );
private:
  typedef Cell< Association<Key,Val> > CellType;
  friend class AssociationsIteratorRep<Key,Val>;
  AssociationsRep(CellType* L)
  CellType* seek( const Key& k ) const
  CellType* theList;
  int theCardinality;
};
```

———-

9.1.4 template AssocRef

— template AssocRef —

```
template <class Key, class Val> struct AssocRef;
```

———-

9.1.5 template AssociationsOf

— template AssociationsOf —

```
template <class Key, class Val> class AssociationsOf :
  public ObjectOf< AssociationsRep<Key,Val> > {
public:
  AssociationsOf( ) : ObjectOf<Rep>(new Rep())
  Val operator [ ] ( const Key& k ) const
  AssocRef<Key,Val> operator [ ] ( const Key& k );
  Val valueOf( const Key& k ) const
  void bind( const Key& k, const Val& v )
  void unbind( const Key& k )
  Bool bound( const Key& k ) const
  ListOf<Key> keys( ) const;
  int cardinality() const
  friend ostream& operator < ( ostream& ostr, const AssociationsOf& A )
  friend istream& operator > ( istream& istr, AssociationsOf& A)
private:
  typedef AssociationsRep<Key,Val> Rep;
  friend class AssociationsIteratorRep<Key,Val>;
};
```

———-

9.1.6 template AssocRef

— template AssocRef —

```
template <class Key, class Val> struct AssocRef {
  AssociationsOf<Key,Val>& asref;
  const Key key;
  AssocRef( AssociationsOf<Key,Val>& a, const Key& k ) : asref(a), key(k)
   const Val& operator = ( const Val& val )
  operator Val ( )
  operator void* ( )
};
```

———-

9.1.7 template AssociationsOf¡Key,Val¿::operator [ ]

— AssociationsOf¡Key,Val¿::operator [ ] —

```
template <class Key, class Val>
inline AssocRef<Key,Val> AssociationsOf<Key,Val>::operator [ ] ( const Key& k )
```

———-

9.1.8 template AssociationsIteratorRep

— template AssociationsIteratorRep —

```
template <class Key, class Val> class AssociationsIteratorRep :
  public RefCounter {
public:
  AssociationsIteratorRep( const AssociationsOf<Key,Val>& A ) :
    theAssociations(A)
  AssociationsIteratorRep *clone() const
  Bool operator == ( const AssociationsIteratorRep& AIR ) const
  Key key( ) const
  Val value( ) const
  Bool next( )
  Bool done( ) const
  void reset( )
  void  write( ostream& ostr ) const;
  void  read( istream& istr );
private:
  const AssociationsOf<Key,Val>& theAssociations;
```

```
  AssociationsRep<Key,Val>::CellType* current;
};
```

———-

9.1.9 template AssociationsIterator

— template AssociationsIterator —

```
template <class Key, class Val> class AssociationsIterator :
public ObjectOf< AssociationsIteratorRep<Key,Val> > {
public:
  AssociationsIterator(const AssociationsOf<Key,Val>& A) :
    ObjectOf<AIR>( new AIR(A) )
  Bool operator == ( const AssociationsIterator& I ) const
  Key key( ) const
  Val value( ) const
  Bool next( )
  Bool done( ) const
  void reset( )
    AssociationsIterator& operator ++ ( )
    AssociationsIterator operator ++ ( int )
    operator void* ( )
  friend ostream& operator < ( ostream& ostr, const AssociationsIterator& AI )
  friend istream& operator > ( istream& istr, AssociationsIterator& AI)
private:
  typedef AssociationsIteratorRep<Key,Val> AIR;
};

template <class Key, class Val> inline ostream& operator <<
  ( ostream& o, const AssociationsOf<Key,Val>& a )
```

———-

9.1.10 template AssociationsOf¡Key,Val¿::keys( )

— AssociationsOf¡Key,Val¿::keys( ) —

```
template <class Key, class Val>
inline ListOf<Key> AssociationsOf<Key,Val>::keys( ) const
```

———-

9.2 general/include/BlackBox.h

— **BlackBox.h** —

```
#include "Chars.h"

extern "C" {
  char* tempnam(const char*,const char*);
}

\getchunk{class BlackBox}
```

———-

9.2.1 class BlackBox

— **class BlackBox** —

```
class BlackBox {
public:
  BlackBox(const Chars& startCommand, const Chars& restartCommand);
  BlackBox(const Chars& startCommand);
  ~BlackBox( );
  Bool start(const char* greeting = NULL);
  Bool restart(const char* greeting = NULL);
  Bool stillRunning( );
  ostream& toCommand( );
  istream& fromCommand( );
private:
  int status;
  Bool started_p;
  Chars theStartCmd;
  Chars theRestartCmd;
  ofstream* streamToCommand;
  ifstream* streamFromCommand;
  char fyle_in[100];
  char fyle_out[100];
  void initialize(const Chars& startCommand, const Chars& restartCommand);
  Bool doStart(const Chars& command, const char* greeting);
  void closeStreams( );
  Bool checkGreeting(const char* greeting);
  BlackBox(const BlackBox&)
  BlackBox& operator = ( const BlackBox& )
};
```

———-

9.3 general/include/BTree.h

— **BTree.h** —

```
#include "global.h"

\getchunk{template BTree}
\getchunk{template BTreeIterator}
\getchunk{template BTreePage}
```

———-

9.3.1 template BTreePage

— **template BTreePage** —

```
template <class Key, class Value> class BTreePage {
  friend class BTree<Key,Value>;
  friend class BTreeIterator<Key,Value>;
public:
  BTreePage( int order ) : theOrder( order ), numOfKeys( 0 ),
      parentLink( NULL )
  ~BTreePage( )
private:
  int theOrder;
  int numOfKeys;
  Key **keys;
  Value **values;
  BTreePage **links;
  BTreePage *parentLink;
  int numOfParentLink;
};
```

———-

9.3.2 template BTree

— **template BTree** —

```
template <class Key, class Value> class BTree {
  friend class BTreeIterator<Key,Value>;
public:
  BTree( int order = 6 ) : theOrder( order )
  BTree( const BTree& );
  BTree& operator = ( const BTree& )
  ~BTree( )
```

```
  bool remove( const Key& key );
  void insert( const Key& key, const Value& value );
  Value* search( const Key& key );
  friend ostream& operator << ( ostream& ostr, const BTree& T )
  friend ostream& operator < ( ostream& ostr, const BTree& T )
  friend istream& operator > ( istream& ostr, const BTree& T )
  friend bool operator == ( const BTree& T,const BTree& T1 )
    void printAll();
protected:
  virtual void theKeyIsFound( BTreePage<Key,Value>& keyPage, int position )
  bool search( const Key& key, const BTreePage<Key,Value>& searchPage,
         BTreePage<Key,Value> **keyPage, int& position );
  void deleteKey( BTreePage<Key,Value> *page, int position );
  void deleteAll( )
  void deleteAllPages( BTreePage<Key,Value> *page );
private:
  int theOrder;
  BTreePage<Key,Value> *root;
};
```

———-

9.3.3 template BTreeIterator

— **template BTreeIterator** —

```
template <class Key, class Value> class BTreeIterator {
  friend class BTree<Key,Value>;
public:
  BTreeIterator( const BTree<Key,Value>& T ) : tree( T )
  bool done( )
  void reset( );
  Value getValue( )
  Key getKey( )
  bool next( );
private:
  const BTree<Key,Value>& tree;
  BTreePage<Key,Value> *page;
  int linkNumber;
  bool bDone;
  Key *key;
  Value *value;
};
```

———-

9.4 general/include/Cell.h

— **Cell.h** —

```
#include "IPC.h"

template<class T>
ostream& operator < ( ostream& ostr, const Cell<T>& C )

template<class T>
istream& operator > ( istream& istr, Cell<T>& C )

\getchunk{template Cell}
```

————-

9.4.1 template Cell

— **template Cell** —

```
template<class T> class Cell {
public:
  Cell*  nextCell;
  Cell() : nextCell(NULL), contents(NULL)
  Cell(const Cell& C) : nextCell(NULL)
  Cell(const T& e, Cell* next = NULL) : nextCell(next)
  ~Cell()
  friend ostream& operator < <T>( ostream& ostr, const Cell& C );
  friend istream& operator > <T>( istream& istr, Cell& C );
  void readContents(istream& istr);
  void writeContents(ostream& ostr) const;
  inline T getContents()
  inline void setContents(const T& t)
private :
  T*     contents;
};
```

————-

9.5 general/include/Chars.h

— **Chars.h** —

```
#include "RefCounter.h"
#include "ObjectOf.h"
```

```
\getchunk{class CharsRep}
\getchunk{class Chars}

Chars ordinalPostfix(int num);
Chars encodeSpaces( const Chars& s );
Chars decodeSpaces( const Chars& s );
```

———-

9.5.1 class CharsRep

— class CharsRep —

```
class CharsRep : public RefCounter {
public:
  CharsRep( );
  CharsRep( const char* s );
  CharsRep( char c );
  CharsRep( const CharsRep& CR );
  CharsRep( int num );
  virtual ~CharsRep( );
  int len( ) const
  const char* getChars( ) const
  char& ref(int i);
  char val(int i) const;
  CharsRep* clone( ) const
  void concat( const char*, int len );
  void readFrom(istream& istr);
  virtual void write( ostream& ostr ) const;
  virtual void read( istream& istr );
private:
  CharsRep operator = ( const CharsRep& );
  int length;
  char* theChars;
  char special[2];
};
```

———-

9.5.2 class Chars

— class Chars —

```
class Chars : public ObjectOf<CharsRep> {
public:
  Chars( ) : ObjectOf<CharsRep>( new CharsRep() )
  Chars( const char* s ) : ObjectOf<CharsRep>( new CharsRep(s) )
```

```
  Chars( const char c ) : ObjectOf<CharsRep>( new CharsRep(c) )
  Chars ( int num ) : ObjectOf<CharsRep>( new CharsRep(num) )
  int length( ) const
  inline friend int operator == ( const Chars& l, const Chars& r )
  bool operator == ( const char * s) const
  inline friend int operator != ( const Chars& l, const Chars& r )
  inline friend int operator != ( const Chars& l, const char* r )
  Chars operator + (const Chars&) const;
  Chars operator + (const char*) const;
  Chars operator + (char) const;
  Chars& operator += (const Chars&);
  Chars& operator += (const char*);
  Chars& operator += (char);
  operator const char*( ) const
  char operator [] ( int i ) const
  char& operator [] ( int i )
  friend ostream& operator << ( ostream& ostr, const Chars& C )
  friend istream& operator >> ( istream& istr, Chars& C )
  friend ostream& operator < ( ostream& ostr, const Chars& c )
  friend istream& operator > ( istream& istr, Chars& c )
};
```

———-

9.6 general/include/conversions.h

— conversions.h —

```
#include "List.h"
#include "Set.h"
#include "Vector.h"

template <class T> VectorOf<T> makeVectorOf(const ListOf<T>& L)
template <class T> VectorOf<T> makeVectorOf(const SetOf<T>& S)
template <class T> SetOf<T> makeSetOf(const VectorOf<T>& V)
template <class T> SetOf<T> makeSetOf(const ListOf<T>& L)
template <class T> ListOf<T> makeListOf(const VectorOf<T>& V)
template <class T> ListOf<T> makeListOf(const SetOf<T>& S)
```

———-

9.7 general/include/DArray.h

— DArray.h —

```
#include "Vector.h"
```

```
#include "IStreamPoll.h"
#include "WordParser.h"

template <class R> class MatrixRep;
template <class R> class MatrixComputations;
template <class R> class DArrayParser;

\getchunk{template DArrayRep}

template <class R> class DArray;
template <class R> ostream& operator < ( ostream& ostr, const DArray<R>& DA )
template <class R> istream& operator > ( istream& istr, DArray<R>& DA)

\getchunk{template DArray}
\getchunk{template MatrixRow}
\getchunk{template WordParser}
\getchunk{template MatrixCell}
```

———-

9.7.1 template DArrayRep

— **template DArrayRep** —

```
template <class R> class DArrayRep : public PureRep {
public:
  DArrayRep( int height, int width): theArray(NULL)
  DArrayRep( int n ): theArray(NULL)
  DArrayRep( const DArrayRep& );
  ~DArrayRep( )
  DArrayRep<R>* clone( ) const
  bool operator == ( const DArrayRep& DA ) const;
  MatrixRow<R>& operator [](  int i )
  inline int getWidth( ) const
  inline int getHeight( ) const
  VectorOf<R> getRow(int i) const;
  VectorOf<R> getCol(int i) const;
  void assignCol(int i,const VectorOf<R>& vc);
  void reSize(int newHeight,int newWidth);
  void colInsBefore(int col,int colsNum);
  void colInsAfter(int col,int colsNum);
  void rowInsBefore(int col,int colsNum);
  void rowInsAfter(int col,int colsNum);
  void colDelBefore(int col,int colsNum, bool reAlloc);
  void colDelAfter(int col,int colsNum, bool reAlloc);
  void colDelRange(int col1,int col2, bool reAlloc);
  void rowDelBefore(int row,int rowsNum, bool reAlloc);
  void rowDelAfter(int row,int rowsNum, bool reAlloc);
```

```
  void rowDelRange(int row1,int row2, bool reAlloc);
  bool readFrom( istream& istr, Chars& errMesg )
  virtual void write( ostream& ostr ) const
  virtual void read( istream& istr )
private:
  friend class MatrixRep<R>;
  friend class MatrixComputations<R>;
  int width, height;
  int rowBegin, colBegin;
  MatrixRow<R>* theArray;
  void makeArray( int mHeight, int mWidth);
  void deleteArray( );
  bool inBounds(int i) const
};
```

———-

9.7.2 template DArray

— **template DArray** —

```
template <class R> class DArray : public ObjectOf< DArrayRep<R> > {
public:
  DArray( int n = 0 ) : ObjectOf<DArrayRep <R> >( new DArrayRep<R>(n) )
  DArray( int h, int w) : ObjectOf<DArrayRep <R> >( new DArrayRep<R>(h,w) )
  bool operator == ( const DArray& DA ) const
  VectorOf<R> operator [](  int i ) const
  MatrixRow< R >& operator [] ( int i )
  VectorOf<R> getRow( int i ) const
  VectorOf<R> getCol( int i ) const
  void assignCol( int i, const VectorOf<R>& vc )
  void reSize(int newHeight,int newWidth)
  void colInsBefore(int col,int colsNum)
  void colInsAfter(int col,int colsNum)
  void rowInsBefore(int row,int rowsNum)
  void rowInsAfter(int row,int rowsNum)
  void colDelBefore(int col,int colsNum,bool reAlloc = TRUE)
  void colDelAfter(int col,int colsNum, bool reAlloc =TRUE)
  void colDelRange(int col1,int col2, bool reAlloc =TRUE)
  void rowDelBefore(int row,int rowsNum,bool reAlloc = TRUE)
  void rowDelAfter(int row,int rowsNum, bool reAlloc =TRUE)
  void rowDelRange(int row1,int row2, bool reAlloc =TRUE)
  int width( ) const
  int height( ) const
  int lastCol( ) const
  int lastRow( ) const
  friend IStreamPoll operator >> ( istream& istr, DArray& M )
  inline friend ostream& operator << ( ostream& o, const DArray& v )
```

```
  friend ostream& operator < <R>( ostream& ostr, const DArray& DA );
  friend istream& operator > <R>( istream& istr, DArray& DA);
protected :
  DArray( DArrayRep<R>* newrep ) : ObjectOf<DArrayRep <R> >(newrep)
};
```

———-

9.7.3 template MatrixRow

— **template MatrixRow** —

```
template <class R> class MatrixRow {
public:
   R& operator [] (int i)
   MatrixRow<R>& operator = (const MatrixRow<R>& mr)
   MatrixRow<R>& operator = (const VectorOf<R>& vc)
   operator VectorOf<R>()
   int length() const
private:
   friend class DArrayParser<R>;
   friend class DArrayRep<R>;
   friend class MatrixCell<R>;
   MatrixRow(const int& len) : width(len)
   MatrixRow(const MatrixRow<R>& mr)
   MatrixRow(const VectorOf<R>& vc)
  ~MatrixRow()
   R* row;
   int width, rowBegin;
};
```

———-

9.7.4 template WordParser

— **template WordParser** —

```
template <class R> class DArrayParser : public WordParser {
public:
  DArrayParser(istream &istr) : WordParser(istr)
  bool parseDArray(Chars& errMesg,MatrixRow<R> **M,int& MHeight,int& MWidth );
protected:
  VectorOf<R> parseMatrixRow( Chars& errMesg );
private:
  void deleteRows(  MatrixCell<R>* rows);
};
```

———-

9.7.5 template MatrixCell

— template MatrixCell —

```
template <class R> class MatrixCell {
 private:
   friend class DArrayParser<R>;
   MatrixCell(const MatrixRow<R>& mr):content(mr),nextCell(NULL){}
   MatrixRow<R> content;
   MatrixCell* nextCell;
};
```

———-

9.8 general/include/DCell.h

— DCell.h —

```
#include <iostream.h>
#include "IPC.h"

\getchunk{template DCell}
```

———-

9.8.1 template DCell

— template DCell —

```
template<class T> class DCell {
public:
  T      contents;
  DCell*  previousCell;
  DCell*  nextCell;
  DCell() {nextCell = previousCell = NULL; }
  DCell(const DCell& C) : contents(C.contents)
  DCell(const T& e, DCell* previous = NULL, DCell* next = NULL) :
  contents(e)
  ~DCell()
  friend ostream& operator < ( ostream& ostr, const DCell& DC )
  friend istream& operator > ( istream& istr, DCell& DC )
};
```

———-

9.9 general/include/DList.h

— **DList.h** —

```
#include "RefCounter.h"
#include "ObjectOf.h"
#include "DCell.h"

\getchunk{template DListRep}
\getchunk{template DListOf}
\getchunk{template DListIteratorRep}
\getchunk{template DListIterator}
```

———-

9.9.1 template DListRep

— **template DListRep** —

```
template <class T> class DListRep : public RefCounter {
public:
  DListRep( ) : root(NULL), last(NULL), len(0)
  DListRep( const T& t )
  DListRep( const DListRep& lr ) : len(lr.len)
  ~DListRep( )
  Bool operator == ( const DListRep& lr ) const
  DListRep* clone( )
  void insert( const T& t, int i )
  void insert( const DListRep* LR, int i )
  void removeElement( const T& t )
  void removeElementOfIndex( int i )
  void splice( const DListRep* rp, int i, int j )
  DListRep* sublist( int i, int j ) const
  DListRep* concatenate( const DListRep* rp ) const
  DListRep* reverse( ) const
  int length( ) const
  T element( int i ) const
  int indexOf( const T& t ) const
  int agreement( const DListRep* rp, int start ) const
  void printOn( ostream& ostr ) const
  void  write( ostream& ostr ) const;
  void  read( istream& istr );
private:
  DListRep& operator = ( const DListRep& );
  typedef DCell<T> CellType;
  friend class DListIteratorRep<T>;
  CellType* root;
  CellType* last;
```

```
  int len;
};
```

———-

9.9.2 template DListOf

— template DListOf —

```
template <class T> class DListOf : ObjectOf< DListRep<T> > {
public:
  typedef T ListElementType;
  DListOf( ) : ObjectOf<Rep>( new Rep() )
  DListOf( const T& t ) : ObjectOf<Rep>( new Rep(t) )
  Bool operator == ( const DListOf& L ) const
  Bool operator != ( const DListOf& L ) const
  T operator [ ] ( int i ) const
  Bool equalTo( const DListOf& L ) const
  T element( int i ) const
  int length( ) const
  Bool contains( const T& t ) const
  int indexOf( const T& t ) const
  Bool prefixOf( const DListOf& L ) const
  Bool properPrefixOf( const DListOf& L ) const
  int agreement( const DListOf& L, int start = 0 ) const
  DListOf sublist( int i, int j ) const
  friend DListOf concatenate( const DListOf& L1, const DListOf& L2 )
  DListOf reverse( ) const
  void append( const T& t )
  void prepend( const T& t )
  void insert( const T& t, int i )
  void insert( const DListOf& L, int i )
  void removeElement( const T& t )
  void removeElementOfIndex( int i )
  void splice( const DListOf& L, int i, int j )
  inline friend ostream& operator << ( ostream& ostr, const DListOf& l )
  friend ostream& operator < ( ostream& ostr, const DListOf& DL )
  friend istream& operator > ( istream& istr, DListOf& DL)
private:
  typedef DCell<T> CellType;
  typedef DListRep<T> Rep;
  friend class DListIteratorRep<T>;
  typedef ObjectOf<Rep> R;
  DListOf( Rep* p ) : R(p) { }
};
```

———-

9.9.3 template DListIteratorRep

— template DListIteratorRep —

```
template <class T> class DListIteratorRep : public RefCounter {
public:
  DListIteratorRep( const DListOf<T>& L ) : theList(L)
  DListIteratorRep *clone() const
  Bool operator == ( const DListIteratorRep& LIR ) const
  T value( ) const
  Bool next( )
  Bool previous( )
  Bool done( ) const
  void reset( )
  void setToEnd( )
  void  write( ostream& ostr ) const;
  void  read( istream& istr );
private:
  const DListOf<T>& theList;
  DCell<T>* current;
};
```

———-

9.9.4 template DListIterator

— template DListIterator —

```
template <class ListType> class DListIterator :
public ObjectOf< DListIteratorRep<typename ListType::ListElementType> > {
public:
  typedef typename ListType::ListElementType T;
  DListIterator(const DListOf<T>& L) : ObjectOf<LIR>( new LIR(L) )
  Bool operator == ( const DListIterator& I ) const
  T value( ) const
  Bool next( )
  Bool previous( )
  Bool done( ) const
  Bool valid( ) const
  void reset( )
  void setToEnd( )
  friend ostream& operator < ( ostream& ostr, const DListIterator& DLI )
  friend istream& operator > ( istream& istr, DListIterator& DLI)
private:
  typedef DListIteratorRep<T> LIR;
};
```

———-

9.10 general/include/File.h

— File.h —

```
#include <stdio.h>
#include <fstream.h>
#include "MagnusHome.h"

const Chars mainColor = "000";
const Chars titleColor = "aaa";

\getchunk{struct File}

inline ostream& ready(ostream& ostr)
inline ostream& end(ostream& ostr)
```

———-

9.10.1 struct File

— struct File —

```
struct File : public fstream {
  File( )
  File( const Chars& fileName ) :
    fstream( fileName, ios::app ), theFileName(fileName)
  Chars getFileName() const
  void setColor( const Chars& colorName )
private:
  Chars theFileName;
  fstream colorFile;
};
```

———-

9.11 general/include/GCD.h

— GCD.h —

```
#include "Integer.h"

int GCD(int a, int b);
Integer GCD(Integer a, Integer b);
int GCD(int a, int b, int& p, int& q);
Integer GCD(Integer a, Integer b, Integer& p, Integer& q);
int LCM(int a, int b);
```

```
Integer LCM(Integer a, Integer b);
```

———-

9.12 general/include/Int2.h

— **Int2.h** —

```
#include "global.h"

\getchunk{struct Int2}
```

———-

9.12.1 struct Int2

— **struct Int2** —

```
struct Int2 {
  Int2( ) : v(0) { }
  Int2( int x ) : v(x) { }
  int compare( Int2 x ) const
  friend ostream& operator << ( ostream& ostr, const Int2& i )
  bool operator == ( const Int2& i ) const
  bool operator != ( const Int2& i ) const
  Int2 operator - ( ) const
  Int2 operator + ( const Int2& i ) const
  Int2& operator += ( const Int2& i )
  Int2 operator * ( const Int2& i ) const
  Int2& operator *= ( const Int2& i )
  bool operator < ( const Int2& i ) const
  bool operator > ( const Int2& i ) const
  int value( ) const
private:
  int v;
};
```

———-

9.13 general/include/IStreamPoll.h

— **IStreamPoll.h** —

```
#include "Chars.h"

\getchunk{class IStreamPoll}
```

———-

9.13.1 class IStreamPoll

— class IStreamPoll —

```
class IStreamPoll {
public :
  IStreamPoll( istream& is, Chars em = "" ) : istr(is), errMesg(em)
  typedef istream& IstreamRef;
  operator int( )
  operator Chars( )
  operator IstreamRef( )
private :
  istream& istr;
  Chars errMesg;
};
```

———-

9.14 general/include/List.h

— List.h —

```
#include "RefCounter.h"
#include "ObjectOf.h"
#include "Cell.h"

\getchunk{template ListRep}
\getchunk{template ListOf}
\getchunk{template ListIteratorRep}
\getchunk{template ListIterator}
```

———-

9.14.1 template ListRep

— template ListRep —

```
template <class T> class ListRep : public RefCounter {
public:
  ListRep( ) : root(NULL), last(NULL), len(0)
  ListRep( const T& t )
  ListRep( const ListRep& lr ) : len(lr.len)
  ~ListRep( )
  Bool operator == ( const ListRep& lr ) const
  ListRep* clone( )
  void insert( const T& t, int i )
  void removeElement( const T& t )
  void removeElementOfIndex( int i )
  void splice( const ListRep* rp, int i, int j )
  ListRep* sublist( int i, int j ) const
  ListRep* concatenate( const ListRep* rp ) const
  ListRep* reverse( ) const
  int length( ) const
  T element( int i ) const
  int indexOf( const T& t ) const
  int agreement( const ListRep* rp, int start ) const
  void printOn( ostream& ostr ) const
  void  write( ostream& ostr ) const;
  void  read( istream& istr );
private:
  ListRep& operator = ( const ListRep& );
  typedef Cell<T> CellType;
  friend class ListIteratorRep<T>;
  CellType* root;
  CellType* last;
  int len;
};
```

———-

9.14.2 template ListOf

— template ListOf —

```
template <class T> class ListOf : ObjectOf< ListRep<T> > {
public:
  typedef T ListElementType;
  ListOf( ) : ObjectOf<Rep>( new Rep() )
  ListOf( const T& t ) : ObjectOf<Rep>( new Rep(t) )
  Bool operator == ( const ListOf& L ) const
  Bool operator != ( const ListOf& L ) const
  T operator [ ] ( int i ) const
  Bool equalTo( const ListOf& L ) const
  T element( int i ) const
  int length( ) const
```

```
  Bool contains( const T& t ) const
  int indexOf( const T& t ) const
  Bool prefixOf( const ListOf& L ) const
  Bool properPrefixOf( const ListOf& L ) const
  int agreement( const ListOf& L, int start = 0 ) const
  ListOf sublist( int i, int j ) const
  friend ListOf concatenate( const ListOf& L1, const ListOf& L2 )
  ListOf reverse( ) const
  void append( const T& t )
  void prepend( const T& t )
  void insert( const T& t, int i )
  void removeElement( const T& t )
  void removeElementOfIndex( int i )
  void splice( const ListOf& L, int i, int j )
  inline friend ostream& operator << ( ostream& ostr, const ListOf& l )
  friend ostream& operator < ( ostream& ostr, const ListOf& L )
  friend istream& operator > ( istream& istr, ListOf& L)
private:
  typedef Cell<T> CellType;
  typedef ListRep<T> Rep;
  friend class ListIteratorRep<T>;
  typedef ObjectOf<Rep> R;
  ListOf( Rep* p ) : R(p)
};
```

———-

9.14.3 template ListIteratorRep

— template ListIteratorRep —

```
template <class T> class ListIteratorRep : public RefCounter {
public:
  ListIteratorRep( const ListOf<T>& L ) : theList(L)
  ListIteratorRep *clone() const
  Bool operator == ( const ListIteratorRep& LIR ) const
  T value( ) const
  Bool next( )
  Bool done( ) const
  void reset( )
  void  write( ostream& ostr ) const;
  void  read( istream& istr );
private:
  const ListOf<T>& theList;
  Cell<T>* current;
};
```

———-

9.14.4 template ListIterator

— **template ListIterator** —

```
template <class ListType> class ListIterator :
  public ObjectOf< ListIteratorRep<typename ListType::ListElementType> > {
public:
  typedef typename ListType::ListElementType T;
  ListIterator(const ListOf<T>& L) : ObjectOf<LIR>( new LIR(L) )
  Bool operator == ( const ListIterator& I ) const
  T value( ) const
  Bool next( )
  Bool done( ) const
  void reset( )
  friend ostream& operator < ( ostream& ostr, const ListIterator& LI )
  friend istream& operator > ( istream& istr, ListIterator& LI)
private:
  typedef ListIteratorRep<T> LIR;
};
```

———-

9.15 general/include/LogWatcher.h

— **LogWatcher.h** —

```
#include <iostream.h>
#include "Chars.h"
#include "Timer.h"

const int WATCHER_BUF_SIZE = 4096;

\getchunk{class LogFileWatcher}
```

———-

9.15.1 class LogFileWatcher

— **class LogFileWatcher** —

```
class LogFileWatcher
{
public:
  LogFileWatcher( const Chars& fname, int timeInterval );
  ~LogFileWatcher( );
  bool changed();
```

```
  Chars getline();
  int getInterval( ) const
  void setInterval( int timeInterval )
private:
  long fileSize( );
  char peekCh( )
  char getCh( )
  fstream file;
  Chars fileName;
  long filePos;
  Chars line;
  char *buffer;
  int  bufpos;
  bool completeLine;
  Timer timer;
  int interval;
private:
  LogFileWatcher( const LogFileWatcher& watch );
  LogFileWatcher& operator=( const LogFileWatcher& watch );
};
```

———-

9.16 general/include/MagnusHome.h

— **MagnusHome.h** —

```
#include "config.h"
#include "Chars.h"

\getchunk{struct MagnusHome}
\getchunk{struct MagnusTmp}
```

———-

9.16.1 struct MagnusHome

— **struct MagnusHome** —

```
struct MagnusHome {
  friend int main(int argc, char* argv[]);
  static Chars magnusHome( )
private:
  static char* magnus_home;
};
```

———-

9.16.2 struct MagnusTmp

— **struct MagnusTmp** —

```
struct MagnusTmp {
  friend int main(int argc, char* argv[]);
  static Chars magnusTmp( )
private:
  static char* magnus_tmp;
};
```

———-

9.17 general/include/QuickAssociations.h

— **QuickAssociations.h** —

```
#include "RefCounter.h"
#include "ObjectOf.h"
#include "List.h"
#include "Set.h"

\getchunk{template QuickAssociation}

template <class Key, class Val>
  inline ostream& operator <<(ostream& o, const QuickAssociation<Key,Val>& qa)

\getchunk{template QuickAssociationsRep}
\getchunk{template QuickAssociationsOf}
\getchunk{template QuickAssocRef}

template <class Key, class Val> class QuickAssociationsIteratorRep;

template <class Key, class Val> inline
  QuickAssocRef<Key,Val> QuickAssociationsOf<Key,Val>::operator [ ]
    ( const Key& k )

\getchunk{template QuickAssociationsIteratorRep}
\getchunk{template QuickAssociationsIterator}

template <class Key, class Val> inline ostream& operator <<
  ( ostream& o, const QuickAssociationsOf<Key,Val>& a )

template <class Key, class Val>
  inline ListOf<Key> QuickAssociationsOf<Key,Val>::keys( ) const
```

———-

9.17.1 template QuickAssociation

— template QuickAssociation —

```
template <class Key, class Val> class QuickAssociation {
public:
  QuickAssociation(const Key& k, const Val& v) : key(k)
  QuickAssociation(const QuickAssociation& qa) : key(qa.key)
  QuickAssociation& operator=( const QuickAssociation& qa )
  ~QuickAssociation()
  int hash() const
  friend inline int operator
    ==(const QuickAssociation<Key,Val>& x, const QuickAssociation<Key,Val>& y)
  Key key;
  Val *val;
private:
  friend class QuickAssociationsRep<Key,Val>;
  QuickAssociation(const Key& k) : key(k), val(0)
```

———-

9.17.2 template QuickAssociationsRep

— template QuickAssociationsRep —

```
template <class Key, class Val> class QuickAssociationsRep :
    public SetData< QuickAssociation<Key,Val> > {
  typedef QuickAssociation<Key,Val> EltType;
public:
  QuickAssociationsRep( int size = 1 ) : Base (size)
  QuickAssociationsRep( const QuickAssociationsRep& ar ) : Base(ar)
  QuickAssociationsRep* clone( ) const
  void unbind( const Key& k )
  void bind( const Key& k, const Val& v )
  Val val( const Key& k ) const
  bool bound( const Key& k ) const
private:
  typedef SetData< QuickAssociation<Key,Val> > Base;
  friend class QuickAssociationsIteratorRep<Key,Val>;
  const bool seek( const Key& k, EltType* &elt ) const
};
```

———-

9.17.3 template QuickAssociationsOf

— template QuickAssociationsOf —

```
template <class Key, class Val> class QuickAssociationsOf :
  public ObjectOf< QuickAssociationsRep<Key,Val> > {
public:
  QuickAssociationsOf( ) : ObjectOf<Rep>(new Rep())
  Val operator [ ] ( const Key& k ) const
  QuickAssocRef<Key,Val> operator [ ] ( const Key& k );
  Val valueOf( const Key& k ) const
  void bind( const Key& k, const Val& v )
  void unbind( const Key& k )
  bool bound( const Key& k ) const
  int cardinality() const
  ListOf<Key> keys( ) const;
private:
  typedef QuickAssociationsRep<Key,Val> Rep;
  friend class QuickAssociationsIteratorRep<Key,Val>;
};
```

———-

9.17.4 template QuickAssocRef

— **template QuickAssocRef** —

```
template <class Key, class Val> struct QuickAssocRef {
  QuickAssociationsOf<Key,Val>& asref;
  const Key key;
  QuickAssocRef( QuickAssociationsOf<Key,Val>& a, const Key& k ) : asref(a), key(k)
  const Val& operator = ( const Val& val )
  operator Val ( )
  operator void* ( )
};
```

———-

9.17.5 template QuickAssociationsIteratorRep

— **template QuickAssociationsIteratorRep** —

```
template <class Key, class Val> class QuickAssociationsIteratorRep :
  public RefCounter {
public:
  QuickAssociationsIteratorRep( const QuickAssociationsOf<Key,Val>& A )
    :  theAssociations(A)
  QuickAssociationsIteratorRep *clone() const
  bool operator == ( const QuickAssociationsIteratorRep& QAIR ) const
  Key key( ) const
  Val value( ) const
```

```
  bool next( )
  bool done( ) const
  void reset( )
protected:
  typedef Cell< QuickAssociation<Key,Val> > CellType;
  CellType*      current;
  int            bucketIndex;
  const QuickAssociationsOf<Key,Val> theAssociations;
};
```

———-

9.17.6 template QuickAssociationsIterator

— **template QuickAssociationsIterator** —

```
template <class Key, class Val> class QuickAssociationsIterator :
public ObjectOf< QuickAssociationsIteratorRep<Key,Val> > {
public:
  QuickAssociationsIterator(const QuickAssociationsOf<Key,Val>& A) :
    ObjectOf<AIR>( new AIR(A) )
  bool operator == ( const QuickAssociationsIterator& I ) const
  Key key( ) const
  Val value( ) const
  bool next( )
  bool done( ) const
  void reset( )
  QuickAssociationsIterator& operator ++ ( )
  QuickAssociationsIterator operator ++ ( int )
  operator void* ( )
private:
  typedef QuickAssociationsIteratorRep<Key,Val> AIR;
};
```

———-

9.18 general/include/RandomNumbers.h

— **RandomNumbers.h** —

```
#include <iostream.h>
#include "IPC.h"
#include "Timer.h"

inline long TimeSeed()
```

```
\getchunk{class UniformRandom}
\getchunk{class NormalRandom}
```

———-

9.18.1 class UniformRandom

— class UniformRandom —

```
class UniformRandom {
public:
  UniformRandom(int seed = TimeSeed())
  void reseed(int seed);
  float rand( );
  int rand(int lower, int upper);
  friend ostream& operator < ( ostream& ostr, const UniformRandom& unRand )
  friend istream& operator > ( istream& istr, UniformRandom& unRand)
private:
  static const int   M1  = 259200;
  static const int   IA1 = 7141;
  static const int   IC1 = 54773;
  static const float RM1 = (1.0/M1);
  static const int   M2  = 134456;
  static const int   IA2 = 8121;
  static const int   IC2 = 28411;
  static const float RM2 = (1.0/M2);
  static const int   M3  = 243000;
  static const int   IA3 = 4561;
  static const int   IC3 = 51349;
  long ix1, ix2, ix3;
  float r[98];
};
```

———-

9.18.2 class NormalRandom

— class NormalRandom —

```
class NormalRandom {
public:
  NormalRandom(int seed = TimeSeed()) : uniformDeviates(seed), iset(0)
  void reseed(int seed)
  float rand( );
  int rand(int mean, int stddev);
  friend ostream& operator < ( ostream& ostr, const NormalRandom& noRand )
  friend istream& operator > ( istream& istr, NormalRandom& noRand)
```

```
private:
  UniformRandom uniformDeviates;
  int iset;
  float gset;
};
```

———-

9.19 general/include/Set.h

— **Set.h** —

```
#include "global.h"
#include "RefCounter.h"
#include "PureRep.h"
#include "ObjectOf.h"
#include "Cell.h"

template <class T> class SetIteratorData;

#define FULLNESS_FACTOR 2

\getchunk{template SetData}
\getchunk{template SetOf}
```

———-

9.19.1 template SetData

— **template SetData** —

```
template<class T> class SetData : public RefCounter {
public:
  SetData(int size)
  SetData( const T& t )
  SetData(const SetData& sd)
  ~SetData()
  SetData* clone( ) const
  Bool operator == ( const SetData& sd ) const
  int hashElement(const T & t) const;
  void rehash( Bool calledByUser = FALSE )
  void removeAllElements()
  int cardinality() const
  Bool contains(const T& e) const
  void adjoinElement(const T& e)
  void removeElement(const T& e)
```

```
  void printOn(ostream& ostr) const
  void write( ostream& ostr ) const;
  void read( istream& istr );
protected:
  typedef Cell<T> CellType;
  friend class SetIteratorData<T>;
  int         userSize;
  int         numberOfElements;
  int         numberOfBuckets;
  CellType**  hashTable;
};
```

———-

9.19.2 template SetOf

— **template SetOf** —

```
template<class T> class SetOf : public ObjectOf< SetData<T> > {
public:
  typedef T SetElementType;
  SetOf( int size = 1 ) : ObjectOf<Rep>( new Rep(size) )
  SetOf( const T& t ) : ObjectOf<Rep>( new Rep(t) )
  Bool operator == ( const SetOf& S ) const
  Bool operator != ( const SetOf& S ) const
  Bool operator < ( const SetOf& S ) const
  Bool operator <= ( const SetOf& S ) const
  Bool operator > ( const SetOf& S ) const
  Bool operator >= ( const SetOf& S ) const
  Bool operator >= ( const T& e ) const
  SetOf operator & ( const SetOf& S ) const
  SetOf operator &= ( const SetOf& S )
  SetOf operator | ( const SetOf& S ) const
  SetOf operator |= ( const SetOf& S )
  SetOf operator |= ( const T& e )
  SetOf operator - ( const SetOf& S ) const
  SetOf operator -= ( const SetOf& S )
  SetOf operator -= ( const T& e )
  SetOf operator ^ ( const SetOf& S ) const
  int cardinality() const
  Bool equalTo(const SetOf& S) const
  Bool contains(const T& e) const
  Bool contains(const SetOf& S) const;
  Bool properlyContains(const SetOf& S) const
  void adjoinElement(const T& e)
  void removeElement(const T& e)
  void shrinkToIntersectionWith(const SetOf& S);
  void adjoinElements(const SetOf& S);
```

```
  void removeElements(const SetOf& S);
  void removeAllElements()
  void rehash( )
  friend ostream& operator << ( ostream& ostr, const SetOf& S )
  friend ostream& operator < ( ostream& ostr, const SetOf& S )
  friend istream& operator > ( istream& istr, SetOf& S )
protected:
  typedef SetData<T> Rep;
  friend class SetIteratorData<T>;
};

template <class T> SetOf<T> setUnion(const SetOf<T>&, const SetOf<T>&);
template <class T> SetOf<T> setIntersection(const SetOf<T>&, const SetOf<T>&);
template <class T> SetOf<T> setMinus(const SetOf<T>&, const SetOf<T>&);
template <class T> SetOf<T>
  setSymmetricDifference(const SetOf<T>&, const SetOf<T>&);

template<class T>
class SetContainer : public ObjectOf< SetData<T> > {
public:
  friend class SetIteratorData<T>;
  SetContainer( const SetOf<T>& S ) : ObjectOf< SetData<T> >(S) { }
};

template<class T>
class SetIteratorData : public PureRep {
public:
  SetIteratorData(const SetOf<T>& S) : theSet(S)
  PureRep *clone() const
  Bool operator == ( const SetIteratorData& sid ) const
  T value( ) const
  Bool next( )
  Bool done( ) const
  void reset( )
private:
  typedef Cell<T> CellType;
  CellType*      current;
  int            bucketIndex;
  const SetContainer<T> theSet;
};

template <class T>
class SetIterator : public ObjectOf< SetIteratorData<T> > {
public:
  SetIterator(const SetOf<T>& S) : ObjectOf<SID>( new SID(S) )
  Bool operator == ( const SetIterator& I ) const
  T value( ) const
  Bool next( )
  Bool done( ) const
  void reset( )
```

```
protected:
  typedef SetIteratorData<T> ThisRep;
  typedef ObjectOf<ThisRep> Base;
  const ThisRep* look( ) const
  ThisRep* enhance( ) const
  ThisRep* change( )
  SetIterator( ThisRep* rep ) : Base(rep)
private:
  typedef SetIteratorData<T> SID;
};

template <class T>
SetOf<T> setUnion(const SetOf<T>& S1, const SetOf<T>& S2)

template <class T>
SetOf<T> setIntersection(const SetOf<T>& S1, const SetOf<T>& S2)

template <class T>
SetOf<T> setMinus(const SetOf<T>& S1, const SetOf<T>& S2)

template <class T>
SetOf<T> setSymmetricDifference(const SetOf<T>& S1, const SetOf<T>& S2)
```

———-

9.20 general/include/Stack.h

— **Stack.h** —

```
#include "RefCounter.h"
#include "ObjectOf.h"
#include "List.h"
#include "Cell.h"

template < class T > class StackOf : public ObjectOf< ListRep<T> > {
public:
  StackOf( ) : ObjectOf< ListRep<T> >( new ListRep<T>() )
  StackOf( const T& t ) : ObjectOf< ListRep<T> >( new ListRep<T>(t) )
  void push( const T& t )
  T pop( )
  void popAll( )
  Bool isEmpty( ) const
  Bool isntEmpty( ) const
};
```

———-

9.21 general/include/Timer.h

— **Timer.h** —

```
#include <sys/time.h>
#include <iostream.h>
#include <IPC.h>
#include "config.h"
```

———-

9.21.1 class Timer

— **class Timer** —

```
class Timer {
public:
  Timer(int miliseconds)
  bool expired( ) const
  void reset(int miliseconds)
  friend ostream& operator < ( ostream& ostr, const Timer& T )
  friend istream& operator > ( istream& istr, Timer& T)
private:
  long alarmSecs, alarmUSecs;
};
```

———-

9.22 general/include/Type.h

— **Type.h** —

———-

9.22.1 class Type

— **class Type** —

```
class Type {
public :
  class notype { };
  class unique { };
  Type( const notype& ) : typeNumber(0)
```

```
  Type( const unique& ) : typeNumber(++uniqueTypeNumber)
  int operator== ( const Type& t)
  int operator!= ( const Type& t)
  void writeTo( ostream& o ) const
  void readMeFrom( istream& i )
private :
  int typeNumber;
  static int uniqueTypeNumber;
};

inline ostream& operator << ( ostream& o, const Type& t )

inline istream& operator >> ( istream& i, Type& t )
```

———-

9.23 general/include/Vector.h

— **Vector.h** —

```
#include "global.h"
#include "RefCounter.h"
#include "ObjectOf.h"
#include "Chars.h"
#include "ExtendedIPC.h"

template <class T> struct VectorRep : public RefCounter {
  public :
  VectorRep( const VectorRep& vr )
  VectorRep( int l )
  VectorRep( int l, bool e )
  ~VectorRep( )
  VectorRep* clone( )
  int length() const
  T& ref(int i)
  const T& constref(int i) const
  T val(int i) const
  void append( const T& t )
  void prepend( const T& t )
  void shrink( int start, int newlen )
  void write( ostream& ostr ) const
  void read( istream& istr )
private :
  VectorRep& operator = ( const VectorRep& );
  bool fastExpansion;
  unsigned int first;
  unsigned int last;
  unsigned int len;
```

```
  T* vec;
};

template <class T> class VectorOf;

template<class T>
ostream& operator < ( ostream& ostr, const VectorOf<T>& v )

template<class T>
istream& operator > ( istream& istr, VectorOf<T>& v )

template <class T> class VectorOf : public ObjectOf< VectorRep<T> > {
  typedef VectorRep< T > Tvr;
  typedef ObjectOf< Tvr > Rep;
public:
  VectorOf( int len = 0 ) : Rep( new Tvr(len) )
  VectorOf( int len, bool e ) : Rep( new Tvr(len,e) )
  VectorOf( int len, const VectorOf& v ) : Rep( new Tvr(len) )
  VectorOf( int len, bool e, const VectorOf& v ) : Rep( new Tvr(len,e) )
  int operator == ( const VectorOf& v ) const
  int operator != ( const VectorOf& v ) const
  T operator [] ( int i ) const
  T& operator [] ( int i )
  T val( int i ) const
  T& ref( int i )
  const T& constref( int i ) const
  int length( ) const
  int hash() const
  int indexOf( const T& t ) const
  void append( const T& t )
  void prepend( const T& t )
  void shrink( int newLength )
  void shrink( int start, int newLength )
  inline friend ostream& operator << ( ostream& o, const VectorOf& v )
  friend ostream& operator < <T>( ostream& ostr, const VectorOf& v );
  friend istream& operator > <T>( istream& istr, VectorOf& v );
private:
};

template <class T>
inline VectorOf<T> concatenate(const VectorOf<T>& v1, const VectorOf<T>& v2)
```

———-

9.24 general/include/WordParser.h

— WordParser.h —

```
#include "Word.h"
#include "Chars.h"
#include "Vector.h"

#define NAME_SIZE      100
#define INPUT_BUF_SIZE 1024
#define MAX_WORD_LENGTH 2000000000

typedef enum
{
   LANGLE, RANGLE, LPAREN, RPAREN, LSQUARE, RSQUARE, STAR, CARET, COMMA, BAR,
   COLON, SEMICOLON, EQUALS, GENERATOR, INT, EOS, BAD, INIT, LSET, RSET,
   ARROW, DOT
}  TokenType;
```

———-

9.24.1 class WordParser

— class WordParser —

```
class WordParser {
public:
  WordParser(istream &str) : istr(str)
  void popToken( )
  Word parseWord( const VectorOf<Chars>&, Chars& );
  Word parseWordVerbatim( const VectorOf<Chars>&, Chars& );
protected:
  int          tokenInt;
  int          tokenBufIndex;
  char         tokenName[NAME_SIZE+1];
  char         tokenBuf[INPUT_BUF_SIZE];
  TokenType    curToken;
  VectorOf<Chars>  genNames;
  Chars          parseErrorMessage;
  istream&       istr;
  char peekCh( );
  char getCh( );
  virtual void getToken( );
  void parseError(const char*);
  Bool atStartOfWord( );
  void invertName(char*);
  bool isInvertibleName( char* );
  ParseNode *parseExpression( );
  ParseNode *parseTerm( );
  ParseNode *parseAtom( );
};
```

```
typedef enum {
  ILLEGAL_NODE, COMMUTATOR_NODE, INT_NODE, CONCAT_NODE,
  IDENT_NODE, POWER_OR_CONJUGATE_NODE
} ParseNodeType;

typedef enum {
  PT_OK = 0, PT_TOO_LONG
} ParseTypeState;

typedef int ParseData;
```

———-

9.24.2 class ParseType

— class ParseType —

```
class ParseType {
public:
  ParseType( ) : len(0), data(0), status(PT_OK)
  ParseType( int gen );
  Word makeWord( ) const;
  ParseType invertWord( ) const;
  ParseType makeCopies( unsigned int numOfCopies ) const;
  static ParseType join( const ParseType& x, const ParseType& y );
  void destroy( )
  ParseTypeState state( ) const
  Chars errorMessage( ) const;
private:
  ParseType( ParseTypeState s, ParseData *ptr, int plen )
    : len(plen), data(ptr), status(s)
  int len;
  ParseData *data;
  ParseTypeState status;
};
```

———-

9.24.3 class ParseNode

— class ParseNode —

```
class ParseNode
{
public:
  ParseNode( )
  virtual ~ParseNode( )
```

```
  virtual ParseType eval( ) = 0;
  virtual ParseNodeType whatAmI( ) = 0;
};
```

———-

9.24.4 class Int

— class Int —

```
class Int : public ParseNode {
public:
  Int( int n )
  ParseType eval( )
  ParseNodeType  whatAmI( )
  int intEval( )
private:
  int number;
};
```

———-

9.24.5 class Ident

— class Ident —

```
class Ident : public ParseNode {
public:
  Ident( ParseType val )
  ~Ident( )
  ParseNodeType  whatAmI( )
  ParseType eval( )
private:
  ParseType  gen;
};
```

———-

9.24.6 class BinOp

— class BinOp —

```
class BinOp : public ParseNode {
public:
  BinOp( ParseNode *left, ParseNode *right ) : arg1(left), arg2(right)
```

```
  ~BinOp( )
protected:
  ParseNode *arg1, *arg2;
  ParseType result1, result2;
};
```

———-

9.24.7 class Concat

— class Concat —

```
class Concat : public BinOp {
public:
  Concat( ParseNode *left, ParseNode *right ) : BinOp( left, right )
  ParseType eval( );
  ParseNodeType whatAmI( ) { return CONCAT_NODE; }
};
```

———-

9.24.8 class PowerOrConjugate

— class PowerOrConjugate —

```
class PowerOrConjugate : public BinOp {
public:
  PowerOrConjugate( ParseNode *left, ParseNode *right) : BinOp( left, right)
  ParseType eval( );
  ParseNodeType whatAmI( )
};
```

———-

9.24.9 class Commutator

— class Commutator —

```
class Commutator : public BinOp {
public:
  Commutator( ParseNode *left, ParseNode *right ) : BinOp( left, right )
  ParseType eval( );
  ParseNodeType whatAmI( )
};
```

———-

10 The Genetic classes

10.1 Genetic/include/ACConfig.h

— ACConfig.h —

```
#include "global.h"
#include <values.h>
```

———-

10.1.1 class ACConfig

— class ACConfig —

```
class ACConfig {
public:
  ACConfig( int pop = 50, int gen = MAXINT, int fit_scale = 1,
      int cross_prob = 70, int mut_prob = 85, int elit = true,
      int penalty = 0, int cross_type = 1,
      int mut_app_prob = 30, int mut_insert_prob = 20,
      int mut_delete_prob = 20,
      int mut_change_prob = 10,
      int mut_permute_prob = 20,
      int use_sum_fit = 1,
      int num_min = 1000,
      int us_w = 1
      );
  int populationSize() const
  int numOfGenerations() const
  bool haveFitnessScaling() const
  bool useSumFitness() const
  double chanceOfCrossover( ) const
  double chanceOfMutation( ) const
  int numOfElitistSelection( ) const
  int penaltyRate() const
  int crossoverType() const
  double chanceOfMutationAppend( ) const
  double chanceOfMutationInsert( ) const
  double chanceOfMutationDelete( ) const
  double chanceOfMutationChange( ) const
  double chanceOfMutationPermute() const
  int numMinmizeAfterOf() const
  bool useWhitehead() const
  friend ostream& operator << ( ostream& ostr, const ACConfig& C )
  friend istream& operator >> ( istream& istr, ACConfig& C )
private:
  void readFrom( istream& istr );
```

```
  void printOn( ostream& ostr ) const;
  void setVariable( const char*, int );
  int population;
  int generations;
  int fitness_scale;
  int crossover_prob;
  int mutation_prob;
  int elittest;
  int penalty_rate;
  int crossover_type;
  int mutation_app_prob;
  int mutation_insert_prob;
  int mutation_delete_prob;
  int mutation_change_prob;
  int mutation_permute_prob;
  int use_sum_fitness;
  int num_min_after;
  int use_whitehead;
};
```

———-

10.2 Genetic/include/ACGA.h

— **ACGA.h** —

```
#include "RandomNumbers.h"
#include "FPGroup.h"
#include "FreeGroup.h"
#include "ACConfig.h"
#include "Associations.h"
```

———-

10.2.1 class ACGA

— **class ACGA** —

```
class ACGA {
public:
  ACGA( const FPGroup& G, const ACConfig& config );
  bool transform( ostream& out, ostream& best_out );
  int fitness(  Word& u, ListOf<Word>& conj );
  Word randomWord( );
  void print( Word& u, const ListOf<Word>& conj, ostream& out ) ;
private:
```

```
  Word mutate( const Word& u );
  int randomGen( );
  Word crossover1( const Word& w1,const Word& w2 );
  Word crossover2( const Word& w1,const Word& w2 );
  int genToInsertAfter( Generator g );
  int genToInsertBefore( Generator g );
  int genToInsert( Generator g1, Generator g2 );
  Word randomPermutation();
  Word randomWordInGroup();
  void applyWhitehead( ostream& out);
  FPGroup theGroup;
  ACConfig cfg;
  UniformRandom r;
  int numberOfRelators;
  int numberOfGens;
  int numberOfTrans;
  VectorOf<Word> transformations;
  VectorOf<Word> relatorsVector;
  VectorOf<Word> transferImage;
  VectorOf<Word> currentBest;
  VectorOf<Chars> trNames;
  AssociationsOf<int,int> reduces;
  int g;
  Word finalResult;
  int numberOfGenTransf;
  int numberOfRelTransf;
  bool useWhitehead;
};
```

———-

10.3 Genetic/include/Config.h

— Config.h —

```
#include "global.h"
```

———-

10.3.1 class GHNConfig

— class GHNConfig —

```
class GHNConfig {
public:
  GHNConfig( int pop = 50, int gen = -1, int cross = 100,
```

```
      int mut = 100, int elit = 1, bool fit = true );
  int populationSize( ) const
  int numOfGenerations( ) const
  double chanceOfCrossover( ) const
  double chanceOfMutation( ) const
  bool haveElitistSelection( ) const
  bool haveStrongElitistSelection( ) const
  bool haveFitnessScaling( ) const
  friend ostream& operator << ( ostream& ostr, const GHNConfig& C )
  friend istream& operator >> ( istream& istr, GHNConfig& C )
private:
  void readFrom( istream& istr );
  void printOn( ostream& ostr ) const;
  void setVariable( const char*, int );
  int population;
  int generations;
  int crossover;
  int mutation;
  int elitistSelection;
  bool bFitnessScaling;
};
```

———-

10.4 Genetic/include/GACPforORGSolver.h

— **GACPforORGSolver.h** —

```
#include "Word.h"
#include "OneRelatorGroup.h"
#include "Associations.h"
#include "RandomNumbers.h"
```

———-

10.4.1 class GACPforORGSolverChromosome

— **class GACPforORGSolverChromosome** —

```
class GACPforORGSolverChromosome {
 public:
  GACPforORGSolverChromosome( const Word& c, bool d ) : deg(d) , con(c)
  bool getDeg( ) const
 protected:
  Word con;
  bool deg;
```

```
  friend class GACPforORGSolverGene;
};
```

———-

10.4.2 class GACPforORGSolverGene

— class GACPforORGSolverGene —

```
class GACPforORGSolverGene {
 public:
  GACPforORGSolverGene( OneRelatorGroup& group , const Word& w1 , const Word& w2 );
  GACPforORGSolverGene( const GACPforORGSolverGene& g );
  GACPforORGSolverGene& operator = ( const class GACPforORGSolverGene& g );
  ~GACPforORGSolverGene( );
  double fitness( );
  bool noConj( ) const
  Word getWord1( ) const
  Word getWord2( ) const
  bool getHasShorterWords( )
  bool getHasConjecture( )   const
  Word getConjectureWord( )  const
  static Word randomWord( int gens , int wLen );
  static double proximity( const Word& W1 , const Word& W2 , int* c1=0 , int* c2=0 );
  GACPforORGSolverChromosome* randomChromosome( bool deg ) const;
public:
  void check( );
  bool mutation( );
  bool permutation( );
  friend bool crossover( GACPforORGSolverGene& g1 , GACPforORGSolverGene& g2 );
private:
  Word theWord1;
  Word theWord2;
  OneRelatorGroup& theGroup;
  static UniformRandom r;
  double fit;
  int nChr;
  void resize( unsigned pos , unsigned newsize );
  GACPforORGSolverChromosome*** chromosomes;
  unsigned* lengthes;
  unsigned* sizes;
  static unsigned jumpSize;
  void clear( );
  void delChr( unsigned p1, unsigned p2 );
  void addChr( GACPforORGSolverChromosome* chr , unsigned p1, unsigned p2 );
  enum _FindPos{ ALL=0, POSITIVE=1, NEGATIVE=2};
  void findPos( _FindPos tp, unsigned num , unsigned& p1 , unsigned& p2) const;
 private:
```

```
  bool hasConjecture;
  Word conjectureWord;
  bool hasShorterWords;
  int exp;
  int curExp;
  int getExp( );
  static const int ANY;
  static const int NOONE;
};
```

———-

10.4.3 class GAConjProblemForORGroupSolver

— class GAConjProblemForORGroupSolver —

```
class GAConjProblemForORGroupSolver {
 public:
  GAConjProblemForORGroupSolver( const OneRelatorGroup& group ,
   const Word& W1 , const Word& W2 , bool createFile = true , bool cp = true );
  ~GAConjProblemForORGroupSolver( );
  GAConjProblemForORGroupSolver operator =
    ( const GAConjProblemForORGroupSolver& solver ) ;
 protected:
  GAConjProblemForORGroupSolver( const OneRelatorGroup& group ,
     const Word& W1 , const Word& W2 , File* f );
 public:
  bool isConj( );
  int getNumberOfIterations( ) const
  Chars getFileName( ) const
 public:
  static int rnd1( int max );
  static int roulette( double d1 , double d2 );
  static int roulette( int num , double* d);
  static Word greedyReduce( const OneRelatorGroup& group , const Word& word );
  static bool oneGreedyReduce( const OneRelatorGroup& group , Word& w );
  static void insert( Word& dst , const Word& src , int pos);
 protected:
  void checkImprovementTime( );
  int reproduction( );
  bool tournament( GACPforORGSolverGene& gene );
  int selectGene( ) const;
  void toStart( const Word& W1 , const Word& W2 );
 protected:
  static double prob[2][3];
  bool conjProblem;
  int lastImprovement;
  int fitnessRate;
```

```
  int theIter1, theIter2;
  double bestFit;
  File* file;
  bool deleteFile;
  AssociationsOf< Word , int > checkedWords;
  int numGenes;
  GACPforORGSolverGene** genes;
  GACPforORGSolverGene *newGene[2];
  OneRelatorGroup theGroup;
  Word theWord1;
  Word theWord2;
  static const int NOCONJ;
};
```

———-

10.4.4 class GAConjProblemForORGroupConjecture

— class GAConjProblemForORGroupConjecture —

```
class GAConjProblemForORGroupConjecture : private GAConjProblemForORGroupSolver
{
 private:
  GAConjProblemForORGroupConjecture( const OneRelatorGroup& group ,
                                     const Word& W , File* f ) :
  GAConjProblemForORGroupSolver( group , Word( ) , W , f )
  bool isConj( int maxIter , int& doneIter );
  friend class GAConjProblemForORGroupSolver;
};
```

———-

10.5 Genetic/include/GAEquationSolver.h

— GAEquationSolver.h —

```
#include "RandomNumbers.h"
#include "FreeGroup.h"
#include "Config.h"
```

———-

10.5.1 class GAEquationSolver

— class GAEquationSolver —

```
class GAEquationSolver {
public:
  GAEquationSolver( const FreeGroup& G, int NumOfVars,
        const GHNConfig& config );
  Map getSolution( const Word& u, ostream* out = NULL, int& g = dummy_g );
private:
  Word mutate( const Word& u );
  int randomGen( );
  Word randomWord( );
  Word crossover( const Word& u, const Word& v );
  FreeGroup theGroup;
  GHNConfig cfg;
  int numOfVars, numOfGens, numOfConsts;
  bool keepDetails;
  UniformRandom r;
  const int maxWordLen;
  static int dummy_g;
};
```

———-

10.5.2 class GraphicEquationSolver

— class GraphicEquationSolver —

```
class GraphicEquationSolver {
public:
  GraphicEquationSolver( FreeGroup F, VectorOf<Chars> vNames,
       Word eq, const GHNConfig& config, Chars fn = "" );
  Map getSolution( const Word& u, ostream* out = NULL );
private:
  Word mutate( const Word& u );
  char mutateChar( char u );
  int randomGen( );
  Word randomWord( );
  Word crossover( const Word& u, const Word& v );
  char crossoverChar( char u, char v );
  int fitness( Word u, Word v ) const;
  FreeGroup theGroup;
  GHNConfig cfg;
  VectorOf<Chars> varNames;
  int numOfVars, numOfConsts;
  bool keepDetails;
  UniformRandom r;
  int maxWordLen;
  Word equation;
  Chars popFile;
};
```

———-

10.6 Genetic/include/GAIsPartOfBasis.h

— GAIsPartOfBasis.h —

```
#include "RandomNumbers.h"
#include "FPGroup.h"
#include "FreeGroup.h"
#include "ACConfig.h"
#include "Associations.h"
```

———-

10.6.1 class GAIsPartOfBasis

— class GAIsPartOfBasis —

```
class GAIsPartOfBasis {
public:
  GAIsPartOfBasis( const FreeGroup& G, const ACConfig& config,
       const VectorOf<Word>& v);
  GAIsPartOfBasis( const FreeGroup& G, const ACConfig& config,
       const VectorOf<Word>&,const VectorOf<Word>&);
  GAIsPartOfBasis( const FreeGroup& G, const ACConfig& config,
       const Word& w);
  Trichotomy transform( ostream& , int );
  bool isPartOfBasis(ostream& out1, ostream& out2);
  bool isPartOfBasis(ostream& out1, const Chars& out2_f_name);
  VectorOf<Word> getAutomorphism() const
  const VectorOf<Word>& getCurrentBestWords()const
  const VectorOf<Word>& getFixedWords() const
  int nOfGenerations() const
  int fitness( const Word& u,ostream& out = cout, bool print = false );
  Word randomWord( );
private:
  void initialize();
  void printWords(ostream& o, const VectorOf<Word>& v) const;
  Word mutate( const Word& u );
  int randomGen( );
  Word crossover1( const Word& w1,const Word& w2 );
  int genToInsertAfter( Generator g );
  int genToInsertBefore( Generator g );
  int genToInsert( Generator g1, Generator g2 );
  VectorOf<Word> computeAuto(const Word&);
  int computeFitness( const VectorOf<Word>& v ) const;
  int computeFitness1( const VectorOf<Word>& v ) const;
```

```
  int hamingDistance(const Word& w1,const Word& w2) const;
  int tupleHamingDistance(const VectorOf<Word>&,const VectorOf<Word>&)const;
  void permuteVector(VectorOf<Word>& v1, int move_to) const;
  FPGroup theGroup;
  ACConfig cfg;
  VectorOf<Word> theWords;
  VectorOf<Word> tmpWords;
  VectorOf<Word> bestWords;
  VectorOf<Word> fixedWords;
  Chars best_out_name;
  bool print_best_each_time;
  friend int main(int, char**);
  UniformRandom r;
  int numberOfGens;
  int numberOfTrans;
  int threshold;
  VectorOf<Word> transformations;
  VectorOf<Word> transferImage;
  VectorOf<Word> theAuto;
  AssociationsOf<int,int> reduces;
  int g;
  int sav_first_max;
  Word finalResult;
  Word bestTransformation;
  int total_tr;
  int swithc_tr;
};
```

———-

10.7 Genetic/include/GASubgroup.h

— GASubgroup.h —

```
#include "SubgroupGraph.h"
#include "RandomNumbers.h"
```

———-

10.7.1 class GASubgroup

— class GASubgroup —

```
class GASubgroup {
public:
  GASubgroup( );
```

```
  GASubgroup( const SetOf<Word>& generators ) : gens( generators )
  SetOf<Word> generators( ) const
  int fitness( const GASubgroup& S ) const;
  GASubgroup mutate( ) const;
  GASubgroup crossover( const GASubgroup& S ) const;
  GASubgroup randomSubgroup( ) const;
  friend ostream& operator << ( ostream& ostr, const GASubgroup& S )
private:
  Word randomWord( ) const;
  int randomGen( ) const;
  SetOf<Word> gens;
  static int maxCard;
  static int maxWordLen;
  static int numOfGens;
  static UniformRandom r;
};
```

———-

10.8 Genetic/include/GAWord.h

— **GAWord.h** —

```
#include "RandomNumbers.h"
#include "Map.h"
#include "FreeGroup.h"
```

———-

10.8.1 class GAWord

— **class GAWord** —

```
class GAWord {
public:
  GAWord( ) : numOfGens( 0 )
  GAWord( int NumOfGens, const Word& w ) : numOfGens(NumOfGens), theWord( w )
  Word getWord( ) const
  GAWord mutate( ) const;
  GAWord crossover( const GAWord& w ) const;
  GAWord randomWord( ) const;
  friend ostream& operator << ( ostream& ostr, const GAWord& w )
private:
  int randomGen( ) const;
  int numOfGens;
  Word theWord;
```

```
  static int maxWordLen;
  static UniformRandom r;
};
```

———-

10.9 Genetic/include/GAWP.h

— GAWP.h —

```
#include "RandomNumbers.h"
#include "FPGroup.h"
#include "FreeGroup.h"
#include "Config.h"
#include "SubgroupGraph.h"
```

———-

10.9.1 class GAWP

— class GAWP —

```
class GAWP {
public:
  GAWP( const FPGroup& G, const GHNConfig& config )
    : theGroup( G ), cfg( config ), I(G.getRelators())
  Trichotomy isTrivial( const Word& u, ostream& out );
private:
  int fitness( const Word& u ) const;
  Word mutate( const Word& u );
  FPGroup theGroup;
  GHNConfig cfg;
  Word w;
  int wLen;
  SetIterator<Word> I;
  UniformRandom r;
};
```

———-

10.9.2 class GAWP2

— class GAWP2 —

```
class GAWP2 {
public:
  GAWP2( const FPGroup& G, const GHNConfig& config );
  Trichotomy isTrivial( const Word& u, ostream* out = NULL );
private:
  int fitness( const Word& u ) const;
  Word mutate( const Word& u );
  FPGroup theGroup;
  GHNConfig cfg;
  SetOf<Word> relators;
  Word w;
  int wLen;
  bool keepDetails;
  SetIterator<Word> I;
  UniformRandom r;
};
```

———-

10.10 Genetic/include/Roulette.h

— Roulette.h —

```
#include "RandomNumbers.h"
```

———-

10.10.1 class RW

— class RW —

```
class RW_Except { };

template < class FType > class RouletteWheel {
public:
  RouletteWheel (size_t sz, FType * weights = NULL,int seed = 0);
  RouletteWheel (const RouletteWheel & rw);
  void operator = (const RouletteWheel & rw);
  ~RouletteWheel();
  FType Change (size_t i,FType weight);
  size_t GetSize()
  FType GetWeight(size_t i);
  size_t GetIndex();
protected:
  size_t  N;
  FType * W;
  FType T;
```

```
  UniformRandom G;
private:
  FType AbsVal(FType f)
  void Copy(const RouletteWheel & rw);
};

template < class FType>
    RouletteWheel<FType>::RouletteWheel(size_t  sz,FType * weights,int seed)

template < class FType>
    void RouletteWheel<FType>::Copy(const RouletteWheel & rw)

template < class FType >
    inline RouletteWheel<FType>::RouletteWheel(const RouletteWheel & rw)

template < class FType >
    inline void RouletteWheel<FType>::operator =(const RouletteWheel & rw)

template < class FType >
    RouletteWheel<FType>::~RouletteWheel()

template < class FType >
    FType RouletteWheel<FType>::Change(size_t i,FType weight)

template < class FType >
    inline FType RouletteWheel<FType>::GetWeight(size_t i)

template < class FType >
    size_t RouletteWheel<FType>::GetIndex()
```

———-

10.11 Genetic/include/TwoCommSolver.h

— TwoCommSolver.h —

```
#include "RandomNumbers.h"
#include "FreeGroup.h"
#include "Config.h"
```

———-

10.11.1 class TwoCommSolver

— class TwoCommSolver —

```
class TwoCommSolver {
public:
  TwoCommSolver( FreeGroup F );
  ~TwoCommSolver( );
  bool isProductOfTwoComms( Word r, Word& x1, Word& x2,
          Word& y1, Word& y2,
          ostream* out = NULL );
private:
  Map getSolution( const Word& u, int& eqInd, int& conjInd,
       ostream* out = NULL );
  Word getImage( Map M, Map solution, int ind, int num );
  Word mutate( const Word& u );
  int randomGen( );
  Word randomWord( );
  Word crossover( const Word& u, const Word& v );
  int fitness( Word u, Word v, int& ) const;
  FreeGroup theGroup;
  GHNConfig cfg;
  VectorOf<Chars> *varNames;
  int numOfVars, numOfConsts, numOfEq;
  bool keepDetails;
  UniformRandom r;
  int maxWordLen;
  Word *equation;
  Chars popFile;
};
```

———-

11 The Group classes

11.1 Group/include/AbelianEquations.h

— AbelianEquations.h —

```
#include "AbelianGroup.h"
#include "FPGroup.h"
#include "FreeGroup.h"
#include "File.h"
#include "Vector.h"
#include "Word.h"
#include <iostream.h>
```

———-

11.1.1 class AbelianEquationsSolver

— class AbelianEquationsSolver —

```
class AbelianEquationsSolver {
public:
  AbelianEquationsSolver( const AbelianGroup& a ,
        const VectorOf<Word>& v , int numOfVar );
  AbelianEquationsSolver( ) : rawA( FPGroup() ), A( FPGroup() )
  void findSolutions( File& file , bool out = true );
  VectorOf<Word> getBasicSolutions() const
  VectorOf< VectorOf< VectorOf<int> > > getTorsionPart() const
  VectorOf< VectorOf< VectorOf<int> > > getParametrizedPart() const
  int getSystemRank() const
  int getNumberOfParams() const
  bool solIsAllGroup( ) const;
  bool haveSolutions() const;
private:
  bool root( Word& , int ) const;
  void makeSystem();
  void printRawSystem( File& ) const;
  void printSystem( File& ) const;
  AbelianGroup rawA;
  AbelianGroup A;
  VectorOf<Word> rawSystem;
  VectorOf<Word> system;
  VectorOf<Word> b;
  int numberOfVariables;
  int sysRank;
  int haveSol;
  VectorOf<Word> x;
  VectorOf< VectorOf< VectorOf<int> > > torsion;
  VectorOf< VectorOf< VectorOf<int> > > params;
};
```

———-

11.2 Group/include/AbelianGroup.h

— AbelianGroup.h —

```
#include "FPGroup.h"
#include "AbelianGroupRep.h"
```

———-

11.2.1 class AbelianGroup

— class AbelianGroup —

```
class AbelianGroup : public ObjectOf<AbelianGroupRep> {
public:
  AbelianGroup( const FPGroup& G, bool makeFile = false )
    : ObjectOf<AbelianGroupRep>( new AbelianGroupRep(G, makeFile) )
  void computeCyclicDecomposition( )
  void findPrimaryBasis()
  bool haveCyclicDecomposition( ) const
  bool havePrimaryDecomposition( ) const
  Chars getFileName( ) const
  Chars getFileNameOfPDGens( ) const
  const FPGroup getFPGroup( ) const
  SetOf<Word> getAllRelators( ) const
  AbelianWord oldInAbelianForm( const Word& w ) const
  int rankOfFreeAbelianFactor( ) const
  VectorOf<Integer> invariants( ) const
  VectorOf<AbelianWord> oldToNewGens() const
  VectorOf<AbelianWord> newToOldGens() const
  AbelianGroup getCanonicalSmithPresentation( ) const
  Integer order() const
  bool isTrivial() const
  bool isFinite() const
  bool isInfinite() const
  bool isFree() const
  bool isomorphicTo( const AbelianGroup& G) const
  AbelianGroup computeIntegralHomology( int n ) const
  Integer orderOfTheTorsionSubgroup( ) const
  AbelianSGPresentation
    makeSubgroupPresentation(const VectorOf<Word>& vG) const;
  VectorOf<Word> findSubgroupIsolator(const VectorOf<Word>& vG) const
  VectorOf<Word> findVirtFreeComplementOfSG(const VectorOf<Word>& vG) const
  VectorOf<Word>
    joinSubgroups(const VectorOf<Word>& vG1,const VectorOf<Word>& vG2) const
  VectorOf<Word> findSubgIntersection( const VectorOf<Word>& vG1 ,
               const VectorOf<Word>& vG2 , File& file ) const
  bool isPureCyclSubgroup(const Word& w) const
  bool areEqual(const Word& u, const Word& v) const
  bool isTrivial( const Word& w ) const
  Integer orderOfElt( const Word& w ) const
  AbelianWord newToOldGens( const AbelianWord& w ) const
  AbelianWord oldToNewGens( const AbelianWord& w ) const
  AbelianWord findEltPrimeForm(const Word& w) const
  AbelianWord pBlockOfElt( const AbelianWord& w,Integer p )const
  AbelianWord pBlockOfElt( const Word& w,Integer p )const
  Integer pHeightOfElt(const Word& w, const Integer& p) const
  Integer
```

```
    powerOfEltInSubgroup(const Word& w,const VectorOf<Word>& sGroup) const
  bool isEltProperPower(const Word& w) const
  void abelianMaximalRoot(const Word& w, Word& maxRoot, Integer& maxExp) const
  AbelianWord primeFormInOldGens(const AbelianWord& w) const
  int isPowerOfSecond(const Word& word1, const Word& word2) const
  Bool isEpimorphism(const VectorOf<Word>& V) const
  int orderOfAuto(const VectorOf<Word>& V) const
  VectorOf<Word> inverseAuto(const VectorOf<Word>& V) const
  VectorOf<Word> fixedPointsOfAuto( const VectorOf<Word>& v ) const
  friend ostream& operator << ( ostream& ostr, const AbelianGroup& G )
  void printWordInNewGens( ostream& ostr, const AbelianWord& w) const
  void printInPrimaryForm(ostream& ostr, const AbelianWord& aw) const
  void printPrimaryDec( ostream& ostr) const
  friend ostream& operator < ( ostream& ostr, const AbelianGroup& G )
  friend istream& operator > ( istream& istr, AbelianGroup& G )
protected:
  friend AbelianSGPresentationRep
    AbelianGroupRep::makeSubgroupPresentation(const VectorOf<Word>& vG)const;
  AbelianGroup( AbelianGroupRep* newrep )
    : ObjectOf<AbelianGroupRep>(newrep)
};
```

———-

11.3 Group/include/AbelianGroupRep.h

— **AbelianGroupRep.h** —

```
#include "AbelianWord.h"
#include "FPGroup.h"
#include "Matrix.h"
#include "PrimeNumbers.h"
#include "File.h"
```

———-

11.3.1 class AbelianGroupRep

— **class AbelianGroupRep** —

```
class AbelianGroupRep : public PureRep {
public:
  AbelianGroupRep( const FPGroup& G, bool makeFile = false );
  void computeCyclicDecomposition( );
  void findPrimaryBasis();
  bool haveCyclicDecomposition( ) const
```

```
bool havePrimaryDecomposition( ) const
Chars AbelianGroupRep::getFileName( ) const;
Chars getFileNameOfPDGens( ) const;
const FPGroup getFPGroup( ) const
SetOf<Word> getAllRelators( ) const;
AbelianWord oldInAbelianForm( const Word& w ) const
int rankOfFreeAbelianFactor( ) const;
VectorOf<Integer> invariants( ) const;
VectorOf<AbelianWord> oldToNewGens() const;
VectorOf<AbelianWord> newToOldGens() const;
AbelianGroupRep getCanonicalSmithPresentation( ) const;
Integer order() const;
bool isTrivial() const;
bool isFree() const;
bool isomorphicTo( const AbelianGroupRep& G) const;
AbelianGroupRep computeIntegralHomology( int n ) const;
Integer orderOfTheTorsionSubgroup() const;
AbelianSGPresentationRep
  makeSubgroupPresentation(const VectorOf<Word>& vG) const;
VectorOf<Word> findSubgroupIsolator(const VectorOf<Word>& vG) const;
VectorOf<Word> findVirtFreeComplementOfSG(const VectorOf<Word>& vG) const;
VectorOf<Word>
  joinSubgroups(const VectorOf<Word>& vG1,const VectorOf<Word>& vG2) const;
VectorOf<Word> findSubgIntersection( const VectorOf<Word>& vG1 ,
             const VectorOf<Word>& vG2 , File& file ) const;
bool isPureCyclSubgroup(const Word& w) const;
bool areEqual( const Word&, const Word& ) const;
bool isTrivial( const Word& ) const;
Integer orderOfElt( const Word& ) const;
AbelianWord newToOldGens( const AbelianWord& ) const;
AbelianWord oldToNewGens( const AbelianWord& ) const;
AbelianWord findEltPrimeForm(const Word& w) const;
AbelianWord pBlockOfElt( const AbelianWord& w,Integer p )const;
Integer pHeightOfElt(const Word& w, const Integer& p) const;
Integer
  powerOfEltInSubgroup(const Word& w,const VectorOf<Word>& sGroup) const;
bool isEltProperPower(const Word& w) const;
void abelianMaximalRoot(const Word& w, Word& maxRoot, Integer& maxExp) const;
AbelianWord primeFormInOldGens(const AbelianWord& w) const;
int isPowerOfSecond(const Word& word1, const Word& word2) const;
Bool isEpimorphism(const VectorOf<Word>& V) const;
int orderOfAuto(const VectorOf<Word>& V) const;
VectorOf<Word> inverseAuto(const VectorOf<Word>& V) const;
VectorOf<Word> fixedPointsOfAuto( const VectorOf<Word>& v ) const;
void printOn( ostream& ) const;
void printWordInNewGens( ostream&, const AbelianWord& ) const;
void printInPrimaryForm(ostream& ostr, const AbelianWord& aw) const;
void printPrimaryDec( ostream& ostr) const;
virtual void write( ostream& ostr ) const;
virtual void read( istream& istr );
```

```
  AbelianGroupRep* clone( ) const
protected:
  Chars theFileName;
  Chars theFileNameOfPD;
  int numOfGensInTorsionPartOfPD() const
private:
  const FPGroup theGroup;
  bool bMakeFile;
  bool bHaveCyclicDecomposition;
  int numOfNewGens;
  int rankOfFreePart;
  VectorOf<Integer> theInvariants;
  VectorOf<AbelianWord> theNewToOldGens;
  VectorOf<AbelianWord> theOldToNewGens;
  bool primeBasisFound;
  int invariantToNewGens( int inv, Integer orderOfCyclic,
         int stPos,Integer power = 1);
  DArray<Integer> primeBasicMatrix;
  Integer** matrix;
  int height;
  int width;
  Integer** newToOldGensMatrix;
  Integer** oldToNewGensMatrix;
  void fillTransformationVectors( );
  void addColumn( int i, int j, Integer k );
  void swapColumns( int i, int j );
  void swapGenColumns( int i, int j );
  void canoniseInvariants( int i, int j );
  void swapInvariants( int i, int j );
  void makeTransformationMatrices( );
  void makeMainMatrix( );
  void destroyMatrices( );
  virtual void makeFile( );
  virtual void makeFileOfPDGens( );
  void sortVector(DArray<Integer>& vc,int colSort,int start, int finish);
  void sortPrimeDecom(DArray<Integer>& m);
  void minimizeWordInNewGens( AbelianWord& w ) const;
  bool isAllZero(int from, int to,MatrixRow<Integer>& vc) const;
  Bool matrixMult(const Matrix<Integer>& m,bool haveTorsion) const;
};
```

———-

11.4 Group/include/AbelianInfinitenessProblem.h

— AbelianInfinitenessProblem.h —

```
#include "FPGroup.h"
```

```
#include "GaussTransformation.h"
```

———-

11.4.1 class AbelianInfinitenessProblem

— class AbelianInfinitenessProblem —

```
class AbelianInfinitenessProblem {
public:
  AbelianInfinitenessProblem( const FPGroup& G) : theGroup( G ),
    bStart( false ), bDone( false )
  ~AbelianInfinitenessProblem( )
  void startComputation( );
  bool continueComputation( )
  bool done( ) const
  bool isInfinite( )
private:
  bool bDone;
  bool bStart;
  bool itIsInfinite;
  Matrix<Rational> *matrix;
  int width;
  int height;
  const FPGroup theGroup;
  GaussTransformation<Rational> *GT;
  AbelianInfinitenessProblem( const AbelianInfinitenessProblem& );
  AbelianInfinitenessProblem& operator = ( const AbelianInfinitenessProblem& );
  void finishComputation( );
};
```

———-

11.5 Group/include/AbelianSGPresentation.h

— AbelianSGPresentation.h —

```
#include "AbelianGroup.h"
#include "AbelianGroupRep.h"
#include "Map.h"
```

———-

11.5.1 struct AbelianSGPresentationRep

— struct AbelianSGPresentationRep —

```
struct AbelianSGPresentationRep : AbelianGroupRep {
  AbelianSGPresentationRep(const AbelianGroup& parent,
         const VectorOf<Word>& sgGens, const FPGroup& thePresentation,
         const DArray<Integer>& sgPGens, const DArray<Integer>& sgPGensInv,
         const VectorOf<Integer>& invariants, bool makeF )
    : AbelianGroupRep(thePresentation,makeF),
       theParent( parent ), theSGGens( sgGens ), theSGPGens( sgPGens ),
       theSGPGensInv( sgPGensInv ), theInvariants( invariants )
  AbelianSGPresentationRep* clone( ) const
  Word fromSGPGensToSGGens(const Word& w)const;
  bool fromSGGensToSGPGens(const Word& w,Word& wInNew)const;
  inline void write( ostream& ostr ) const
  inline void read( istream& istr )
private:
  VectorOf<Word>  theSGGens;
  DArray<Integer> theSGPGens;
  DArray<Integer> theSGPGensInv;
  VectorOf<Integer> theInvariants;
  AbelianGroup    theParent;
  void makeFile( );
  void makeFileOfPDGens( );
};
```

———-

11.5.2 class AbelianSGPresentation

— class AbelianSGPresentation —

```
class AbelianSGPresentation :
  public DerivedObjectOf<AbelianGroup,AbelianSGPresentationRep> {
public:
  AbelianSGPresentation() :
    DerivedObjectOf<AbelianGroup,AbelianSGPresentationRep>
      (new AbelianSGPresentationRep(AbelianGroup(FPGroup()),
       VectorOf<Word>(),FPGroup(),DArray<Integer>(),DArray<Integer>(),
       VectorOf<Integer>(),false))
  Word fromSGPGensToSGGens(const Word& w)const
  bool fromSGGensToSGPGens(const Word& w,Word& wInNew)const
protected:
private:
  friend AbelianSGPresentation
    AbelianGroup::makeSubgroupPresentation(const VectorOf<Word>& vG) const;
  AbelianSGPresentation(const AbelianGroup& parent,
```

```
      const VectorOf<Word>& sgGens, const FPGroup& thePresentation,
      const DArray<Integer>& sgPGens, const DArray<Integer>& sgPGensInv,
      const VectorOf<Integer>& invariants, bool makeF = false) :
    DerivedObjectOf<AbelianGroup,AbelianSGPresentationRep>
      (new AbelianSGPresentationRep(parent,sgGens,thePresentation,
                                    sgPGens,sgPGensInv,invariants,makeF))
  AbelianSGPresentation( AbelianSGPresentationRep* newrep ) :
    DerivedObjectOf<AbelianGroup,AbelianSGPresentationRep>(newrep)
};
```

———-

11.6 Group/include/EqSystemParser.h

— **EqSystemParser.h** —

```
#include "EquationParser.h"
#include "PresentationParser.h"
#include "conversions.h"
```

———-

11.6.1 class EqSystemParser

— **class EqSystemParser** —

```
class EqSystemParser : protected PresentationParser {
public:
  EqSystemParser(istream &istr) : PresentationParser(istr)
  VectorOf<Word> parseEqSystem( const VectorOf<Chars>& names,
        VectorOf<Chars>& new_names, Chars& errMesg );
  Word parseEquation( const VectorOf<Chars>& names,
          VectorOf<Chars>& new_names, Chars& errMesg );
  virtual void getToken( );
  virtual Word parseRelator( const VectorOf<Chars>&, Chars& );
};
```

———-

11.7 Group/include/EquationParser.h

— **EquationParser.h** —

```
#include "PresentationParser.h"
```

———-

11.7.1 class EquationParser

— class EquationParser —

```
class EquationParser : protected PresentationParser
{
public:
  EquationParser(istream &istr) : PresentationParser(istr)
  virtual Word parseRelator( const VectorOf<Chars>&, Chars& );
  virtual void getToken( );
  Word parseEquation( const VectorOf<Chars>& names,
               VectorOf<Chars>& new_names, Chars& errMesg );
};
```

———-

11.8 Group/include/FGGroup.h

— FGGroup.h —

```
#include "DerivedObjectOf.h"
#include "Group.h"
#include "FGGroupRep.h"
```

———-

11.8.1 class FGGroup

— class FGGroup —

```
class FGGroup : public DerivedObjectOf<Group,FGGroupRep> {
public:
  FGGroup( const Group& g ) : DerivedObjectOf<Group,FGGroupRep>( g )
  static Type type( )
  int numberOfGenerators( ) const
  Chars nameOfGenerator(int i) const;
  Chars nameOfGenerator(Generator g) const;
  VectorOf<Chars> namesOfGenerators( ) const
  Elt eval( const Word& w ) const
  Trichotomy wordProblem( const Word& w ) const
  Trichotomy conjugacyProblem( const Word& u, const Word& v ) const
  void closeUnderCyclicPermutations(SetOf<Word>& S) const;
  Word readWord(istream& istr, Chars& errMesg) const;
  SetOf<Word> readSetOfWords(istream& istr, Chars& errMesg) const;
  VectorOf<Word> readVectorOfWords(istream& istr, Chars& errMesg) const;
  void printGenerator( ostream& ostr, const Generator& g ) const
```

```
  void printGenerators( ostream& ostr ) const
  void printWord( ostream& ostr, const Word& w ) const
  void printSetOfWords( ostream& ostr, const SetOf<Word>& S ) const
  void printVectorOfWords( ostream& ostr, const VectorOf<Word>& V,
      char* leftBracket = "{", char* rightBracket = "}" ) const
  Subgroup randomSubgroup( ) const;
protected:
  FGGroup( FGGroupRep* newrep ) : DerivedObjectOf<Group,FGGroupRep>(newrep)
};
```

———-

11.9 Group/include/FGGroupRep.h

— **FGGroupRep.h** —

```
#include "Chars.h"
#include "Vector.h"
#include "Word.h"
#include "GroupRep.h"
```

———-

11.9.1 struct FGGroupRep

— **struct FGGroupRep** —

```
struct FGGroupRep : GroupRep {
public:
  FGGroupRep( int ngens ) :
    theNumberOfGenerators(ngens), theNamesOfGenerators(0)
  FGGroupRep( const VectorOf<Chars>& gennames ) :
    theNumberOfGenerators(gennames.length()), theNamesOfGenerators(gennames)
private:
  FGGroupRep& operator = ( const FGGroupRep& );
public:
  static const Type theFGGroupType;
  static Type type( )
  Type actualType( ) const
  bool compare( const GroupRep* G ) const;
  Elt makeIdentity( ) const
  Bool isSyntacticIdentity( const Elt& e) const;
  Elt firstElt( ) const
  Elt nextElt( const Elt& e ) const;
  Elt multiply( const Elt& e1, const Elt& e2 ) const;
  Elt inverseOf( const Elt& e ) const;
```

```
  virtual Elt eval( const Word& w ) const = 0;
  virtual Trichotomy wordProblem( const Word& w ) const = 0;
  virtual Trichotomy conjugacyProblem( const Word& u, const Word& v) const = 0;
  virtual void printGenerator( ostream& ostr, int n ) const;
  virtual void printGenerators( ostream& ostr ) const;
  virtual void printWord( ostream& ostr, const Word& w ) const;
  virtual void printSetOfWords( ostream& ostr, const SetOf<Word>& S ) const;
  virtual void printVectorOfWords( ostream& ostr, const VectorOf<Word>& V,
              char* leftBracket = "{", char* rightBracket = "}" ) const;
  void write( ostream& ostr ) const ;
  void read( istream& istr ) ;
  int theNumberOfGenerators;
  VectorOf<Chars> theNamesOfGenerators;
  bool consistent( ) const

};
```

———-

11.10 Group/include/FPGroup.h

— **FPGroup.h** —

```
#include "FGGroup.h"
#include "FPGroupRep.h"
#include "File.h"
```

———-

11.10.1 class FPGroup

— **class FPGroup** —

```
class FPGroup : public DerivedObjectOf<FGGroup,FPGroupRep> {
public:
  FPGroup( int ngens = 0 ) :
    DerivedObjectOf<FGGroup,FPGroupRep>( new FPGroupRep(ngens) )
  FPGroup( int ngens, const SetOf<Word>& rels ) :
    DerivedObjectOf<FGGroup,FPGroupRep>( new FPGroupRep(ngens, rels) )
  FPGroup( const VectorOf<Chars>& gennames ) :
    DerivedObjectOf<FGGroup,FPGroupRep>( new FPGroupRep(gennames) )
  FPGroup( const VectorOf<Chars>& gennames,
                  const SetOf<Word>& rels ) :
    DerivedObjectOf<FGGroup,FPGroupRep>( new FPGroupRep(gennames, rels) )
  FPGroup( const Group& G ) : DerivedObjectOf<FGGroup,FPGroupRep>(G)
  static Type type( )
```

```
  SetOf<Word> getRelators( ) const
  SetOf<Word> setRelators( const SetOf<Word>& r )
  SetOf<Word> addRelators( const SetOf<Word>& r )
  SetOf<Word> removeRelators( const SetOf<Word>& r )
  Trichotomy isFree( ) const
  Trichotomy isMetricSmallCancellationGroup( ) const
  int cancellationLambda( ) const
  Word shortenByRelators(const Word& w) const
  Chars productOfCommutators(const Word& w,File& file)
  Chars productOfSquares(const Word& w,File& file)
  void printRelators(ostream& ostr) const
private:
  void computeCancellationLambda( ) const
protected:
   FPGroup( FPGroupRep* newrep ) : DerivedObjectOf<FGGroup,FPGroupRep>(newrep)
private:
};
```

———-

11.11 Group/include/FPGroupRep.h

— **FPGroupRep.h** —

```
#include <Integer.h>
#include "FGGroupRep.h"
#include "File.h"
```

———-

11.11.1 struct FPGroupRep

— **struct FPGroupRep** —

```
struct FPGroupRep : FGGroupRep {
  FPGroupRep( int ngens )
  : FGGroupRep(ngens),
   isMetricSmallCancellation(MetricSmallCancellationLambda::infinity)
  FPGroupRep( const VectorOf<Chars>& gennames )
  : FGGroupRep( gennames ),
   isMetricSmallCancellation(MetricSmallCancellationLambda::infinity)
  FPGroupRep( int ngens, const SetOf<Word>& rels )
  : FGGroupRep(ngens),
   relators(rels)
  FPGroupRep( const VectorOf<Chars>& gennames, const SetOf<Word>& rels )
  : FGGroupRep( gennames ),
```

```
    relators(rels)
  virtual SetOf<Word>& setRelators( const SetOf<Word>& r )
  virtual SetOf<Word>& addRelators( const SetOf<Word>& r )
  virtual SetOf<Word>& removeRelators( const SetOf<Word>& r )
  PureRep* clone( ) const
  static const Type theFPGroupType;
  static Type type( )
  Type actualType( ) const
  int order( ) const;
  Trichotomy isTrivial( ) const;
  Trichotomy isFinite( ) const;
  Trichotomy isInfinite( ) const;
  Trichotomy isAbelian( ) const;
  virtual Trichotomy isFree( ) const;
  bool compare( const GroupRep* G ) const;
  void printOn(ostream&) const;
  GroupRep* readFrom(istream&, Chars&) const;
  virtual void printRelators(ostream&) const;
  virtual Trichotomy isTrivialElt( const Elt& e ) const
  Trichotomy areEqual(const Elt& e1, const Elt& e2) const
  Elt eval( const Word& w ) const
  Trichotomy wordProblem( const Word& w ) const
  Trichotomy conjugacyProblem( const Word& u, const Word& v ) const
  Word shortenByRelators(const Word& w) const;
  Chars productOfCommutators(const Word& w,File& file);
  Chars productOfSquares(const Word& w,File& file);
  Integer decideOrder(FPGroupRep* Gr) const;
  inline void write( ostream& ostr ) const
  inline void read( istream& istr )
  SetOf<Word> relators;
```

———-

11.11.2 class MetricSmallCancellationLambda

— class MetricSmallCancellationLambda —

```
class MetricSmallCancellationLambda {
    int lambda_val;
public:
  MetricSmallCancellationLambda( ) : lambda_val(dontknow)
  MetricSmallCancellationLambda( int l ) : lambda_val(l)
  friend ostream& operator < ( ostream& ostr,
                               const MetricSmallCancellationLambda& iMSC)
  friend istream& operator > ( istream& istr,
                               MetricSmallCancellationLambda& iMSC )
  Trichotomy operator()( ) const
  int lambda( ) const
```

```
  void setLambda( int l )
  void reset( )
  enum { dontknow = -1, infinity = 0 };
};
```

———-

— **eh?** —

```
  MetricSmallCancellationLambda isMetricSmallCancellation;

};
```

———-

11.12 Group/include/FreeByCyclic.h

— **FreeByCyclic.h** —

```
#include "FreeGroup.h"
#include "Map.h"
```

———-

11.12.1 class FreeByCyclic

— **class FreeByCyclic** —

```
class FreeByCyclic
{
public:
  FreeByCyclic(const FreeGroup F, const Map aut);
  FreeByCyclic& operator=( const FreeByCyclic& G );
  VectorOf<Chars> namesOfGenerators( ) const;
  SetOf<Word> getRelators( ) const;
  Word normalForm(const Word& w) const;
  friend ostream& operator<<( ostream& ostr, const FreeByCyclic& G );
  friend istream& operator>>( istream& istr,       FreeByCyclic& G );
private:
  const FreeGroup theGroup;
  const Map theAut;
  const Map  theAutInv;
  int newGenerator;
};
```

———-

11.13 Group/include/FreeGroup.h

— **FreeGroup.h** —

```
#include "FGGroup.h"
#include "FreeGroupRep.h"
#include "Map.h"
#include "File.h"
```

———-

11.13.1 class FreeGroup

— **class FreeGroup** —

```
class FreeGroup : public DerivedObjectOf<FGGroup,FreeGroupRep> {
public:
  FreeGroup( int rank = 0 ) :
    DerivedObjectOf<FGGroup,FreeGroupRep>( new FreeGroupRep(rank) )
  FreeGroup( const VectorOf<Chars>& gennames ) :
    DerivedObjectOf<FGGroup,FreeGroupRep>( new FreeGroupRep(gennames) )
  static Type type( )
  int rank( ) const
  Aut randomAut( ) const;
  Trichotomy conjugacyProblem( const Word& u, const Word& v ,Word& c) const
  Bool inCommutatorSG(const Word& w) const
  Bool isCommutator(const Word& w, Word& u1, Word& u2) const
  Chars productOfCommutators( const Word& w , File& file )
  Chars productOfSquares( const Word& w , File& file )
  Word getN_thElement( int n ) const
  int numberOfElement( const Word& w ) const
  VectorOf<Word> nielsenBasis(const VectorOf<Word>& V) const
  typedef FreeGroupRep::NielsenBasis NielsenBasis;
  NielsenBasis nielsenBasis(const VectorOf<Word>& V, bool writeToFile ) const
  Bool isAutomorphism(const VectorOf<Word>& V) const
  Bool isInnerAutomorphism(const VectorOf<Word>& V) const
  bool isIAAutomorphism(const VectorOf<Word>& V) const
  VectorOf<Word> inverseAutomorphism(const VectorOf<Word>& V) const
  Map inverseAutomorphism( const Map& M) const
protected:
  FreeGroup( FreeGroupRep* newrep ) :
    DerivedObjectOf<FGGroup,FreeGroupRep>(newrep)
private:
};
```

———-

11.14 Group/include/FreeGroupRep.h

— **FreeGroupRep.h** —

```
#include "FGGroupRep.h"
#include "File.h"
```

———-

11.14.1 struct FreeGroupRep

— **struct FreeGroupRep** —

```
struct FreeGroupRep : FGGroupRep {
public:
  FreeGroupRep( int rank ) : FGGroupRep( rank )
  FreeGroupRep( const VectorOf<Chars>& gennames ) : FGGroupRep( gennames )
private:
  FreeGroupRep& operator = ( const FreeGroupRep& fgr );
public:
  PureRep* clone( ) const
  static const Type theFreeGroupType;
  static Type type( )
  Type actualType( ) const
  int order( ) const;
  Trichotomy isTrivial( ) const;
  Trichotomy isFinite( ) const;
  Trichotomy isInfinite( ) const;
  Trichotomy isAbelian( ) const;
  bool compare( const GroupRep* G ) const;
  VectorOf<Word> nielsenBasis(const VectorOf<Word>& V) const;
  Bool isAutomorphism(const VectorOf<Word>& V) const;
  Bool isInnerAutomorphism(const VectorOf<Word>& V) const;
  bool isIAAutomorphism(const VectorOf<Word>& V) const;
  VectorOf<Word> inverseAutomorphism(const VectorOf<Word>& V) const;
  Trichotomy isTrivialElt( const Elt& e ) const
  Elt eval( const Word& w ) const;
  Trichotomy areEqual(const Elt& e1, const Elt& e2) const;
  Trichotomy wordProblem( const Word& w ) const;
  Trichotomy conjugacyProblem( const Word& u, const Word& v ) const;
  Trichotomy conjugacyProblem( const Word& u, const Word& v ,Word& c) const;
  Bool inCommutatorSG(const Word& w) const;
  Bool isCommutator(const Word& w, Word& u1, Word& u2) const;
  Chars productOfCommutators( const Word& w , File& file );
  Chars productOfSquares( const Word& w , File& file );
  Word getN_thElement( int n ) const;
  int numberOfElement( const Word& w ) const;
  void printOn(ostream&) const;
```

```
  GroupRep* readFrom(istream&, Chars&) const;
public:
```

———-

11.14.2 class NielsenBasis

— class NielsenBasis —

```
  class NielsenBasis {
  public:
    NielsenBasis( const VectorOf<Word>& vec,
      const VectorOf<Chars>& namesOfGenerators,
      bool writeToFile );
    VectorOf<Word> oldGenerators( ) const
    VectorOf<Word> newGenerators( );
    VectorOf<Word> expressNewGensInOldGens( );
    Chars getFileName( ) const;
  private:
    void computeNielsenBasis( );
    void printParentGroup( ostream& ostr ) const;
    void printWord( ostream& ostr, const Word& w ) const;
    void printBasis( ostream& ostr, const VectorOf<Word>& basis,
                     bool lineByLine ) const;
    void printTransformation( ostream& ostr, const VectorOf<Word>& basis,
            int i, int g1, int g2 ) const;
    VectorOf<Chars> theNamesOfGenerators;
    VectorOf<Word> theOldGenerators;
    VectorOf<Word> theNewGenerators;
    VectorOf<Word> newGeneratorsExprssedInOldGerators;
    Chars theFileName;
    bool keepDetails;
    bool basisComputed;
  };
```

———-

— t1 —

```
private:
  Word getN_thWord( int n, int numberOfGenerators ) const;
  Generator numToGen( int n ) const;
  int genToNum( Generator g ) const;

};
```

———-

11.15 Group/include/GeneralWhitehead.h

— **GeneralWhitehead.h** —

```
#include "Map.h"
#include "FreeGroup.h"
#include "Timer.h"
#include "File.h"
```

———-

11.15.1 class GeneralWhitehead

— **class GeneralWhitehead** —

```
class GeneralWhitehead {
public:
  GeneralWhitehead( const FreeGroup& F, bool keepDetails = false );
  ~GeneralWhitehead( );
  void startComputation( const VectorOf<Word>& );
  void startComputation( const Word& );
  bool continueComputation( );
  bool done( ) const
  bool extendsToFreeBasis( )
  Map getAutomorphism( )
  Chars getFileName( )
private:
  static const int timerValue = 1000;
  bool bDone;
  bool bStart;
  bool doesExtend;
  bool itIsInterrupted;
  Timer timer;
  const FreeGroup theGroup;
  int numberOfGenerators;
  VectorOf<Word> images;
  int *numOfAuto;
  Map automorphism;
  VectorOf<Word> theVector;
  int theVectorLen;
  int theVectorCommonLength;
  int saveN;
  File* file;
  bool hasOneWordOnly;
  GeneralWhitehead( const GeneralWhitehead& );
  GeneralWhitehead& operator = ( const GeneralWhitehead& );
  Word makeWhiteheadAutomorpism( Generator, int, Generator ) const;
  void prepareAutomorphism( Map& automorphism ) const;
```

```
  void setResult( bool DoesExtend );
  void finishComputation( bool DoesExtend );
  int commonLength( const VectorOf<Word>& );
};

inline void GeneralWhitehead::finishComputation( bool DoesExtend )
```

———-

11.16 Group/include/GroupFastChecks.h

— **GroupFastChecks.h** —

```
#include "FPGroup.h"
```

———-

11.16.1 class GroupFastChecks

— **class GroupFastChecks** —

```
class GroupFastChecks {
public:
  GroupFastChecks( const FPGroup& G ) :
    theGroup( G ), numOfGens( G.numberOfGenerators() ),
    relators( G.getRelators() )
  bool hasTrivialPresentation( )
  bool numOfRelsLessNumOfGens( )
  bool existsGenWithExpSumZeroInEveryRelator( Generator& );
  VectorOf<int> getExpSumOfGen( const Generator& g);
  int GCDOfExpSumOfGen( const Generator& g );
private:
  FPGroup theGroup;
  int numOfGens;
  SetOf<Word> relators;
};
```

———-

11.17 Group/include/Group.h

— **Group.h** —

```
#include "Type.h"
#include "IStreamPoll.h"
#include "GroupRep.h"
```

———-

11.17.1 class Group

— **class Group** —

```
class Group : public GenericObject {
public:
  int operator == ( const Group& G ) const
  int operator != ( const Group& G ) const
  static Type type( )
  Type actualType( ) const
  int order( ) const;
  Trichotomy isTrivial( ) const
  Trichotomy isFinite( ) const
  Trichotomy isInfinite( ) const
  Trichotomy isAbelian( ) const
  Elt makeIdentity( ) const
  Bool isSyntacticIdentity(const Elt& e) const
  Trichotomy isTrivialElt( const Elt& e ) const
  Trichotomy areEqual(const Elt& e1, const Elt& e2) const
  Elt firstElt( ) const
  Elt nextElt(const Elt& e) const
  Elt multiply(const Elt& e1, const Elt& e2) const
  Elt inverseOf(const Elt& e) const
  Elt raiseToPower(const Elt& e, int n) const
  Elt conjugateBy(const Elt& e1, const Elt& e2) const
  Elt commutator(const Elt& e1, const Elt& e2) const
  SetOf<Elt> setMultiply(const SetOf<Elt>& S1, const SetOf<Elt>& S2) const;
  SetOf<Elt> setMultiply(const Elt& e, const SetOf<Elt>& S) const;
  SetOf<Elt> setMultiply(const SetOf<Elt>& S, const Elt& e) const;
  SetOf<Elt> conjugateBy(const SetOf<Elt>& S1, const SetOf<Elt>& S2) const;
  SetOf<Elt> conjugateBy(const Elt& e, const SetOf<Elt>& S) const;
  SetOf<Elt> conjugateBy(const SetOf<Elt>& S, const Elt& e) const;
  void closeUnderInverses(SetOf<Elt>& S) const;
  friend ostream& operator << ( ostream&, const Group& );
  friend IStreamPoll operator >> ( istream&, Group& );
  friend ostream& operator < ( ostream& ostr, const Group& G )
  friend istream& operator > ( istream& istr, Group& G )
protected:
  const GroupRep* look( ) const
  GroupRep* enhance( ) const
  GroupRep* change( )
  Group( GroupRep* newrep ) : GenericObject(newrep)
```

```
private:
  friend void debugPrint(ostream&, const Group& g);
public:
  bool consistent( ) const
};
```

———-

11.18 Group/include/GroupRep.h

— **GroupRep.h** —

```
#include "GenericObject.h"
#include "Type.h"
#include "Chars.h"
#include "Elt.h"
#include "Set.h"
#include "IPC.h"
#include "ExtendedIPC.h"
```

———-

11.18.1 struct GroupRep

— **struct GroupRep** —

```
struct GroupRep : GenericRep {
public:
  GroupRep( ) : theOrder(-1)
  static const Type theGroupType;
  static Type type( )
  virtual Type actualType( ) const
private:
  GroupRep& operator = ( const GroupRep& );
public:
  virtual bool compare( const GroupRep* G ) const = 0;
  virtual int order( ) const = 0;
  virtual Trichotomy isTrivial( ) const = 0;
  virtual Trichotomy isFinite( ) const = 0;
  virtual Trichotomy isInfinite( ) const = 0;
  virtual Trichotomy isAbelian( ) const = 0;
  virtual Elt makeIdentity( ) const = 0;
  virtual Bool isSyntacticIdentity(const Elt&) const = 0;
  virtual Trichotomy isTrivialElt( const Elt& e ) const
  virtual Trichotomy areEqual(const Elt&, const Elt&) const = 0;
  virtual Elt firstElt( ) const = 0;
```

```
  virtual Elt nextElt(const Elt&) const = 0;
  virtual Elt multiply(const Elt&, const Elt&) const = 0;
  virtual Elt inverseOf(const Elt&) const = 0;
  virtual Elt raiseToPower(const Elt&, int) const;
  virtual Elt conjugateBy(const Elt&, const Elt&) const;
  virtual Elt commutator(const Elt&, const Elt&) const;
  virtual void printOn(ostream&) const = 0;
  virtual GroupRep* readFrom(istream&, Chars&) const = 0;
  virtual void write( ostream& ostr ) const
  virtual void read( istream& istr )
  int theOrder;
};
```

———-

11.19 Group/include/Homology.h

— **Homology.h** —

```
#include "BlackBox.h"
#include "KBMachine.h"
#include "SmithNormalForm.h"
```

———-

11.19.1 class Homology

— **class Homology** —

```
class Homology {
public:
  Homology(const KBMachine M, int start_dim, int end_dim);
  ~Homology( );
  bool workOnNextGroup( );
  int getTorsionFreeRank( ) const;
  VectorOf<Integer> getTorsionInvariants( ) const;
private:
  char tempFileName[100];
  BlackBox* chom;
  enum State { READING, REDUCING, GOT_ONE };
  State myState;
  int dimensionToDo;
  int lastDimension;
  SmithNormalForm* SNF;
  bool booted;
  int previousTorsionFreeRank;
```

```
  int columns;
  Integer** readMatrix(int& rows, int& cols) const;
};
```

———-

11.20 Group/include/MSCGConjugacyProblem.h

— MSCGConjugacyProblem.h —

```
#include "MSCGroup.h"
```

———-

11.20.1 class MSCGConjugacyProblem

— class MSCGConjugacyProblem —

```
class MSCGConjugacyProblem {
public:
  MSCGConjugacyProblem ( const MSCGroup& G, const Word& u, const Word& v ) :
    MSCG ( G ), U( u.freelyReduce() ), V( v.freelyReduce() ),
    areConjugate ( DONTKNOW ), theConjugator ( ), doneStatus ( false ),
    startStatus ( false ), UConjugator( ), VConjugator( )
  void startComputation( );
  bool continueComputation( );
  bool done( ) const
  Trichotomy answer( ) const
  Word conjugator( ) const
private:
  MSCGConjugacyProblem( const MSCGConjugacyProblem& );
  MSCGConjugacyProblem& operator = ( const MSCGConjugacyProblem& );
  void setCPResult( Trichotomy result );
  void finishCP( Trichotomy result );
  const MSCGroup& MSCG;
  bool doneStatus;
  bool startStatus;
  Trichotomy areConjugate;
  Word theConjugator;
  int maxLen;
  Word UConjugator;
  Word VConjugator;
  enum stateType{CYCLE_BY_RELATORS, CYCLE_BY_PIECES, CYCLE_BY_RELATORS2};
  stateType state;
  bool firstPart;
  Word U;
```

```
  Word V;
  int ULen;
  int VLen;
  SetOf<Word> cycV;
  SymmetricRelators *shortRelators;
  SymmetricRelatorsIterator *shortIter;
  SymmetricRelatorsIterator *shortIter2;
  Word relator;
  int relatorLen;
  int pieceLen;
  Word relatorOne;
  int relatorOneLen;
  int pieceOneLen;
  Word pieceOne;
  Word relatorTwo;
  int relatorTwoLen;
};
```

———-

11.21 Group/include/MSCGroup.h

— **MSCGroup.h** —

```
#include "FPGroupRep.h"
#include "FPGroup.h"
#include "SymmetricRelators.h"
#include "ShortenByRelators.h"
#include "PresentationParser.h"
```

———-

11.21.1 class MSCGroup

— **class MSCGroup** —

```
class MSCGroup {
friend class MSCGConjugacyProblem;
public:
  MSCGroup( int ngens = 0 ) : numberOfGenerators ( ngens )
  MSCGroup( int ngens, const SetOf<Word>& rels, int lambda = -1 )
  MSCGroup( FPGroup G, int lambda = -1 )
  ~MSCGroup( )
  int order() const;
  Trichotomy isTrivial() const;
  Trichotomy isFinite() const;
```

```
  Trichotomy isInfinite() const;
  Trichotomy isAbelian() const;
  Trichotomy isFree() const;
  int getMSCLambda() const
  Word shortenByRelators(const Word& w) const;
  Word cyclicallyShortenByRelators(const Word& w) const;
  Elt eval( const Word& w ) const;
  Trichotomy wordProblem( const Word& w ) const;
  Trichotomy areEqual(const Word& w1, const Word& w2) const
private:
  MSCGroup( const MSCGroup& );
  MSCGroup& operator = ( const MSCGroup& );
  void calculateLambda( );
  void makeMSCGroup( int ngens, const SetOf<Word>& rels, int lambda = -1 );
  int numOfGens() const
  SetOf<Word> getRelators() const
  Word cyclicallyShortenByRelators( const Word& w, Word& conjugator) const;
  SymmetricRelators *symmetricRelators;
  ShortenByRelators *shortenByRels;
  int mscLambda;
  int numberOfGenerators;
};
```

———-

11.22 Group/include/ORWordProblem.h

— **ORWordProblem.h** —

```
#include "BlackBox.h"
#include "Word.h"
```

———-

11.22.1 class ORWordProblem

— **class ORWordProblem** —

```
class ORWordProblem {
public:
  ORWordProblem(const Word& relator);
  ~ORWordProblem( )
  void findAnswer(const Word& testWord);
  bool goodStatus( )
  bool isTrivial( );
private:
```

```
  BlackBox* orwp;
  const Word theRelator;
  bool status;
  bool result;
  void printWord(const Word& w, ostream& ostr);
};
```

———-

11.23 Group/include/PowerSeriesWP.h

— **PowerSeriesWP.h** —

```
#include "FreeGroup.h"
#include "Word.h"
```

———-

11.23.1 class PowerSeriesWP

— **class PowerSeriesWP** —

```
class PowerSeriesWP {
public:
  PowerSeriesWP( const FreeGroup& group, int nClass );
  bool isTrivial(const Word& w);
private:
```

———-

11.23.2 struct State

— **struct State** —

```
  struct State {
    State( int Coef = 1, int WPos = 0, int VPos = 0, int Power = 1) {
      coef = Coef; wPos = WPos; vPos = VPos, power = Power; }
    int coef; int wPos; int vPos; int power;
  };
```

———-

11.23.3 struct Stack

— **struct Stack** —

```
  struct Stack{
    Stack( int c ) : stackLen(0)
    ~Stack( )
    void put( State s )
    void get( State& s)
    int length( )
    State& state()
  private:
    State* states;
    int stackLen;
  };
  FreeGroup F;
  int c;
};
```

———-

11.24 Group/include/PresentationParser.h

— **PresentationParser.h** —

```
#include "Set.h"
#include "WordParser.h"
#include "FreeGroup.h"
#include "FPGroup.h"
```

———-

11.24.1 class PresentationParser

— **class PresentationParser** —

```
class PresentationParser : public WordParser {
public:
  enum BraceType { SET, PAREN };
public:
  PresentationParser(istream &istr) : WordParser(istr)
  VectorOf<Chars> parseGeneratorList( Chars& );
  SetOf<Word> parseWordSet( const VectorOf<Chars>&, Chars& );
  VectorOf<Word> parseWordList( const VectorOf<Chars>&, Chars& );
  SetOf<Word> parseSetOfWords( const VectorOf<Chars>&, Chars& );
  VectorOf<Word> parseVectorOfWords( const VectorOf<Chars>&, Chars&,
```

```
            BraceType brace = SET );
  virtual Word parseRelator( const VectorOf<Chars>&, Chars& );
  SetOf<Word> parseRelatorList( const VectorOf<Chars>&, Chars& );
  FreeGroupRep* parseFreeGroup( Chars& );
  FPGroupRep* parseFPGroup( Chars& );
private:
  bool getGeneratorRange( const Chars& , VectorOf<Chars>&, Chars& );
  bool getRangeOf( const Chars& ,Chars&, int& );
};
```

———-

11.25 Group/include/PrimeNumbers.h

— **PrimeNumbers.h** —

```
#include "Integer.h"
#include "Vector.h"
#include "DArray.h"
```

———-

11.25.1 struct IntProblems

— **struct IntProblems** —

```
struct IntProblems
{
  Integer gcdOfVector(const VectorOf<Integer>& vc)const;
  Integer powerOf(const Integer& o, const Integer& p)const;
  void findCoeff(Integer p, Integer q, Integer& x, Integer& y,
                 Integer mod) const;

};
```

———-

11.25.2 class PrimeNumbers

— **class PrimeNumbers** —

```
class PrimeNumbers {
public:
   PrimeNumbers() : currentNumber(2)
   const Integer& current() const
```

```
   const Integer&  nextNumber()
   void setCurrent(const Integer& num)
   bool isPrime(Integer num) const
   DArray<Integer> getPrimeDecom(const Integer& num);
   void printPrimeDecom(ostream& ostr,const Integer& num);
private:
   Integer currentNumber;
};
```

———-

11.26 Group/include/Products.h

— **Products.h** —

```
#include "FPGroup.h"
#include "Map.h"
```

———-

11.26.1 class FreeProduct

— **class FreeProduct** —

```
class FreeProduct
{
public:
  enum CreatError { NONE, DOUBLE_GENS };
  FreeProduct( const FPGroup& g1, const FPGroup& g2):
    theStatus( NONE ),
    G1( g1 ),
    G2( g2 ),
    gensInitialized( false )
  virtual Map mapFromG1() const;
  virtual Map mapFromG2() const;
  virtual Map mapToG1() const;
  virtual Map mapToG2() const;
  operator FPGroup( ) const
protected:
  void createGenerators();
  void createRelators();
  VectorOf<Chars> theGenerators;
  SetOf<Word> theRelators;
  FPGroup G1;
  FPGroup G2;
  CreatError theStatus;
```

```
private:
  FreeProduct( const FreeProduct&);
  FreeProduct& operator = ( const FreeProduct& );
  bool gensInitialized;
};
```

———-

11.26.2 class DirectProduct

— class DirectProduct —

```
class DirectProduct : public FreeProduct
{
 public:
   DirectProduct( const FPGroup& g1, const FPGroup& g2);
 private:
  DirectProduct( const DirectProduct&);
  DirectProduct& operator = ( const DirectProduct& );
};
```

———-

11.27 Group/include/RandomAutomorphism.h

— RandomAutomorphism.h —

```
#include "RandomNumbers.h"
#include "FreeGroup.h"
#include "Map.h"
```

———-

11.27.1 class RandomAutomorphism

— class RandomAutomorphism —

```
class RandomAutomorphism
{
public:
  RandomAutomorphism( const FreeGroup& F, int seed );
  ~RandomAutomorphism( );
  VectorOf<Word> getGeneratingVector( int avgNumGens );
  Map getAutomorphism( int avgnumGens );
private:
```

```
  FreeGroup theGroup;
  int numberOfGroupGens;
  NormalRandom numGensPicker;
  UniformRandom typeGenPicker;
};
```

———-

11.28 Group/include/RipsConstruction.h

— RipsConstruction.h —

```
#include "FPGroup.h"
```

———-

11.28.1 class RipsConstruction

— class RipsConstruction —

```
class RipsConstruction {
public:
  RipsConstruction( const FPGroup& G ) :  theGroup( G )
  FPGroup getRipsConstruction( ostream* out = NULL ) const;
private:
  FPGroup theGroup;
};
```

———-

11.29 Group/include/ShortenByRelators.h

— ShortenByRelators.h —

```
#include "SymmetricRelators.h"
```

———-

11.29.1 class ShortenByRelators

— class ShortenByRelators —

```
class ShortenByRelators {
public:
  ShortenByRelators ( const SymmetricRelators& symmetricRelators );
  ShortenByRelators ( const SetOf<Word>& );
  ~ShortenByRelators( );
  Word getShortenWord( const Word& w ) const;
private:
  Word* sortRelators;
  int sortRelatorsLen;
};
```

———-

11.30 Group/include/SmithNormalForm1.h

— **SmithNormalForm1.h** —

```
#include <Integer.h>
#include "Vector.h"
#include "File.h"
#include "DArray.h"
```

———-

11.30.1 class SmithNormalForm1

— **class SmithNormalForm1** —

```
class SmithNormalForm1 {
public:
  SmithNormalForm1()
  SmithNormalForm1(DArray<Integer>);
  DArray<Integer> LeftFactor() const
  DArray<Integer> LeftFactorInv() const
  DArray<Integer> RightFactor() const
  DArray<Integer> RightFactorInv() const
  VectorOf<Integer> invariants() const
  int rank() const
  Chars getFileName() const
private:
  File f;
  DArray<Integer> A, P, P1, Q, Q1;
  VectorOf<Integer> inv;
  int findmin(int);
  int plus(int,int,Integer,char);
  void swap(int,int,char);
```

```
  void show(Chars,DArray<Integer>);
};
```

———-

11.31 Group/include/SmithNormalForm.h

— **SmithNormalForm.h** —

```
#include "ExtendedIPC.h"

#include "Vector.h"
```

———-

11.31.1 class SmithNormalForm

— **class SmithNormalForm** —

```
class SmithNormalForm {
public:
  SmithNormalForm(Integer** theMatrix, int rows, int cols);
  bool continueComputation( );
  int getTorsionFreeRank( ) const;
  VectorOf<Integer> getTorsionInvariants( ) const;
private:
  Integer** matrix;
  int height, width;
  VectorOf<Integer> theInvariants;
  int rankOfFreePart;
  int i, j;
  VectorOf<Integer> resultTemp;
  bool done;
  Integer abs( const Integer& a ) const
  Integer sign( const Integer& a ) const
  Integer GCD( Integer a, Integer b ) const;
};
```

———-

11.32 Group/include/SymmetricRelators.h

— **SymmetricRelators.h** —

```
#include "Set.h"
#include "Word.h"
#include "Vector.h"

template <class T>
T arbitraryElementOf ( const SetOf<T>& S );

SetOf<Word>& unsymmetrise ( SetOf<Word>& S );

Word minimalWord( const Word& w );

SetOf<Word>& minimizeSet( SetOf<Word>& S );

inline int rootLength( const Word& w )
```

———-

11.32.1 class SymmetricRelators

— class SymmetricRelators —

```
class SymmetricRelators {
friend class SymmetricRelatorsIterator;
public:
  SymmetricRelators ( const SetOf<Word>& relators );
  ~SymmetricRelators ( );
  int cardinality( ) const
  const VectorOf<Word>& getRelators( )
private:
  VectorOf<Word> symmetricRelators;
  int *rootLengths;
};
```

———-

11.32.2 class SymmetricRelatorsIterator

— class SymmetricRelatorsIterator —

```
class SymmetricRelatorsIterator {
public:
  SymmetricRelatorsIterator(const SymmetricRelators& S) :
      iterSet(S), relatorNumber(0), cpNum(0), invFlag(false),
      iterSetLen(S.symmetricRelators.length())
  SymmetricRelatorsIterator( const SetOf<Word>& S );
  Bool operator == ( const SymmetricRelatorsIterator& I ) const
  Word value( ) const
```

```
  Bool next( )
  bool done( ) const
  void reset( )
private:
  const SymmetricRelators& iterSet;
  int iterSetLen;
  int relatorNumber;
  int cpNum;
  int rootLen;
  bool invFlag;
  Word curRelator;
};
```

———-

11.33 Group/include/TietzeTrekker.h

— **TietzeTrekker.h** —

```
#include "BlackBox.h"
#include "FPGroup.h"
```

———-

11.33.1 class TietzeTrekker

— **class TietzeTrekker** —

```
class TietzeTrekker {
public:
  TietzeTrekker(const FPGroup& G);
  TietzeTrekker(const VectorOf<Chars>& genNames, const SetOf<Word>& relators);
  Bool sanityCheck( )
  Bool getFactoid(Bool queuePresentations, long giveUpLimit = 100);
  Bool knownToBeTrivial( )
  Bool knownToBeCyclic( )
  Bool knownToHaveFreeFactor( )
  Bool knownToBeFinite( )
  Bool knownToBeFree( )
  int getOrder( );
  int getRank( );
  FPGroup getPresentation( );
private:
  enum { TRIVIAL = 1, CYCLIC = 2, HAS_FREE_FACTOR = 4, FINITE = 8, FREE = 16 };
  ListOf<FPGroup> presentations;
  BlackBox TietzeTrek;
```

```
  int flags;
  int order;
  int rank;
  int status;
  void initialize(const VectorOf<Chars>& genNames,
                  const SetOf<Word>& relators);
  void printWord(ostream& ostr, const Word& w);
  void printGenerator(ostream& ostr, int g);
};
```

———-

11.34 Group/include/TTP.h

— **TTP.h** —

```
#include "File.h"
#include "FPGroup.h"
```

———-

11.34.1 class TTP

— **class TTP** —

```
class TTP {
public:
  TTP( const FPGroup& G ) : theGroup( G )
  Chars getFileName( ) const
  bool run( ) const;
private:
  FPGroup theGroup;
  File file;
};
```

———-

11.35 Group/include/WordEnumerator.h

— **WordEnumerator.h** —

```
#include "FreeGroup.h"
```

———-

11.35.1 class VectorEnumerator

— class VectorEnumerator —

```
class VectorEnumerator
{
public:
  VectorEnumerator( int n = 2) : current(0)
  void reset(int c = 0)
  void next()
  VectorOf<Integer> vector() const
  VectorOf<Integer> getVectorOf ( Integer c ) const;
  Integer getNumberOfVector(VectorOf<Integer> v) const;
  int vectorsLength() const
 private:
  VectorOf<Integer> getPairOf ( Integer c ) const;
  Integer getNumberOfPair(const Integer& n,const Integer& m ) const
  int current;
  int lengthOfVectors;
};
```

———-

11.35.2 class WordEnumerator

— class WordEnumerator —

```
class WordEnumerator
{
public:
  WordEnumerator( const FreeGroup& f, int n = 2): ve( n ), theGroup( f )
  void reset(int c = 0)
  void next()
  VectorOf<Word> getWords();
private:
  FreeGroup theGroup;
  VectorEnumerator ve;
};
```

———-

12 The KB classes

12.1 KB/include/DiffHistory.h

— DiffHistory.h —

```
#include "Word.h"
#include "Vector.h"

typedef int State;
#include "DiffHistoryRep.h"
#include "WordOrder.h"
```

———-

12.1.1 class DiffHistory

— class DiffHistory —

```
class DiffHistory : public ObjectOf<DiffHistoryRep> {
public:
  DiffHistory() : ObjectOf< DiffHistoryRep > (new DiffHistoryRep())
  DiffHistory(const WordOrder & word_order) :
    ObjectOf<DiffHistoryRep> ( word_order.buildDiffHistoryRep() )
  DiffHistory(State d,int g,int h,const WordOrder & word_order) :
    ObjectOf<DiffHistoryRep> ( word_order.buildDiffHistoryRep(d,g,h) )
  DiffHistory(const DiffHistory & dh,State d,int g,int h,
              const Word & wd,const WordOrder & word_order) :
    ObjectOf<DiffHistoryRep> (word_order.update(*dh.look(),d,g,h,wd) )
  int hash() const
  Bool empty() const
  State getDiff() const
  int operator == ( const DiffHistory & dh ) const
  int operator != ( const DiffHistory & dh ) const
  Bool sameLengthWords() const
  void improveBy(const DiffHistory & dh)
  Bool reduction(int g,int h,const WordOrder & word_order) const
  Bool possibleReductionAhead() const
  AheadInfoRep *  buildAheadInfoRep() const
  void printOn(ostream &str = cout) const
  inline friend ostream& operator << ( ostream& ostr, const DiffHistory& dh )
protected:
  typedef ObjectOf<DiffHistoryRep> R;
  DiffHistory( DiffHistoryRep *p ) : R(p)
};
```

———-

12.1.2 class AheadInfo

— class AheadInfo —

```
class AheadInfo : public ObjectOf<AheadInfoRep> {
public:

  AheadInfo() :
    ObjectOf< AheadInfoRep > (new AheadInfoRep())
  AheadInfo(const DiffHistory & dh) :
  ObjectOf<AheadInfoRep> ( dh.buildAheadInfoRep() )
  AheadInfo
    (const AheadInfo & ai,int g,const WordOrder & word_order) :
  ObjectOf<AheadInfoRep> (word_order.update(*ai.look(),g) )
  Bool possibleReduction(int g,const WordOrder & word_order) const
protected:
  typedef ObjectOf<AheadInfoRep> R;
  AheadInfo( AheadInfoRep *p ) : R(p)
};
```

———-

12.1.3 class DiffHistoryVtx

— class DiffHistoryVtx —

```
class DiffHistoryVtx : public ObjectOf<DiffHistoryVtxRep> {
public:
  DiffHistoryVtx() :
    ObjectOf< DiffHistoryVtxRep > (new DiffHistoryVtxRep())
  DiffHistoryVtx(const WordOrder & word_order) :
  ObjectOf<DiffHistoryVtxRep>
    (word_order.buildDiffHistoryVtxRep() )
  DiffHistoryVtx(State d,int g,int h,const WordOrder & word_order)
    :ObjectOf<DiffHistoryVtxRep>
            ( word_order.buildDiffHistoryVtxRep(d,g,h))
  DiffHistoryVtx
    (DiffHistoryVtx * dhp,State d,int g,int h,const WordOrder & word_order) :
  ObjectOf<DiffHistoryVtxRep> (word_order.update(*(dhp->look()),d,g,h,dhp) )
  Bool reduction(int g,int h,const WordOrder & word_order) const
  Trichotomy betterThan(DiffHistoryVtx* dhp) const
  Bool possibleReductionAhead() const
  Bool possibleReductionAhead(int g,const WordOrder & word_order) const
  void printOn(ostream &str = cout) const
  inline friend ostream& operator << (ostream& ostr,const DiffHistoryVtx& dh)
  State getDiff() const
  int getGenerator() const
  DiffHistoryVtx *getBackptr() const
  int getLength() const
protected:
  typedef ObjectOf<DiffHistoryVtxRep> R;
  DiffHistoryVtx( DiffHistoryVtxRep *p ) : R(p)
```

```
};
```

———-

12.2 KB/include/DiffHistoryRep.h

— **DiffHistoryRep.h** —

```
#include "Word.h"
#include "Vector.h"
#include "RefCounter.h"

typedef int State;
```

———-

12.2.1 class AheadInfoRep

— **class AheadInfoRep** —

```
class AheadInfoRep : public RefCounter {
public:
  AheadInfoRep()
  AheadInfoRep (const AheadInfoRep & ai)
  virtual AheadInfoRep *clone() const
  virtual ~AheadInfoRep()
};
```

———-

12.2.2 class SLAheadInfoRep

— **class SLAheadInfoRep** —

```
class SLAheadInfoRep : public AheadInfoRep {
public:
  SLAheadInfoRep()
  SLAheadInfoRep(const SLAheadInfoRep & ai)
  AheadInfoRep *clone() const
  ~SLAheadInfoRep()
private:
};
```

———-

12.2.3 class WtSLAheadInfoRep

— class WtSLAheadInfoRep —

```
class WtSLAheadInfoRep : public AheadInfoRep {
public:
  WtSLAheadInfoRep()
  WtSLAheadInfoRep(int wd)
  WtSLAheadInfoRep (const WtSLAheadInfoRep & ai)
  AheadInfoRep *clone() const
  ~WtSLAheadInfoRep()
  int getWtDiff() const
private:
  int wtdiff;
};
```

———-

12.2.4 class WtLexAheadInfoRep

— class WtLexAheadInfoRep —

```
class WtLexAheadInfoRep : public AheadInfoRep {
public:
  WtLexAheadInfoRep(): sign(0)
  WtLexAheadInfoRep(int wd,int sgn): wtdiff(wd), sign(sgn)
  WtLexAheadInfoRep(const WtLexAheadInfoRep & ai) :
     wtdiff(ai.wtdiff), sign(ai.sign)
  AheadInfoRep *clone() const
  ~WtLexAheadInfoRep()
  int getWtDiff() const
  int getSign() const
private:
  int wtdiff;
  int sign;
};
```

———-

12.2.5 class DiffHistoryRep

— class DiffHistoryRep —

```
class DiffHistoryRep : public RefCounter {
public:
  DiffHistoryRep()
```

```
  DiffHistoryRep(const DiffHistoryRep & dh)
  virtual DiffHistoryRep *clone() const
  virtual ~DiffHistoryRep()
  virtual int hash() const
  virtual Bool empty() const
  virtual State getDiff() const
  virtual int operator== ( const DiffHistoryRep & dh) const
  virtual DiffHistoryRep&  operator= ( const DiffHistoryRep & dh )
  virtual Bool sameLengthWords() const
  virtual void improveBy(const DiffHistoryRep & dh)
  virtual Bool possibleReductionAhead() const
  virtual AheadInfoRep *  buildAheadInfoRep() const
  virtual void printOn(ostream &ostr = cout) const
  virtual inline ostream& operator << ( const DiffHistoryRep& dh )
```

———-

12.2.6 class SLDiffHistoryRep

— class SLDiffHistoryRep —

```
class SLDiffHistoryRep : public DiffHistoryRep {
public:
  SLDiffHistoryRep(): d(0),c0(0),c1(0)
  SLDiffHistoryRep(State D,int C0,int C1):
    d(D),c0(C0),c1(C1)
  SLDiffHistoryRep(const SLDiffHistoryRep & dh) :
    d(dh.d), c0(dh.c0), c1 (dh.c1)
  DiffHistoryRep *clone() const
  ~SLDiffHistoryRep()
  int hash() const
  Bool empty() const
  int getDiff() const
  int getC0() const
  int getC1() const
  int operator== ( const DiffHistoryRep & dh) const
  DiffHistoryRep&  operator= ( const DiffHistoryRep & dh )
  Bool sameLengthWords() const
  void improveBy(const DiffHistoryRep & dh)
  Bool possibleReductionAhead() const
  AheadInfoRep * buildAheadInfoRep() const
  void printOn(ostream &ostr = cout) const
  inline friend ostream& operator << ( ostream& ostr,
                                       const SLDiffHistoryRep& dh )
private:
  State d;
  int c0, c1;
};
```

———-

12.2.7 class WtSLDiffHistoryRep

— class WtSLDiffHistoryRep —

```
class WtSLDiffHistoryRep : public DiffHistoryRep {

public:

  WtSLDiffHistoryRep(): d(0),c0(0),w0(0),c1(0),w1(0){};
  WtSLDiffHistoryRep(State D,int C0,int W0,int C1,int W1):
    d(D),c0(C0),w0(W0),c1(C1),w1(W1){};

  WtSLDiffHistoryRep(const WtSLDiffHistoryRep & dh):
    d(dh.d), c0(dh.c0), w0(dh.w0), c1(dh.c1), w1(dh.w1)
  DiffHistoryRep *clone() const
  ~WtSLDiffHistoryRep()
  int hash() const
  Bool empty() const
  int getDiff() const
  int getC0() const
  int getW0() const
  int getC1() const
  int getW1() const
  int operator== ( const DiffHistoryRep & dh) const
  DiffHistoryRep&  operator= ( const DiffHistoryRep & dh )
  Bool sameLengthWords() const
  void improveBy(const DiffHistoryRep & dh)
  Bool possibleReductionAhead() const
  AheadInfoRep * buildAheadInfoRep() const
  void printOn(ostream &ostr = cout) const
  inline friend ostream& operator << (ostream& ostr,
                                      const WtSLDiffHistoryRep& dh )
private:
  State d;
  int c0, c1;
  int w0, w1;
};
```

———-

12.2.8 class WtLexDiffHistoryRep

— class WtLexDiffHistoryRep —

```
class WtLexDiffHistoryRep : public DiffHistoryRep {
```

```
public:
  WtLexDiffHistoryRep(): d(0),c0(0),w0(0),c1(0),w1(0)
  WtLexDiffHistoryRep(State D,int C0,int W0,int C1,int W1):
    d(D),c0(C0),w0(W0),c1(C1),w1(W1)
  WtLexDiffHistoryRep(const WtLexDiffHistoryRep & dh):
    d(dh.d), c0(dh.c0), w0(dh.w0), c1(dh.c1), w1(dh.w1)
  DiffHistoryRep *clone() const
  ~WtLexDiffHistoryRep()
  int hash() const
  Bool empty() const
  int getDiff() const
  int getC0() const
  int getW0() const
  int getC1() const
  int getW1() const
  int operator== ( const DiffHistoryRep & dh) const
  DiffHistoryRep&  operator= ( const DiffHistoryRep & dh )
  Bool sameLengthWords() const
  void improveBy(const DiffHistoryRep & dh)
  Bool possibleReductionAhead() const
  AheadInfoRep * buildAheadInfoRep() const
  void printOn(ostream &ostr = cout) const
  inline friend ostream& operator <<
        ( ostream& ostr, const WtLexDiffHistoryRep& dh )
private:
  State d;
  int c0, c1;
  int w0, w1;
};
```

———-

12.2.9 class DiffHistoryVtxRep

— class DiffHistoryVtxRep —

```
class DiffHistoryVtxRep : public RefCounter {
public:
  DiffHistoryVtxRep(): diff(0), gen(0), backptr(0),len(0)
  DiffHistoryVtxRep(State D,int G,DiffHistoryVtx * ptr,int L):
     diff(D), gen(G), backptr(ptr),len(L)
  DiffHistoryVtxRep(const DiffHistoryVtxRep & dh)
  DiffHistoryVtxRep *clone() const
  ~DiffHistoryVtxRep()
  State getDiff() const
  int getGenerator() const
  DiffHistoryVtx *getBackptr() const
  int getLength() const
```

```
  int operator== ( const DiffHistoryVtxRep & dh) const
  DiffHistoryVtxRep&  operator= ( const DiffHistoryVtxRep & dh )
  virtual Trichotomy betterThan(const DiffHistoryVtxRep & dh) const
  virtual Bool possibleReductionAhead() const
  void printOn(ostream &ostr = cout) const
  inline friend ostream& operator << (ostream & ostr,
                                      const DiffHistoryVtxRep& dh )
private:
  State diff;
  int gen;
  DiffHistoryVtx * backptr;
  int len;
};
```

———-

12.2.10 class SLDiffHistoryVtxRep

— class SLDiffHistoryVtxRep —

```
class SLDiffHistoryVtxRep : public DiffHistoryVtxRep {
public:
  SLDiffHistoryVtxRep(): DiffHistoryVtxRep(),lexComp(0)
  SLDiffHistoryVtxRep(State D,int G,DiffHistoryVtx * ptr,int L,int C):
    DiffHistoryVtxRep(D,G,ptr,L),lexComp(C)
  SLDiffHistoryVtxRep(const SLDiffHistoryVtxRep & dh)
    : DiffHistoryVtxRep(dh), lexComp(dh.lexComp)
  DiffHistoryVtxRep *clone() const
  ~SLDiffHistoryVtxRep()
  int getLexComp() const
  DiffHistoryVtxRep&  operator= ( const DiffHistoryVtxRep & dh )
  Trichotomy betterThan(const DiffHistoryVtxRep & dh) const
  Bool possibleReductionAhead() const
  void printOn(ostream &ostr = cout) const
  inline friend ostream& operator <<
        ( ostream& ostr, const SLDiffHistoryVtxRep& dh )
private:
  int lexComp;
};
```

———-

12.2.11 class WtSLDiffHistoryVtxRep

— class WtSLDiffHistoryVtxRep —

```
class WtSLDiffHistoryVtxRep : public DiffHistoryVtxRep {
public:
  WtSLDiffHistoryVtxRep(): DiffHistoryVtxRep(),lexComp(0), wtdiff(0)
  WtSLDiffHistoryVtxRep(State D,int G,DiffHistoryVtx * ptr,int L,
                        int C, int WD):
    DiffHistoryVtxRep(D,G,ptr,L),lexComp(C), wtdiff(WD)
  WtSLDiffHistoryVtxRep(const WtSLDiffHistoryVtxRep & dh)
    : DiffHistoryVtxRep(dh), lexComp(dh.lexComp), wtdiff(dh.wtdiff)
  DiffHistoryVtxRep *clone() const
  ~WtSLDiffHistoryVtxRep()
  int getLexComp() const
  int getWtDiff() const
  DiffHistoryVtxRep&  operator= ( const DiffHistoryVtxRep & dh )
  Trichotomy betterThan(const DiffHistoryVtxRep & dh) const
  Bool possibleReductionAhead() const
  void printOn(ostream &ostr = cout) const
  inline friend ostream& operator <<
        ( ostream& ostr, const WtSLDiffHistoryVtxRep& dh )
private:
  int lexComp;
  int wtdiff;
};
```

———-

12.2.12 class WtLexDiffHistoryVtxRep

— class WtLexDiffHistoryVtxRep —

```
class WtLexDiffHistoryVtxRep : public DiffHistoryVtxRep {
public:
  WtLexDiffHistoryVtxRep(): DiffHistoryVtxRep(),lexComp(0), wtdiff(0)
  WtLexDiffHistoryVtxRep(State D,int G,DiffHistoryVtx * ptr,int L,
                        int C, int WD):
    DiffHistoryVtxRep(D,G,ptr,L),lexComp(C), wtdiff(WD)
  WtLexDiffHistoryVtxRep(const WtLexDiffHistoryVtxRep & dh)
    : DiffHistoryVtxRep(dh), lexComp(dh.lexComp), wtdiff(dh.wtdiff)
  DiffHistoryVtxRep *clone() const
  ~WtLexDiffHistoryVtxRep()
  int getLexComp() const
  int getWtDiff() const
  DiffHistoryVtxRep&  operator= ( const DiffHistoryVtxRep & dh )
  Trichotomy betterThan(const DiffHistoryVtxRep & dh) const
  Bool possibleReductionAhead() const
  void printOn(ostream &ostr = cout) const
  inline friend ostream& operator <<
        ( ostream& ostr, const WtLexDiffHistoryVtxRep& dh )
private:
```

```
  int lexComp;
  int wtdiff;
};
```

———-

12.3 KB/include/DiffMachine.h

— DiffMachine.h —

```
#include "Word.h"
#include "Set.h"
#include "Vector.h"
#include "FSA.h"
#include "DFSA.h"
#include "DiffMachineRep.h"
```

———-

12.3.1 class DiffMachine

— class DiffMachine —

```
class DiffMachine : public FSA {
  friend class RKBPackage;
  typedef DFSARep::State State;
public:
  DiffMachine( ) : FSA( new DiffMachineRep() )
  DiffMachine( const VectorOf<Chars> genNames ) :
  FSA( new DiffMachineRep( genNames) )
  DiffMachine( const VectorOf<Chars> genNames,const WordOrder & word_order ) :
  FSA( new DiffMachineRep( genNames,word_order) )
  void addDifferencesEqn
       (const Word & w1,const Word & w2,const WordOrder & word_order)
  void closeUnderSubstrings(int mode,const WordOrder & word_order)
  void closeUnderInverses(const WordOrder & word_order)
  Bool rewrite(Word & w) const
  Bool rewrite(Word & w,const WordOrder & word_order) const
  Word getDifference(State s) const
  GroupDFSA wordAcceptor(const WordOrder & word_order) const
  void buildDifferenceMachine(DiffMachine & D,
   const SetOf<Word> & differences, const WordOrder & word_order)
  void rebuildDifferenceMachine(const WordOrder & word_order)
protected:
  DiffMachine( DiffMachineRep * rep ) : FSA((FSARep *)rep) { }
  const DiffMachineRep *look() const {
```

```
    return (DiffMachineRep*)FSA::look(); }
  DiffMachineRep *change() {
    return (DiffMachineRep*)FSA::change(); }
};
```

———-

12.4 KB/include/DiffMachineRep.h

— **DiffMachineRep.h** —

```
#include "Word.h"
#include "Set.h"
#include "Vector.h"
#include "DFSARep.h"
#include "DFSA.h"
#include "DiffHistory.h"
#include "WordOrder.h"
```

———-

12.4.1 class DiffMachineRep

— **class DiffMachineRep** —

```
class DiffMachineRep : public GroupDFSARep {
public:
  DiffMachineRep() : GroupDFSARep()
  DiffMachineRep(const VectorOf<Chars> & genNames) :
    GroupDFSARep("",genNames),
    diffs(1,1)
  DiffMachineRep(const VectorOf<Chars> & genNames,
                 const WordOrder & word_order) :
    GroupDFSARep("",genNames,word_order), diffs(1,1)
  DiffMachineRep( const DiffMachineRep& D ) :
    GroupDFSARep(D), diffs(D.diffs)
  DiffMachineRep & operator = ( const DiffMachineRep & D )
  FSARep *clone() const
  Bool operator == ( const DiffMachineRep& D ) const
  void minimize()
  void readFrom(istream &str = cin);
  void printAccepting(ostream &str = cout) const ;
  void printStates(ostream &str = cout) const ;
  void setNumStates(int numOfStates);
  Word getDifference(State s) const
  void setDifference(State s,Word w)
```

```
  void addDifferencesEqn
       (const Word & w1,const Word & w2,const WordOrder & word_order);
  void closeUnderSubstrings(int mode,const WordOrder & word_order);
  void closeUnderInverses(const WordOrder & word_order);
  Bool rewrite(Word & w) const;
  Bool rewrite(Word & w,const WordOrder & word_order) const;
  GroupDFSA wordAcceptor(const WordOrder & word_order) const;
  void DiffMachineRep::buildDifferenceMachine(DiffMachineRep & D,
       const SetOf<Word> & differences,const WordOrder & word_order) const;
  void DiffMachineRep::rebuildDifferenceMachine(const WordOrder & word_order);
  void write( ostream& ostr ) const
  void read( istream& istr )
private:
  VectorOf<Word> diffs;
  void buildWordAcceptor(GroupDFSARep & Wp,const WordOrder & word_order) const;
  GroupDFSA convertToGroupDFSA(GroupDFSARep * Wp) const;
  Bool reductionAhead(State d,const AheadInfo & ai,
                      const WordOrder & word_order) const;
};
```

———-

12.5 KB/include/GenMult.h

— **GenMult.h** —

```
#include "Word.h"
#include "Set.h"
#include "Vector.h"
#include "FSA.h"
#include "DFSA.h"
#include "GenMultRep.h"
```

———-

12.5.1 class GenMult

— **class GenMult** —

```
class GenMult : public FSA {
public:
  GenMult( ) : FSA( new GenMultRep() )
  GenMult( const VectorOf<Chars> &  genNames ) :
    FSA( new GenMultRep( genNames) )
  GenMult( const VectorOf<Chars> & genNames, const WordOrder & word_order) :
    FSA( new GenMultRep( genNames,word_order) )
```

```
protected:
  const GenMultRep *look() const
  GenMultRep *change()
};
```

———-

12.6 KB/include/GenMultRep.h

— **GenMultRep.h** —

```
#include "Word.h"
#include "Set.h"
#include "Vector.h"
#include "DFSARep.h"
```

———-

12.6.1 class GenMultRep

— **class GenMultRep** —

```
class GenMultRep : public GroupDFSARep {
public:
  GenMultRep() : GroupDFSARep()
  GenMultRep(const VectorOf<Chars> & genNames) :
    GroupDFSARep("",genNames),
    mults(1,YES)
  GenMultRep(const VectorOf<Chars> & genNames,const WordOrder & word_order) :
    GroupDFSARep("",genNames,word_order),
    mults(1,YES)
  GenMultRep( const GenMultRep& GM ) :
    GroupDFSARep(GM), mults(GM.mults)
  GenMultRep & operator = ( const GenMultRep & GM )
  FSARep *clone() const
  Bool operator == ( const DFSARep& ) const
  void minimize()
  void readFrom(istream &str = cin);
  void printAccepting(ostream &str = cout) const ;
  void printStates(ostream &str = cout) const ;
  void setNumStates(int numOfStates);
  void setMultiplier(State s,int i)
  int getMultiplier(State s)
  void write( ostream& ostr ) const
  void read( istream& istr )
private:
```

```
  VectorOf<int> mults;
};
```

———-

12.7 KB/include/KBMachine.h

— **KBMachine.h** —

```
#include <Integer.h>
#include "Vector.h"
#include "FSA.h"
#include "DFSA.h"
#include "KBMachineRep.h"
```

———-

12.7.1 class KBMachine

— **class KBMachine** —

```
class KBMachine : public FSA {
  friend class RKBPackage;
public:
  KBMachine( ) : FSA( new KBMachineRep() )
  KBMachine( const VectorOf<Chars> & genNames ) :
    FSA( new KBMachineRep( genNames ) )
  KBMachine( const VectorOf<Chars> & genNames,const WordOrder & word_order ) :
    FSA( new KBMachineRep( genNames,word_order) )
  KBMachine( const VectorOf<Chars> & genNames,const WordOrder & word_order,
          int numOfRules) :
    FSA( new KBMachineRep( genNames,word_order,numOfRules) )
  KBMachine( const VectorOf<Chars> & genNames,const WordOrder & word_order,
        int numOfRules, int numOfStates) :
    FSA( new KBMachineRep( genNames,numOfRules,numOfStates) )
  void oldFormatPrintOn(ostream & str=cout) const
  Bool rewrite(Word & w) const
protected:
  KBMachine( KBMachineRep * rep ) : FSA((FSARep *)rep)
  const KBMachineRep *look() const
  KBMachineRep *change()
};
```

———-

12.8 KB/include/KBMachineRep.h

— **KBMachineRep.h** —

```
#include <Integer.h>

#include "Word.h"
#include "Vector.h"
#include "DFSARep.h"
```

———-

12.8.1 class KBMachineRep

— **class KBMachineRep** —

```
class KBMachineRep : public GroupDFSARep {
public:
  KBMachineRep() :
    GroupDFSARep(),
    numRules(0),
    lhs(0),
    rhs(0),
    length(1,1),
    ruleIndex(1,1)

  KBMachineRep(const VectorOf<Chars> & genNames) :
    GroupDFSARep("",genNames),
    numRules(0),
    lhs(1),
    rhs(1),
    length(1,1),
    ruleIndex(1,1)

  KBMachineRep(const VectorOf<Chars> & genNames,const WordOrder & word_order) :
    GroupDFSARep("",genNames,word_order),
    numRules(0),
    lhs(1),
    rhs(1),
    length(1,1),
    ruleIndex(1,1)

  KBMachineRep(const VectorOf<Chars> & genNames,int numOfRules) :
    GroupDFSARep("",genNames),
    numRules(numOfRules),
    lhs(numRules+1),
    rhs(numRules+1),
    length(1,1),
```

```
  ruleIndex(1,1)

KBMachineRep(const VectorOf<Chars> & genNames,
                          const WordOrder & word_order,int numOfRules) :
  GroupDFSARep("",genNames,word_order),
  numRules(numOfRules),
  lhs(numRules+1),
  rhs(numRules+1),
  length(1,1),
  ruleIndex(1,1)

KBMachineRep(const VectorOf<Chars> & genNames,int numOfRules,int numOfStates)
     : GroupDFSARep("",genNames,numOfStates,1),
  numRules(numOfRules),
  lhs(numRules+1),
  rhs(numRules+1),
  length(numOfStates+1,1),
  ruleIndex(numOfStates+1,1)

KBMachineRep(const VectorOf<Chars> & genNames,const WordOrder & word_order,
    int numOfRules,int numOfStates)
     : GroupDFSARep("",genNames,word_order,numOfStates,1),
  numRules(numOfRules),
  lhs(numRules+1),
  rhs(numRules+1),
  length(numOfStates+1,1),
  ruleIndex(numOfStates+1,1)

~KBMachineRep()
KBMachineRep( const KBMachineRep& K ) :
  GroupDFSARep(K), numRules(K.numRules), lhs(K.lhs), rhs(K.rhs),
  length(K.length), ruleIndex(K.ruleIndex)

KBMachineRep & operator = ( const KBMachineRep & K )

FSARep *clone() const
Bool operator == ( const KBMachineRep& K ) const
Bool accepts(Word w) const
Bool rejectsInState(Word w, int& state) const
Bool nextAcceptedWord(Word& w) const
void minimize()
void printOn(ostream &str = cout) const ;
void oldFormatPrintOn(ostream &str = cout) const ;
void setNumStates(int i);
void setNumRules(int i)
int getNumRules()
int getLength(State s) const
void setLength(State s,int i)
int getRuleIndex(State s) const
void setRuleIndex(State s,int i)
```

```
  Word getLHS(int i)
  Word getRHS(int i)
  void setLHS(int i,Word w)
  void setRHS(int i,Word w)
  Bool  rewrite(Word & w) const;
  void write( ostream& ostr ) const
  void read( istream& istr )
private:
  int numRules;
  VectorOf<Word> lhs;
  VectorOf<Word> rhs;
  VectorOf<int> length;
  VectorOf<int> ruleIndex;
};
```

———-

12.9 KB/include/KBmagPackage.h

— **KBmagPackage.h** —

```
#include <iomanip.h>

#include "Vector.h"
#include "Chars.h"
#include "Set.h"
#include "Word.h"
#include "DFSA.h"
#include "DiffMachine.h"
#include "GenMult.h"
#include "WordOrder.h"

extern "C" {
  char* tempnam(const char*,const char *);

}
```

———-

12.9.1 class KBmagPackage

— **class KBmagPackage** —

```
class KBmagPackage {
public:
  KBmagPackage(const VectorOf<Chars>& genNames, const SetOf<Word>& rels,
          const WordOrder & word_order,int tidyint) ;
```

```
  KBmagPackage(const VectorOf<Chars>& genNames, const SetOf<Word>& rels,
          const WordOrder & word_order) ;
  KBmagPackage(const VectorOf<Chars>& genNames, const SetOf<Word>& rels);
  ~KBmagPackage();
  Bool sanityCheck() const
  Trichotomy autgroup();
  Trichotomy gpaxioms();
  Trichotomy gpcheckmult();
  Trichotomy gpgenmult(Bool eqcheck=YES);
  Trichotomy gpmakefsa(Bool eqcheck=YES);
  Trichotomy gpwa();
  Trichotomy kbprog (int haltingfactor=100,int tidyint=20,int maxeqns=200,
         int maxstates=1000);
  Chars getName() const
  Bool isProvedAutomatic( ) const
  Bool isProvedConfluent( ) const
  Chars getGeneratorName(Generator g) const
  GroupDFSA wordAcceptor();
  GenMult generalMultiplier();
  DiffMachine differenceMachine(int i);
  void setWordAcceptor(const GroupDFSA & WA);
  void setGeneralMultiplier(const GenMult & GM);
  void setDifferenceMachine(const DiffMachine & Diff,int i);
  GroupDFSARep wordAcceptorRep();
  GenMultRep generalMultiplierRep();
  DiffMachineRep differenceMachineRep(int i);
  void setWordAcceptor(const GroupDFSARep & WARep);
  void setGeneralMultiplier(const GenMultRep & GMRep);
  void setDifferenceMachine(const DiffMachineRep & DiffRep,int i);
  Bool minimize(DFSA & D);
  Bool minimize(GroupDFSA & D);
  Bool minimize(DFSARep & D);
  Bool minimize(GroupDFSARep & D);
  Bool gpcomp(GroupDFSA & D1,GroupDFSA & D2,GroupDFSA & D3);
  Bool gpcomp(GroupDFSARep & D1,GroupDFSARep & D2,GroupDFSARep & D3);
private:
  int numOfSymbols;
  int tidyInterval;
  Chars problemName;
  VectorOf<Chars> generatorNames;
  SetOf<Word> relators;
  WordOrder order;
  Bool error;
  Bool provedAutomatic;
  Bool provedConfluent;
  Chars cdbin;
  Bool findInputFile(const Chars & fname);
  void createRWSFile();
  Word readWord(istream& istr);
  void printWord(ostream& str,const Word& w);
```

```
};
```

———-

12.10 KB/include/RKBPackage.h

— **RKBPackage.h** —

```
#include <iomanip.h>

#include "BlackBox.h"
#include "Vector.h"
#include "Chars.h"
#include "Set.h"
#include "Word.h"
#include "DiffMachine.h"
#include "KBMachine.h"
#include "MagnusHome.h"
#include "WordOrder.h"
```

———-

12.10.1 class RKBPackage

— **class RKBPackage** —

```
class RKBPackage {
public:
  RKBPackage(const VectorOf<Chars>& genNames, const SetOf<Word>& relators)
  : theRKBPackage(Chars("cd ")
      + MagnusHome::magnusHome()
      + "/back_end/black_boxes/rkbp/bin; ./rkbp"
      ),
    index(1+genNames.length()),index_inv(1+genNames.length()),
    genNumber(2*genNames.length())
  RKBPackage(const VectorOf<Chars>& genNames, const SetOf<Word>& relators,
    const WordOrder & word_order)
  : theRKBPackage(Chars("cd ")
            + MagnusHome::magnusHome()
            + "/back_end/black_boxes/rkbp/bin; ./rkbp"
            ),
     index(1+genNames.length()),index_inv(1+genNames.length()),
     genNumber(2*genNames.length())
  ~RKBPackage();
  Chars getName() const
  void runKB(int MaxLen, int numIterations, int saveLHSLen, int saveRHSLen);
```

```
  float* growthRate(int maxLen, int numTrials, int seed);
  void saveToFiles();
  void printRules(ostream& str = cout);
  void readRule(istream& str, Word& left, Word& right);
  void oneOffRewrite(Word& w);
  void enterRewriteMode();
  void quitRewriteMode();
  void rewrite(Word& w);
  SetOf<Word> wordDifferences();
  DiffMachine differenceMachine(const SetOf<Word> & differences);
  DiffMachine differenceMachine()
  KBMachine KnuthBendixMachine();
  VectorOf<Chars> getGeneratorNames() const
  const char* rulesFilename() const
  int currentNumOfRules() const
  Bool isProvedConfluent() const
  Bool rulesReduced(Word * lhs,Word * rhs,int numOfrules);
  Bool sanityCheck() const
  Chars getGeneratorName(Generator g) const
private:
  Bool error;
  Bool runningProcess;
  Bool rewriteMode;
  Bool filesOutOfDate;
  Bool provedConfluent;
  int numRules;
  int lenLeftMin;
  int lenLeftMax;
  int lenLeftTotal;
  int lenRightMin;
  int lenRightMax;
  int lenRightTotal;
  char problemName[64];
  char inFileName[64];
  char outFileName[64];
  char rulesFileName[64];
  char subsysFileName[64];
  char sysFileName[64];
  VectorOf<Chars> generatorNames;
  WordOrder order;
  VectorOf<int> index;
  VectorOf<int> index_inv;
  VectorOf<int> genNumber;
  BlackBox theRKBPackage;
  void initialize(const VectorOf<Chars>& genNames, const SetOf<Word>& relators,
                  const WordOrder & word_order);
  void restartProcess();
  void haltProcess();
  void createSystemFile();
  void setWeightsAndLevels(ofstream & rkbp_system);
```

```
  void createSubsystemFile();
  void createRulesFile(const SetOf<Word>& relators);
  void skipToNextPrompt();
  void skipToImmediatePrompt();
  void printWord(ostream& str,const Word& w);
  Word readWord(istream& str);
  void readSummary();
  float* readProbe(int maxLen);
  DiffMachine convertToDiffMachine(DiffMachineRep * R)const ;
  KBMachine convertToKBMachine(KBMachineRep * R) const;
  void buildDifferenceMachine(DiffMachineRep & D,
                         const SetOf<Word> & differences);
  void buildKnuthBendixMachine(KBMachineRep & K);
};
```

———-

12.11 KB/include/WordOrder.h

— **WordOrder.h** —

```
#include "Word.h"
#include "Vector.h"

#include "Set.h"
#include "DiffHistoryRep.h"
#include "WordOrderRep.h"
typedef int State;
```

———-

12.11.1 class WordOrder

— **class WordOrder** —

```
class WordOrder : public ObjectOf<WordOrderRep> {
public:
  WordOrder() : ObjectOf< WordOrderRep > (new WordOrderRep())
  WordOrder(const Chars & oType) :
    ObjectOf< WordOrderRep > (
    (oType==Chars("ShortLex")? new ShortLexRep() :
      (oType==Chars("WtShortLex")? new WtShortLexRep():
        (oType==Chars("WtLex")? new WtLexRep(): new WordOrderRep()))))
  WordOrder(const Chars & oType, int numOfSymbols) :
    ObjectOf< WordOrderRep > (
    (oType==Chars("ShortLex")? new ShortLexRep(numOfSymbols) :
```

```
      (oType==Chars("WtShortLex")? new WtShortLexRep(numOfSymbols):
        (oType==Chars("WtLex")? new
          WtLexRep(numOfSymbols): new WordOrderRep(numOfSymbols)))))
  WordOrder(const Chars & oType,const VectorOf<int> & v) :
    ObjectOf< WordOrderRep > (
    (oType==Chars("ShortLex")? new ShortLexRep(v) :
      (oType==Chars("WtShortLex")? new WtShortLexRep(v):
        (oType==Chars("WtLex")? new WtLexRep(v): new WordOrderRep(v)))))
  WordOrder(const Chars & oType,const VectorOf<int> & o,
            const VectorOf<int> & w)
   : ObjectOf< WordOrderRep > (
     (oType==Chars("WtShortLex")? new WtShortLexRep(o,w):
        (oType==Chars("WtLex")? new WtLexRep(o,w): new WordOrderRep(o))))
  int signature(const Word & w1,const Word & w2) const
  int signature(Generator g,Generator h) const
  int signature(int i,int j) const
  void balancedEquationFromRelator(const Word & w,Word & lhs,Word & rhs) const
  int historyBound(const VectorOf<Word> & diffs) const
  DiffHistoryRep * buildDiffHistoryRep() const
  DiffHistoryRep * buildDiffHistoryRep(State d,int g,int h) const
  DiffHistoryRep * update
     (const DiffHistoryRep & dh,State d,int g,int h,const Word & wd) const
  Bool reduction(const DiffHistoryRep & dh,int g,int h) const
  Bool possibleReduction(const AheadInfoRep & ai,int g) const
  AheadInfoRep *  update(const AheadInfoRep & ai,int g) const
  DiffHistoryVtxRep * buildDiffHistoryVtxRep() const
  DiffHistoryVtxRep * buildDiffHistoryVtxRep(State d,int g,int h) const
  DiffHistoryVtxRep * update (const DiffHistoryVtxRep & dh,State d,
                              int g,int h,DiffHistoryVtx * ptr) const
  Bool reduction(const DiffHistoryVtxRep & dh,int g,int h) const
  Bool possibleReductionAhead(const DiffHistoryVtxRep & dh,int g) const
  Chars getOrderType() const
  int getNumSymbols() const
  int getSymbolIndex(int i) const
  Generator getWeight(int i) const
  int getWeight(const Word & w) const
  Generator getSymbol(int i) const
  int getPosition(Generator g) const
  int selfInverse(Generator g) const
  Word inverse(const Word & w) const
protected:
  typedef ObjectOf<WordOrderRep> R;
  WordOrder( WordOrderRep *p ) : R(p)
};
```

12.12 KB/include/WordOrderRep.h

— **WordOrderRep.h** —

```
#include "Word.h"
#include "Vector.h"
#include "RefCounter.h"

#include "Set.h"
#include "DiffHistoryRep.h"
typedef int State;
```

———-

12.12.1 class WordOrderRep

— **class WordOrderRep** —

```
class WordOrderRep : public RefCounter {
public:
  WordOrderRep()
  WordOrderRep(const Chars & oType ) : orderType(oType),numSyms(0), order(0),
    posn(0), invposn(0)
  WordOrderRep(const VectorOf<int> & o) : numSyms(o.length()), order(o),
    posn(o.length()), invposn(o.length())
  WordOrderRep(int numOfSymbols) : numSyms(numOfSymbols), order(0),
    posn(0), invposn(0)
  WordOrderRep(const Chars & oType, int numOfSymbols) :
    orderType(oType), numSyms(numOfSymbols), order(0), posn(0), invposn(0)
  WordOrderRep(const Chars & oType, const VectorOf<int> & o) :
    orderType(oType), numSyms(o.length()), order(o), posn(o.length()),
      invposn(o.length())
  WordOrderRep (const WordOrderRep & word_order)
  virtual WordOrderRep *clone() const
  ~WordOrderRep()
  virtual int signature(const Word & w1,const Word & w2) const
  virtual int signature(int i, int j) const
  virtual int signature(Generator g,Generator h) const
  virtual void balancedEquationFromRelator
                      (const Word & w,Word & lhs,Word & rhs) const
  virtual int historyBound(const VectorOf<Word> & diffs) const
  virtual DiffHistoryRep * buildDiffHistoryRep() const
  virtual DiffHistoryRep * buildDiffHistoryRep(State d,int g,int h) const
  virtual DiffHistoryRep * update
     (const DiffHistoryRep & dh,State d,int g,int h,const Word & wd) const
  virtual Bool reduction(const DiffHistoryRep & dh,int g,int h) const
  virtual Bool possibleReduction(const AheadInfoRep & ai,int g) const
  virtual AheadInfoRep *  update(const AheadInfoRep & ai,int g) const
```

```
  virtual DiffHistoryVtxRep * buildDiffHistoryVtxRep() const
  virtual DiffHistoryVtxRep
      * buildDiffHistoryVtxRep(State d,int g,int h) const
  virtual DiffHistoryVtxRep * update
     (const DiffHistoryVtxRep & dh,State d, int g, int h, DiffHistoryVtx * ptr)
           const
  virtual Bool reduction(const DiffHistoryVtxRep & dh,int g,int h) const
  virtual Bool possibleReductionAhead(const DiffHistoryVtxRep & dh,int g) const
  Chars getOrderType() const
  int getNumSymbols() const
  int getSymbolIndex(int i) const
  Generator getSymbol(int i) const
  int getPosition(Generator g) const
  Word inverse(const Word & w) const
  Bool selfInverse(Generator g) const
  virtual int getWeight(int i) const
  virtual int getWeight(Generator g) const
  virtual int getWeight(const Word & w) const
protected:
  VectorOf<int> order;
  VectorOf<int> posn;
  VectorOf<int> invposn;
  int numSyms;
private:
  Chars orderType;
};
```

———-

12.12.2 class ShortLexRep

— **class ShortLexRep** —

```
class ShortLexRep : public WordOrderRep {
public:
  ShortLexRep() : WordOrderRep("ShortLex")
  ShortLexRep(const VectorOf<int> &  o) : WordOrderRep("ShortLex",o)
  ShortLexRep(int numOfSymbols) : WordOrderRep("ShortLex",numOfSymbols)
  ShortLexRep( const ShortLexRep& wsl ) :
    WordOrderRep(wsl)
  WordOrderRep *clone() const
  int signature(const Word & w1,const Word & w2) const
  int signature(int i, int j) const
  int signature(Generator g,Generator h) const
  void balancedEquationFromRelator(const Word & w,Word & lhs,Word & rhs) const
  int historyBound(const VectorOf<Word> & diffs) const
  DiffHistoryRep * buildDiffHistoryRep() const
  DiffHistoryRep * buildDiffHistoryRep(State d,int g,int h) const
```

```
  DiffHistoryRep * update
    (const DiffHistoryRep & dh,State d,int g,int h,const Word & wd) const
  Bool reduction(const DiffHistoryRep & dh,int g,int h) const
  Bool possibleReduction(const AheadInfoRep & ai,int g) const
  AheadInfoRep *  update(const AheadInfoRep & ai,int g) const
  DiffHistoryVtxRep * buildDiffHistoryVtxRep() const
  DiffHistoryVtxRep * buildDiffHistoryVtxRep(State d,int g,int h) const
  DiffHistoryVtxRep * update
    (const DiffHistoryVtxRep & dh,State d,int g,int h, DiffHistoryVtx * ptr)
       const
  Bool reduction(const DiffHistoryVtxRep & dh,int g,int h) const
  Bool possibleReduction(const DiffHistoryVtxRep & dh,int g) const
  int getWeight(int i) const
  int getWeight(Generator g) const
  int getWeight(const Word & w) const
};
```

———-

12.12.3 class WtShortLexRep

— class WtShortLexRep —

```
class WtShortLexRep : public WordOrderRep {
public:
  WtShortLexRep() : WordOrderRep("WtShortLex")
  WtShortLexRep(int numOfSymbols) :
    WordOrderRep("WtShortLex",numOfSymbols), weight(0)
  WtShortLexRep(const VectorOf<int>  & w) :
    WordOrderRep("WtShortLex",w.length()), weight(w)
  WtShortLexRep(const VectorOf<int> & o,const VectorOf<int> & w) :
    WordOrderRep("WtShortLex",o) , weight(w)
  WtShortLexRep( const WtShortLexRep& wsl ) :
    WordOrderRep(wsl), weight(wsl.weight)
  WordOrderRep *clone() const
  int signature(const Word & w1,const Word & w2) const
  int signature(int i, int j) const
  int signature(Generator g,Generator h) const
  void balancedEquationFromRelator(const Word & w,Word & lhs,Word & rhs) const
  int historyBound(const VectorOf<Word> & diffs) const
  DiffHistoryRep * buildDiffHistoryRep() const
  DiffHistoryRep * buildDiffHistoryRep(State d,int g,int h) const
  DiffHistoryRep * update
    (const DiffHistoryRep & dh,State d,int g,int h,const Word & wd) const
  Bool reduction(const DiffHistoryRep & dh,int g,int h) const
  Bool possibleReduction(const AheadInfoRep & ai,int g) const
  AheadInfoRep * update(const AheadInfoRep & ai,int g) const
  DiffHistoryVtxRep * buildDiffHistoryVtxRep() const
```

```
  DiffHistoryVtxRep * buildDiffHistoryVtxRep(State d,int g,int h) const
  DiffHistoryVtxRep * update
    (const DiffHistoryVtxRep & dh,State d,int g,int h, DiffHistoryVtx * ptr)
       const
  Bool reduction(const DiffHistoryVtxRep & dh,int g,int h) const
  Bool possibleReductionAhead(const DiffHistoryVtxRep & dh,int g) const
private:
  VectorOf<int> weight;
};
```

———-

12.12.4 class WtLexRep

— class WtLexRep —

```
class WtLexRep : public WordOrderRep {
public:
  WtLexRep() : WordOrderRep("WtLex")
  WtLexRep(int numOfSymbols) :
    WordOrderRep("WtLex",numOfSymbols), weight(0)
  WtLexRep(const VectorOf<int>  & w) :
    WordOrderRep("WtLex",w.length()), weight(w)
  WtLexRep(const VectorOf<int> & o,const VectorOf<int> & w) :
    WordOrderRep("WtLex",o) , weight(w)
  WtLexRep( const WtLexRep& wsl ) :
    WordOrderRep(wsl), weight(wsl.weight)
  WordOrderRep *clone() const
  int signature(const Word & w1,const Word & w2) const
  int signature(int i, int j) const
  int signature(Generator g, Generator h) const
  void balancedEquationFromRelator(const Word & w,Word & lhs,Word & rhs) const
  int historyBound(const VectorOf<Word> & diffs) const
  DiffHistoryRep * buildDiffHistoryRep() const
  DiffHistoryRep * buildDiffHistoryRep(State d,int g,int h) const
  DiffHistoryRep * update
    (const DiffHistoryRep & dh,State d,int g,int h,const Word & wd) const
  Bool reduction(const DiffHistoryRep & dh,int g,int h) const
  Bool possibleReduction(const AheadInfoRep & ai,int g) const
  AheadInfoRep * update(const AheadInfoRep & ai,int g) const
  DiffHistoryVtxRep * buildDiffHistoryVtxRep() const
  DiffHistoryVtxRep * buildDiffHistoryVtxRep(State d,int g,int h) const
  DiffHistoryVtxRep * update
    (const DiffHistoryVtxRep & dh,State d,int g,int h, DiffHistoryVtx * ptr)
      const
  Bool reduction(const DiffHistoryVtxRep & dh,int g,int h) const
  Bool possibleReductionAhead(const DiffHistoryVtxRep & dh,int g) const
  int getWeight(int i) const
```

```
  int getWeight(Generator g) const
  int getWeight(const Word & w) const
private:
  VectorOf<int> weight;
};
```

———-

12.12.5 class InvPairWreathRep

— class InvPairWreathRep —

```
class InvPairWreathRep : public WordOrderRep {
public:
  InvPairWreathRep() : WordOrderRep("InvPairWreath")
  InvPairWreathRep(const VectorOf<int> & o) : WordOrderRep("InvPairWreath",o)
  int signature(int g, int h) const
};
```

———-

13 The libg++ classes

13.1 libg++/include/AllocRing.h

— AllocRing.h —

———-

13.1.1 class AllocRing

— class AllocRing —

```
class AllocRing
{
```

———-

13.1.2 struct AllocQNode

— struct AllocQNode —

```
  struct AllocQNode
  {
    void*  ptr;
    int    sz;
  };

  AllocQNode* nodes;
  int         n;
  int         current;
  int         find(void* p);
public:
              AllocRing(int max);
             ~AllocRing();
  void*       alloc(int size);
  int         contains(void* ptr);
  void        clear();
  void        free(void* p);
};
```

13.2 libg++/include/builtin.h

— builtin.h —

```
#define _builtin_h 1

#include <stddef.h>
#include <std.h>
#include <cmath>

typedef void (*one_arg_error_handler_t)(const char*);
typedef void (*two_arg_error_handler_t)(const char*, const char*);
long         gcd(long, long);
long         lg(unsigned long);
double       pow(double, long);
long         pow(long, long);
extern "C" double       start_timer();
extern "C" double       return_elapsed_time(double last_time = 0.0);
char*        dtoa(double x, char cvt = 'g', int width = 0, int prec = 6);
unsigned int hashpjw(const char*);
unsigned int multiplicativehash(int);
unsigned int foldhash(double);
extern void default_one_arg_error_handler(const char*)
    __attribute__ ((noreturn));
extern void default_two_arg_error_handler(const char*, const char*)
    __attribute__ ((noreturn));
```

```
extern two_arg_error_handler_t lib_error_handler;
extern two_arg_error_handler_t
       set_lib_error_handler(two_arg_error_handler_t f);
```

———-

13.3 libg++/include/Integer.h

— **Integer.h** —

———-

13.3.1 struct IntRep

— **struct IntRep** —

```
struct IntRep
{
  unsigned short  len;
  unsigned short  sz;
  short           sgn;
  unsigned short  s[1];
};

#define STATIC_IntRep(rep) ((rep)->sz==0)
extern IntRep*  Ialloc(IntRep*, const unsigned short *, int, int, int);
extern IntRep*  Icalloc(IntRep*, int);
extern IntRep*  Icopy_ulong(IntRep*, unsigned long);
extern IntRep*  Icopy_long(IntRep*, long);
extern IntRep*  Icopy(IntRep*, const IntRep*);
extern IntRep*  Iresize(IntRep*, int);
extern IntRep*  add(const IntRep*, int, const IntRep*, int, IntRep*);
extern IntRep*  add(const IntRep*, int, long, IntRep*);
extern IntRep*  multiply(const IntRep*, const IntRep*, IntRep*);
extern IntRep*  multiply(const IntRep*, long, IntRep*);
extern IntRep*  lshift(const IntRep*, long, IntRep*);
extern IntRep*  lshift(const IntRep*, const IntRep*, int, IntRep*);
extern IntRep*  bitop(const IntRep*, const IntRep*, IntRep*, char);
extern IntRep*  bitop(const IntRep*, long, IntRep*, char);
extern IntRep*  power(const IntRep*, long, IntRep*);
extern IntRep*  div(const IntRep*, const IntRep*, IntRep*);
extern IntRep*  mod(const IntRep*, const IntRep*, IntRep*);
extern IntRep*  div(const IntRep*, long, IntRep*);
extern IntRep*  mod(const IntRep*, long, IntRep*);
extern IntRep*  compl(const IntRep*, IntRep*);
```

```
extern IntRep*  abs(const IntRep*, IntRep*);
extern IntRep*  negate(const IntRep*, IntRep*);
extern IntRep*  pow(const IntRep*, long);
extern IntRep*  gcd(const IntRep*, const IntRep* y);
extern int      compare(const IntRep*, const IntRep*);
extern int      compare(const IntRep*, long);
extern int      ucompare(const IntRep*, const IntRep*);
extern int      ucompare(const IntRep*, long);
extern char*    Itoa(const IntRep* x, int base = 10, int width = 0);
extern char*    cvtItoa(const IntRep* x, char* fmt, int& fmtlen, int base,
                        int showbase, int width, int align_right,
                        char fillchar, char Xcase, int showpos);
extern IntRep*  atoIntRep(const char* s, int base = 10);
extern long     Itolong(const IntRep*);
extern int      Iislong(const IntRep*);
extern long     lg(const IntRep*);
extern IntRep _ZeroRep, _OneRep, _MinusOneRep;
```

———-

13.3.2 class Integer

— **class Integer** —

```
class Integer
{
protected:
  IntRep*         rep;
public:
                  Integer();
                  Integer(int);
                  Integer(long);
                  Integer(unsigned long);
                  Integer(IntRep*);
                  Integer(const Integer&);
                  ~Integer();
  Integer&        operator =  (const Integer&);
  Integer&        operator =  (long);
  Integer&        operator ++ ();
  Integer&        operator -- ();
  void            negate();
  void            abs();
  void            complement();
  Integer&        operator += (const Integer&);
  Integer&        operator -= (const Integer&);
  Integer&        operator *= (const Integer&);
  Integer&        operator /= (const Integer&);
  Integer&        operator %= (const Integer&);
```

```
  Integer&        operator <<=(const Integer&);
\begin{chunk}{Integer&        operator >>=(const Integer}
  Integer&        operator &= (const Integer&);
  Integer&        operator |= (const Integer&);
  Integer&        operator ^= (const Integer&);
  Integer&        operator += (long);
  Integer&        operator -= (long);
  Integer&        operator *= (long);
  Integer&        operator /= (long);
  Integer&        operator %= (long);
  Integer&        operator <<=(long);
\begin{chunk}{Integer&        operator >>=(lon}
  Integer&        operator &= (long);
  Integer&        operator |= (long);
  Integer&        operator ^= (long);
  friend long     lg (const Integer&);
  friend double   ratio(const Integer& x, const Integer& y);
  friend Integer  gcd(const Integer&, const Integer&);
  friend int      even(const Integer&);
  friend int      odd(const Integer&);
  friend int      sign(const Integer&);
  friend void     (setbit)(Integer& x, long b);
  friend void     clearbit(Integer& x, long b);
  friend int      testbit(const Integer& x, long b);
  friend void     abs(const Integer& x, Integer& dest);
  friend void     negate(const Integer& x, Integer& dest);
  friend void     complement(const Integer& x, Integer& dest);
  friend int      compare(const Integer&, const Integer&);
  friend int      ucompare(const Integer&, const Integer&);
  friend void     add(const Integer& x, const Integer& y, Integer& dest);
  friend void     sub(const Integer& x, const Integer& y, Integer& dest);
  friend void     mul(const Integer& x, const Integer& y, Integer& dest);
  friend void     div(const Integer& x, const Integer& y, Integer& dest);
  friend void     mod(const Integer& x, const Integer& y, Integer& dest);
  friend void     divide(const Integer& x, const Integer& y,
                         Integer& q, Integer& r);
  friend void     and(const Integer& x, const Integer& y, Integer& dest);
  friend void     or(const Integer& x, const Integer& y, Integer& dest);
  friend void     xor(const Integer& x, const Integer& y, Integer& dest);
  friend void     lshift(const Integer& x, const Integer& y, Integer& dest);
  friend void     rshift(const Integer& x, const Integer& y, Integer& dest);
  friend void     pow(const Integer& x, const Integer& y, Integer& dest);
  friend int      compare(const Integer&, long);
  friend int      ucompare(const Integer&, long);
  friend void     add(const Integer& x, long y, Integer& dest);
  friend void     sub(const Integer& x, long y, Integer& dest);
  friend void     mul(const Integer& x, long y, Integer& dest);
  friend void     div(const Integer& x, long y, Integer& dest);
  friend void     mod(const Integer& x, long y, Integer& dest);
  friend void     divide(const Integer& x, long y, Integer& q, long& r);
```

```
  friend void     and(const Integer& x, long y, Integer& dest);
  friend void     or(const Integer& x, long y, Integer& dest);
  friend void     xor(const Integer& x, long y, Integer& dest);
  friend void     lshift(const Integer& x, long y, Integer& dest);
  friend void     rshift(const Integer& x, long y, Integer& dest);
  friend void     pow(const Integer& x, long y, Integer& dest);
  friend int      compare(long, const Integer&);
  friend int      ucompare(long, const Integer&);
  friend void     add(long x, const Integer& y, Integer& dest);
  friend void     sub(long x, const Integer& y, Integer& dest);
  friend void     mul(long x, const Integer& y, Integer& dest);
  friend void     and(long x, const Integer& y, Integer& dest);
  friend void     or(long x, const Integer& y, Integer& dest);
  friend void     xor(long x, const Integer& y, Integer& dest);
  int             fits_in_long() const { return Iislong(rep); }
  int             fits_in_double() const;
  long      as_long() const { return Itolong(rep); }
  double    as_double() const;
  friend char*    Itoa(const Integer& x, int base = 10, int width = 0);
  friend Integer  atoI(const char* s, int base = 10);
  void      printon(ostream& s, int base = 10, int width = 0) const;
  friend istream& operator >> (istream& s, Integer& y);
  friend ostream& operator << (ostream& s, const Integer& y);
  int             initialized() const;
  void   error(const char* msg) const;
  int             OK() const;
};

  int      operator == (const Integer&, const Integer&);
  int      operator == (const Integer&, long);
  int      operator != (const Integer&, const Integer&);
  int      operator != (const Integer&, long);
  int      operator <  (const Integer&, const Integer&);
  int      operator <  (const Integer&, long);
  int      operator <= (const Integer&, const Integer&);
  int      operator <= (const Integer&, long);
  int      operator >  (const Integer&, const Integer&);
  int      operator >  (const Integer&, long);
  int      operator >= (const Integer&, const Integer&);
  int      operator >= (const Integer&, long);
  Integer  operator -  (const Integer&);
  Integer  operator ~  (const Integer&);
  Integer  operator +  (const Integer&, const Integer&);
  Integer  operator +  (const Integer&, long);
  Integer  operator +  (long, const Integer&);
  Integer  operator -  (const Integer&, const Integer&);
  Integer  operator -  (const Integer&, long);
  Integer  operator -  (long, const Integer&);
  Integer  operator *  (const Integer&, const Integer&);
  Integer  operator *  (const Integer&, long);
```

```
  Integer  operator *  (long, const Integer&);
  Integer  operator /  (const Integer&, const Integer&);
  Integer  operator /  (const Integer&, long);
  Integer  operator %  (const Integer&, const Integer&);
  Integer  operator %  (const Integer&, long);
  Integer  operator << (const Integer&, const Integer&);
  Integer  operator << (const Integer&, long);
  Integer  operator >> (const Integer&, const Integer&);
  Integer  operator >> (const Integer&, long);
  Integer  operator &  (const Integer&, const Integer&);
  Integer  operator &  (const Integer&, long);
  Integer  operator &  (long, const Integer&);
  Integer  operator |  (const Integer&, const Integer&);
  Integer  operator |  (const Integer&, long);
  Integer  operator |  (long, const Integer&);
  Integer  operator ^  (const Integer&, const Integer&);
  Integer  operator ^  (const Integer&, long);
  Integer  operator ^  (long, const Integer&);
  Integer  abs(const Integer&);
  Integer  sqr(const Integer&);
  Integer  pow(const Integer& x, const Integer& y);
  Integer  pow(const Integer& x, long y);
  Integer  Ipow(long x, long y);
extern char*    dec(const Integer& x, int width = 0);
extern char*    oct(const Integer& x, int width = 0);
extern char*    hex(const Integer& x, int width = 0);
extern Integer  sqrt(const Integer&);
extern Integer  lcm(const Integer& x, const Integer& y);
typedef Integer IntTmp;
inline Integer::Integer() :rep(&_ZeroRep)
inline Integer::Integer(IntRep* r) :rep(r)
inline Integer::Integer(int y) :rep(Icopy_long(0, (long)y))
inline Integer::Integer(long y) :rep(Icopy_long(0, y))
inline Integer::Integer(unsigned long y) :rep(Icopy_ulong(0, y))
inline Integer::Integer(const Integer&  y) :rep(Icopy(0, y.rep))
inline Integer::~Integer()
inline Integer&  Integer::operator = (const Integer&  y)
inline Integer& Integer::operator = (long y)
inline int Integer::initialized() const
inline int compare(const Integer& x, const Integer& y)
inline int ucompare(const Integer& x, const Integer& y)
inline int compare(const Integer& x, long y)
inline int ucompare(const Integer& x, long y)
inline int compare(long x, const Integer& y)
inline int ucompare(long x, const Integer& y)
inline void  add(const Integer& x, const Integer& y, Integer& dest)
inline void  sub(const Integer& x, const Integer& y, Integer& dest)
inline void  mul(const Integer& x, const Integer& y, Integer& dest)
inline void  div(const Integer& x, const Integer& y, Integer& dest)
inline void  mod(const Integer& x, const Integer& y, Integer& dest)
```

```
inline void  and(const Integer& x, const Integer& y, Integer& dest)
inline void  or(const Integer& x, const Integer& y, Integer& dest)
inline void  xor(const Integer& x, const Integer& y, Integer& dest)
inline void  lshift(const Integer& x, const Integer& y, Integer& dest)
inline void  rshift(const Integer& x, const Integer& y, Integer& dest)
inline void  pow(const Integer& x, const Integer& y, Integer& dest)
inline void  add(const Integer& x, long y, Integer& dest)
inline void  sub(const Integer& x, long y, Integer& dest)
inline void  mul(const Integer& x, long y, Integer& dest)
inline void  div(const Integer& x, long y, Integer& dest)
inline void  mod(const Integer& x, long y, Integer& dest)
inline void  and(const Integer& x, long y, Integer& dest)
inline void  or(const Integer& x, long y, Integer& dest)
inline void  xor(const Integer& x, long y, Integer& dest)
inline void  lshift(const Integer& x, long y, Integer& dest)
inline void  rshift(const Integer& x, long y, Integer& dest)
inline void  pow(const Integer& x, long y, Integer& dest)
inline void abs(const Integer& x, Integer& dest)
inline void negate(const Integer& x, Integer& dest)
inline void complement(const Integer& x, Integer& dest)
inline void  add(long x, const Integer& y, Integer& dest)
inline void  sub(long x, const Integer& y, Integer& dest)
inline void  mul(long x, const Integer& y, Integer& dest)
inline void  and(long x, const Integer& y, Integer& dest)
inline void  or(long x, const Integer& y, Integer& dest)
inline void  xor(long x, const Integer& y, Integer& dest)
inline int operator == (const Integer&  x, const Integer&  y)
inline int operator == (const Integer&  x, long y)
inline int operator != (const Integer&  x, const Integer&  y)
inline int operator != (const Integer&  x, long y)
inline int operator <  (const Integer&  x, const Integer&  y)
inline int operator <  (const Integer&  x, long y)
inline int operator <= (const Integer&  x, const Integer&  y)
inline int operator <= (const Integer&  x, long y)
inline int operator >  (const Integer&  x, const Integer&  y)
inline int operator >  (const Integer&  x, long y)
inline int operator >= (const Integer&  x, const Integer&  y)
inline int operator >= (const Integer&  x, long y)
inline Integer&  Integer::operator += (const Integer& y)
inline Integer&  Integer::operator += (long y)
inline Integer& Integer::operator ++ ()
inline Integer& Integer::operator -= (const Integer& y)
inline Integer& Integer::operator -= (long y)
inline Integer& Integer::operator -- ()
inline Integer& Integer::operator *= (const Integer& y)
inline Integer& Integer::operator *= (long y)
inline Integer& Integer::operator &= (const Integer& y)
inline Integer& Integer::operator &= (long y)
inline Integer& Integer::operator |= (const Integer& y)
inline Integer& Integer::operator |= (long y)
```

```
inline Integer& Integer::operator ^= (const Integer& y)
inline Integer& Integer::operator ^= (long y)
inline Integer& Integer::operator /= (const Integer& y)
inline Integer& Integer::operator /= (long y)
inline Integer& Integer::operator <<= (const Integer&  y)
inline Integer& Integer::operator <<= (long  y)
\begin{chunk}{line Integer& Integer::operator >>= (const Integer& }
\begin{chunk}{line Integer& Integer::operator >>= (long}
inline void Integer::abs()
inline void Integer::negate()
inline void Integer::complement()
inline int sign(const Integer& x)
inline int even(const Integer& y)
inline int odd(const Integer& y)
inline char* Itoa(const Integer& y, int base, int width)
inline long lg(const Integer& x)
inline Integer  operator +  (const Integer& x, const Integer& y) return r
inline Integer  operator +  (const Integer& x, long y) return r
inline Integer  operator +  (long  x, const Integer& y) return r
inline Integer  operator -  (const Integer& x, const Integer& y) return r
inline Integer  operator -  (const Integer& x, long y) return r
inline Integer  operator -  (long  x, const Integer& y) return r
inline Integer  operator *  (const Integer& x, const Integer& y) return r
inline Integer  operator *  (const Integer& x, long y) return r
inline Integer  operator *  (long  x, const Integer& y) return r
inline Integer sqr(const Integer& x) return r
inline Integer  operator &  (const Integer& x, const Integer& y) return r
inline Integer  operator &  (const Integer& x, long y) return r
inline Integer  operator &  (long  x, const Integer& y) return r
inline Integer  operator |  (const Integer& x, const Integer& y) return r
inline Integer  operator |  (const Integer& x, long y) return r
inline Integer  operator |  (long  x, const Integer& y) return r
inline Integer  operator ^  (const Integer& x, const Integer& y) return r
inline Integer  operator ^  (const Integer& x, long y) return r
inline Integer  operator ^  (long  x, const Integer& y) return r
inline Integer  operator /  (const Integer& x, const Integer& y) return r
inline Integer operator /  (const Integer& x, long y) return r
inline Integer operator %  (const Integer& x, const Integer& y) return r
inline Integer operator %  (const Integer& x, long y) return r
inline Integer operator <<  (const Integer& x, const Integer& y) return r
inline Integer operator <<  (const Integer& x, long y) return r
inline Integer operator >>  (const Integer& x, const Integer& y) return r;
inline Integer operator >>  (const Integer& x, long y) return r
inline Integer pow(const Integer& x, long y) return r
inline Integer Ipow(long x, long y) return r(x)
inline Integer pow(const Integer& x, const Integer& y) return r
inline Integer abs(const Integer& x) return r
inline Integer operator - (const Integer& x) return r
inline Integer operator ~ (const Integer& x) return r
inline Integer  atoI(const char* s, int base) return r
```

```
inline Integer  gcd(const Integer& x, const Integer& y) return r
```

———-

13.4 libg++/include/Integer.hP

— **Integer.hP** —

```
#define I_SHIFT         (sizeof(short) * CHAR_BIT)
#define I_RADIX         ((unsigned long)(1L << I_SHIFT))
#define I_MAXNUM        ((unsigned long)((I_RADIX - 1)))
#define I_MINNUM        ((unsigned long)(I_RADIX >> 1))
#define I_POSITIVE      1
#define I_NEGATIVE      0
#define SHORT_PER_LONG  ((unsigned)(((sizeof(long) + sizeof(short) - 1) / sizeof(short))))
#define CHAR_PER_LONG   ((unsigned)sizeof(long))
#define MINIntRep_SIZE   16
#define MAXIntRep_SIZE   I_MAXNUM
```

———-

13.5 libg++/include/Rational.h

— **Rational.h** —

```
#define _Rational_h 1
#include <Integer.h>
#include <math.h>
```

———-

13.5.1 class Rational

— **class Rational** —

```
class Rational
{
protected:
  Integer          num;
  Integer          den;
  void             normalize();
public:
                   Rational();
                   Rational(double);
                   Rational(int n);
                   Rational(long n);
```

```
                Rational(int n, int d);
                Rational(long n, long d);
                Rational(long n, unsigned long d);
                Rational(unsigned long n, long d);
                Rational(unsigned long n, unsigned long d);
                Rational(const Integer& n);
                Rational(const Integer& n, const Integer& d);
                Rational(const Rational&);
               ~Rational();
Rational&       operator =  (const Rational& y);
friend int      operator == (const Rational& x, const Rational& y);
friend int      operator != (const Rational& x, const Rational& y);
friend int      operator <  (const Rational& x, const Rational& y);
friend int      operator <= (const Rational& x, const Rational& y);
friend int      operator >  (const Rational& x, const Rational& y);
friend int      operator >= (const Rational& x, const Rational& y);
friend Rational operator +  (const Rational& x, const Rational& y);
friend Rational operator -  (const Rational& x, const Rational& y);
friend Rational operator *  (const Rational& x, const Rational& y);
friend Rational operator /  (const Rational& x, const Rational& y);
Rational&       operator += (const Rational& y);
Rational&       operator -= (const Rational& y);
Rational&       operator *= (const Rational& y);
Rational&       operator /= (const Rational& y);
friend Rational operator <? (const Rational& x, const Rational& y);
friend Rational operator >? (const Rational& x, const Rational& y);
friend Rational operator - (const Rational& x);
void            negate();
void            invert();
friend int      sign(const Rational& x);
friend Rational abs(const Rational& x);
friend Rational sqr(const Rational& x);
friend Rational pow(const Rational& x, long y);
friend Rational pow(const Rational& x, const Integer& y);
const Integer&  numerator() const;
const Integer&  denominator() const;
                operator double() const;
friend Integer  floor(const Rational& x);
friend Integer  ceil(const Rational& x);
friend Integer  trunc(const Rational& x);
friend Integer  round(const Rational& x);
friend istream& operator >> (istream& s, Rational& y);
friend ostream& operator << (ostream& s, const Rational& y);
int       fits_in_float() const;
int       fits_in_double() const;
friend int      compare(const Rational& x, const Rational& y);
friend void     add(const Rational& x, const Rational& y, Rational& dest);
friend void     sub(const Rational& x, const Rational& y, Rational& dest);
friend void     mul(const Rational& x, const Rational& y, Rational& dest);
friend void     div(const Rational& x, const Rational& y, Rational& dest);
```

```
  void    error(const char* msg) const;
  int             OK() const;

};

typedef Rational RatTmp;
inline Rational::Rational() : num(&_ZeroRep), den(&_OneRep)
inline Rational::~Rational()
inline Rational::Rational(const Rational& y) :num(y.num), den(y.den)
inline Rational::Rational(const Integer& n) :num(n), den(&_OneRep)
inline Rational::Rational(const Integer& n, const Integer& d) :num(n),den(d)
inline Rational::Rational(long n) :num(n), den(&_OneRep)
inline Rational::Rational(int n) :num(n), den(&_OneRep)
inline Rational::Rational(long n, long d) :num(n), den(d)
inline Rational::Rational(int n, int d) :num(n), den(d)
inline Rational::Rational(long n, unsigned long d) :num(n), den(d)
inline Rational::Rational(unsigned long n, long d) :num(n), den(d)
inline Rational::Rational(unsigned long n, unsigned long d) :num(n), den(d)
inline Rational& Rational::operator =  (const Rational& y)
inline int operator == (const Rational& x, const Rational& y)
inline int operator != (const Rational& x, const Rational& y)
inline int operator <  (const Rational& x, const Rational& y)
inline int operator <= (const Rational& x, const Rational& y)
inline int operator >  (const Rational& x, const Rational& y)
inline int operator >= (const Rational& x, const Rational& y)
inline int sign(const Rational& x)
inline void Rational::negate()
inline Rational& Rational::operator += (const Rational& y)
inline Rational& Rational::operator -= (const Rational& y)
inline Rational& Rational::operator *= (const Rational& y)
inline Rational& Rational::operator /= (const Rational& y)
inline const Integer& Rational::numerator() const { return num; }
inline const Integer& Rational::denominator() const { return den; }
inline Rational::operator double() const { return ratio(num, den); }
inline Rational operator <? (const Rational& x, const Rational& y)
inline Rational operator >? (const Rational& x, const Rational& y)
inline Rational operator + (const Rational& x, const Rational& y) return r
inline Rational operator - (const Rational& x, const Rational& y) return r
inline Rational operator * (const Rational& x, const Rational& y) return r
inline Rational operator / (const Rational& x, const Rational& y) return r
```

———-

13.6 libg++/include/std.h

— std.h —

```
#include <_G_config.h>
#include <cstdlib>
#include <cstring>
#include <unistd.h>
#include <cstdio>
#include <cerrno>
#include <fcntl.h>
extern "C" {
int strcasecmp _G_ARGS((const char*, const char*));
}
```

———-

13.7 libg++/include/String.h

— **String.h** —

```
#define _String_h 1
#include <iostream.h>
```

———-

13.7.1 struct StrRep

— **struct StrRep** —

```
struct StrRep
{
  unsigned short    len;
  unsigned short    sz;
  char              s[1];
};
StrRep*     Salloc(StrRep*, const char*, int, int);
StrRep*     Scopy(StrRep*, const StrRep*);
StrRep*     Scat(StrRep*, const char*, int, const char*, int);
StrRep*     Scat(StrRep*, const char*, int,const char*,int, const char*,int);
StrRep*     Sprepend(StrRep*, const char*, int);
StrRep*     Sreverse(const StrRep*, StrRep*);
StrRep*     Supcase(const StrRep*, StrRep*);
StrRep*     Sdowncase(const StrRep*, StrRep*);
StrRep*     Scapitalize(const StrRep*, StrRep*);
```

———-

13.7.2 class SubString

— class SubString —

```
class SubString
{
  friend class      String;
protected:
  String&           S;
  unsigned short    pos;
  unsigned short    len;
  void              assign(const StrRep*, const char*, int = -1);
                    SubString(String& x, int p, int l);
                    SubString(const SubString& x);
public:
                   ~SubString();
  SubString&        operator =  (const String&     y);
  SubString&        operator =  (const SubString&  y);
  SubString&        operator =  (const char* t);
  SubString&        operator =  (char        c);
  int               contains(char        c) const;
  int               contains(const String&     y) const;
  int               contains(const SubString&  y) const;
  int               contains(const char* t) const;
  friend ostream&   operator<<(ostream& s, const SubString& x);
  unsigned int      length() const;
  int               empty() const;
  const char*       chars() const;
  int               OK() const;
};
```

———-

13.7.3 class String

— class String —

```
class String
{
  friend class      SubString;
protected:
  StrRep*           rep;
  int               search(int, int, const char*, int = -1) const;
  int               search(int, int, char) const;
  int               match(int, int, int, const char*, int = -1) const;
  int               _gsub(const char*, int, const char* ,int);
  SubString         _substr(int, int);
public:
```

```
                String();
                String(const String& x);
                String(const SubString&  x);
                String(const char* t);
                String(const char* t, int len);
                String(char c);
                ~String();
String&         operator =  (const String&     y);
String&         operator =  (const char* y);
String&         operator =  (char        c);
String&         operator =  (const SubString&  y);
String&         operator += (const String&     y);
String&         operator += (const SubString&  y);
String&         operator += (const char* t);
String&         operator += (char        c);
void            prepend(const String&     y);
void            prepend(const SubString&  y);
void            prepend(const char* t);
void            prepend(char        c);
friend inline void cat(const String&, const String&, String&);
friend inline void cat(const String&, const SubString&, String&);
friend inline void cat(const String&, const char*, String&);
friend inline void cat(const String&, char, String&);
friend inline void cat(const SubString&, const String&, String&);
friend inline void cat(const SubString&, const SubString&, String&);
friend inline void cat(const SubString&, const char*, String&);
friend inline void cat(const SubString&, char, String&);
friend inline void cat(const char*, const String&, String&);
friend inline void cat(const char*, const SubString&, String&);
friend inline void cat(const char*, const char*, String&);
friend inline void cat(const char*, char, String&);
friend inline void cat(const String&,const String&, const String&,String&);
friend inline void cat(const String&,const String&,const SubString&,String&);
friend inline void cat(const String&,const String&, const char*, String&);
friend inline void cat(const String&,const String&, char, String&);
friend inline void cat(const String&,const SubString&,const String&,String&);
inline friend void cat(const String&,const SubString&,
                       const SubString&,String&);
friend inline void cat(const String&,const SubString&, const char*, String&);
friend inline void cat(const String&,const SubString&, char, String&);
friend inline void cat(const String&,const char*, const String&,    String&);
friend inline void cat(const String&,const char*, const SubString&, String&);
friend inline void cat(const String&,const char*, const char*, String&);
friend inline void cat(const String&,const char*, char, String&);
friend inline void cat(const char*, const String&, const String&,String&);
friend inline void cat(const char*,const String&,const SubString&,String&);
friend inline void cat(const char*,const String&, const char*, String&);
friend inline void cat(const char*,const String&, char, String&);
friend inline void cat(const char*,const SubString&,const String&,String&);
friend inline void cat(const char*,const SubString&,
```

```
                      const SubString&,String&);
friend inline void cat(const char*,const SubString&, const char*, String&);
friend inline void cat(const char*,const SubString&, char, String&);
friend inline void cat(const char*,const char*, const String&,    String&);
friend inline void cat(const char*,const char*, const SubString&, String&);
friend inline void cat(const char*,const char*, const char*, String&);
friend inline void cat(const char*,const char*, char, String&);
int               index(char        c, int startpos = 0) const;
int               index(const String&     y, int startpos = 0) const;
int               index(const SubString&  y, int startpos = 0) const;
int               index(const char* t, int startpos = 0) const;
int               contains(char        c) const;
int               contains(const String&     y) const;
int               contains(const SubString&  y) const;
int               contains(const char* t) const;
int               contains(char        c, int pos) const;
int               contains(const String&     y, int pos) const;
int               contains(const SubString&  y, int pos) const;
int               contains(const char* t, int pos) const;
int               matches(char        c, int pos = 0) const;
int               matches(const String&     y, int pos = 0) const;
int               matches(const SubString&  y, int pos = 0) const;
int               matches(const char* t, int pos = 0) const;
int               freq(char        c) const;
int               freq(const String&     y) const;
int               freq(const SubString&  y) const;
int               freq(const char* t) const;
SubString         at(int         pos, int len);
SubString         operator () (int         pos, int len);
SubString         at(const String&     x, int startpos = 0);
SubString         at(const SubString&  x, int startpos = 0);
SubString         at(const char* t, int startpos = 0);
SubString         at(char        c, int startpos = 0);
SubString         before(int          pos);
SubString         before(const String&      x, int startpos = 0);
SubString         before(const SubString&   x, int startpos = 0);
SubString         before(const char*  t, int startpos = 0);
SubString         before(char         c, int startpos = 0);
SubString         through(int          pos);
SubString         through(const String&      x, int startpos = 0);
SubString         through(const SubString&   x, int startpos = 0);
SubString         through(const char*  t, int startpos = 0);
SubString         through(char         c, int startpos = 0);
SubString         from(int          pos);
SubString         from(const String&      x, int startpos = 0);
SubString         from(const SubString&   x, int startpos = 0);
SubString         from(const char*  t, int startpos = 0);
SubString         from(char         c, int startpos = 0);
SubString         after(int         pos);
SubString         after(const String&     x, int startpos = 0);
```

```
  SubString         after(const SubString&  x, int startpos = 0);
  SubString         after(const char* t, int startpos = 0);
  SubString         after(char        c, int startpos = 0);
  void              del(int         pos, int len);
  void              del(const String&     y, int startpos = 0);
  void              del(const SubString&  y, int startpos = 0);
  void              del(const char* t, int startpos = 0);
  void              del(char        c, int startpos = 0);
  int               gsub(const String&     pat, const String&     repl);
  int               gsub(const SubString&  pat, const String&     repl);
  int               gsub(const char* pat, const String&     repl);
  int               gsub(const char* pat, const char* repl);
  friend int        split(const String& x, String res[], int maxn,
                          const String& sep);
  friend String     common_prefix(const String& x, const String& y,
                                  int startpos = 0);
  friend String     common_suffix(const String& x, const String& y,
                                  int startpos = -1);
  friend String     replicate(char        c, int n);
  friend String     replicate(const String&     y, int n);
  friend String     join(String src[], int n, const String& sep);
  friend inline String     reverse(const String& x);
  friend inline String     upcase(const String& x);
  friend inline String     downcase(const String& x);
  friend inline String     capitalize(const String& x);
  void              reverse();
  void              upcase();
  void              downcase();
  void              capitalize();
  char&             operator [] (int i);
  const char&       operator [] (int i) const;
  char              elem(int i) const;
  char              firstchar() const;
  char              lastchar() const;
                    operator const char*() const;
  const char*       chars() const;
  friend inline ostream&   operator<<(ostream& s, const String& x);
  friend ostream&   operator<<(ostream& s, const SubString& x);
  friend istream&   operator>>(istream& s, String& x);
  friend int        readline(istream& s, String& x,
                             char terminator = '\n',
                             int discard_terminator = 1);
  unsigned int      length() const;
  int               empty() const;
  void              alloc(int newsize);
  int               allocation() const;
  void     error(const char* msg) const;
  int               OK() const;
};
```

```
typedef String StrTmp;
int        compare(const String&    x, const String&     y);
int        compare(const String&    x, const SubString&  y);
int        compare(const String&    x, const char* y);
int        compare(const SubString& x, const String&     y);
int        compare(const SubString& x, const SubString&  y);
int        compare(const SubString& x, const char* y);
int        fcompare(const String&   x, const String&     y);
extern StrRep  _nilStrRep;
extern String _nilString;
inline unsigned int String::length() const
inline int         String::empty() const
inline const char* String::chars() const
inline int         String::allocation() const
inline unsigned int SubString::length() const
inline int         SubString::empty() const
inline const char* SubString::chars() const
inline String::String() : rep(&_nilStrRep)
inline String::String(const String& x) : rep(Scopy(0, x.rep))
inline String::String(const char* t) : rep(Salloc(0, t, -1, -1))
inline String::String(const char* t, int tlen) : rep(Salloc(0, t, tlen, tlen))
inline String::String(const SubString& y)
  : rep(Salloc(0, y.chars(), y.length(), y.length()))
inline String::String(char c) : rep(Salloc(0, &c, 1, 1))
inline String::~String()
inline SubString::SubString(const SubString& x)
  :S(x.S), pos(x.pos), len(x.len)
inline SubString::SubString(String& x, int first, int l)
  :S(x), pos(first), len(l)
inline SubString::~SubString()
inline String& String::operator =  (const String& y)
inline String& String::operator=(const char* t)
inline String& String::operator=(const SubString&  y)
inline String& String::operator=(char c)
inline SubString& SubString::operator = (const char* ys)
inline SubString& SubString::operator = (char ch)
inline SubString& SubString::operator = (const String& y)
inline SubString& SubString::operator = (const SubString& y)
inline void cat(const String& x, const String& y, String& r)
inline void cat(const String& x, const SubString& y, String& r)
inline void cat(const String& x, const char* y, String& r)
inline void cat(const String& x, char y, String& r)
inline void cat(const SubString& x, const String& y, String& r)
inline void cat(const SubString& x, const SubString& y, String& r)
inline void cat(const SubString& x, const char* y, String& r)
inline void cat(const SubString& x, char y, String& r)
inline void cat(const char* x, const String& y, String& r)
inline void cat(const char* x, const SubString& y, String& r)
inline void cat(const char* x, const char* y, String& r)
inline void cat(const char* x, char y, String& r)
```

```
inline void cat(const String& a, const String& x, const String& y, String& r)
inline void cat(const String& a, const String& x,const SubString& y, String& r)
inline void cat(const String& a, const String& x, const char* y, String& r)
inline void cat(const String& a, const String& x, char y, String& r)
inline void cat(const String& a, const SubString& x,const String& y, String& r)
inline void cat(const String& a, const SubString& x,
                const SubString& y, String& r)
inline void cat(const String& a, const SubString& x, const char* y, String& r)
inline void cat(const String& a, const SubString& x, char y, String& r)
inline void cat(const String& a, const char* x, const String& y, String& r)
inline void cat(const String& a, const char* x, const SubString& y, String& r)
inline void cat(const String& a, const char* x, const char* y, String& r)
inline void cat(const String& a, const char* x, char y, String& r)
inline void cat(const char* a, const String& x, const String& y, String& r)
inline void cat(const char* a, const String& x, const SubString& y, String& r)
inline void cat(const char* a, const String& x, const char* y, String& r)
inline void cat(const char* a, const String& x, char y, String& r)
inline void cat(const char* a, const SubString& x, const String& y, String& r)
inline void cat(const char* a, const SubString& x,const SubString& y,String& r)
inline void cat(const char* a, const SubString& x, const char* y, String& r)
inline void cat(const char* a, const SubString& x, char y, String& r)
inline void cat(const char* a, const char* x, const String& y, String& r)
inline void cat(const char* a, const char* x, const SubString& y, String& r)
inline void cat(const char* a, const char* x, const char* y, String& r)
inline void cat(const char* a, const char* x, char y, String& r)
inline String& String::operator +=(const String& y)
inline String& String::operator +=(const SubString& y)
inline String& String::operator += (const char* y)
inline String& String:: operator +=(char y)
inline String operator + (const String& x, const String& y) return r;
inline String operator + (const String& x, const SubString& y) return r;
inline String operator + (const String& x, const char* y) return r;
inline String operator + (const String& x, char y) return r;
inline String operator + (const SubString& x, const String& y) return r;
inline String operator + (const SubString& x, const SubString& y) return r;
inline String operator + (const SubString& x, const char* y) return r;
inline String operator + (const SubString& x, char y) return r;
inline String operator + (const char* x, const String& y) return r;
inline String operator + (const char* x, const SubString& y) return r;
inline String reverse(const String& x) return r;
inline String upcase(const String& x) return r;
inline String downcase(const String& x) return r;
inline String capitalize(const String& x) return r;
inline String operator + (const String& x, const String& y)
inline String operator + (const String& x, const SubString& y)
inline String operator + (const String& x, const char* y)
inline String operator + (const String& x, char y)
inline String operator + (const SubString& x, const String& y)
inline String operator + (const SubString& x, const SubString& y)
inline String operator + (const SubString& x, const char* y)
```

```
inline String operator + (const SubString& x, char y)
inline String operator + (const char* x, const String& y)
inline String operator + (const char* x, const SubString& y)
inline String reverse(const String& x)
inline String upcase(const String& x)
inline String downcase(const String& x)
inline String capitalize(const String& x)
inline void String::prepend(const String& y)
inline void String::prepend(const char* y)
inline void String::prepend(char y)
inline void String::prepend(const SubString& y)
inline void String::reverse()
inline void String::upcase()
inline void String::downcase()
inline void String::capitalize()
inline char&  String::operator [] (int i)
inline const char&  String::operator [] (int i) const
inline char  String::elem (int i) const
inline char  String::firstchar() const
inline char  String::lastchar() const
inline int String::index(char c, int startpos) const
inline int String::index(const char* t, int startpos) const
inline int String::index(const String& y, int startpos) const
inline int String::index(const SubString& y, int startpos) const
inline int String::contains(char c) const
inline int String::contains(const char* t) const
inline int String::contains(const String& y) const
inline int String::contains(const SubString& y) const
inline int String::contains(char c, int p) const
inline int String::contains(const char* t, int p) const
inline int String::contains(const String& y, int p) const
inline int String::contains(const SubString& y, int p) const
inline int String::matches(const SubString& y, int p) const
inline int String::matches(const String& y, int p) const
inline int String::matches(const char* t, int p) const
inline int String::matches(char c, int p) const
inline int SubString::contains(const char* t) const
inline int SubString::contains(const String& y) const
inline int SubString::contains(const SubString&  y) const
inline int SubString::contains(char c) const
inline int String::gsub(const String& pat, const String& r)
inline int String::gsub(const SubString&  pat, const String& r)
inline int String::gsub(const char* pat, const String& r)
inline int String::gsub(const char* pat, const char* r)
inline  ostream& operator<<(ostream& s, const String& x)
inline int operator==(const String& x, const String& y)
inline int operator!=(const String& x, const String& y)
inline int operator>(const String& x, const String& y)
inline int operator>=(const String& x, const String& y)
inline int operator<(const String& x, const String& y)
```

```
inline int operator<=(const String& x, const String& y)
inline int operator==(const String& x, const SubString&  y)
inline int operator!=(const String& x, const SubString&  y)
inline int operator>(const String& x, const SubString&  y)
inline int operator>=(const String& x, const SubString&  y)
inline int operator<(const String& x, const SubString&  y)
inline int operator<=(const String& x, const SubString&  y)
inline int operator==(const String& x, const char* t)
inline int operator!=(const String& x, const char* t)
inline int operator>(const String& x, const char* t)
inline int operator>=(const String& x, const char* t)
inline int operator<(const String& x, const char* t)
inline int operator<=(const String& x, const char* t)
inline int operator==(const SubString& x, const String& y)
inline int operator!=(const SubString& x, const String& y)
inline int operator>(const SubString& x, const String& y)
inline int operator>=(const SubString& x, const String& y)
inline int operator<(const SubString& x, const String& y)
inline int operator<=(const SubString& x, const String& y)
inline int operator==(const SubString& x, const SubString&  y)
inline int operator!=(const SubString& x, const SubString&  y)
inline int operator>(const SubString& x, const SubString&  y)
inline int operator>=(const SubString& x, const SubString&  y)
inline int operator<(const SubString& x, const SubString&  y)
inline int operator<=(const SubString& x, const SubString&  y)
inline int operator==(const SubString& x, const char* t)
inline int operator!=(const SubString& x, const char* t)
inline int operator>(const SubString& x, const char* t)
inline int operator>=(const SubString& x, const char* t)
inline int operator<(const SubString& x, const char* t)
inline int operator<=(const SubString& x, const char* t)
inline SubString String::_substr(int first, int l)
```

———-

14 The Map classes

14.1 Map/include/Automorphism.h

— Automorphism.h —

```
#include "Endomorphism.h"
#include "FreeGroup.h"
```

———-

14.1.1 class Automorphism

— class Automorphism —

```
class Automorphism : public Endomorphism {
public:
  Automorphism( const FGGroup& dom) : Endomorphism(dom)
  Automorphism( const FGGroup& dom,  const VectorOf<Word>& generatingImages )
    : Endomorphism(dom,generatingImages)
  Automorphism(FGGroup& dom, Generator x, Word& w) : Endomorphism(dom)
  Automorphism(FGGroup& dom, Word& w) : Endomorphism(dom)
  Automorphism( const Map& m ) : Endomorphism(m)
  Automorphism inverse() const
};
```

———-

14.2 Map/include/Endomorphism.h

— Endomorphism.h —

```
#include "Map.h"
```

———-

14.2.1 class Endomorphism

— class Endomorphism —

```
class Endomorphism : public Map {
public:
  Endomorphism( const FGGroup& dom) : Map(dom, dom)
  Endomorphism( const FGGroup& dom, const VectorOf<Word>& generatingImages )
    : Map(dom, dom, generatingImages)
  Endomorphism( const Map& m ) : Map(m)
  void makeIdentity()
  void reduceGenImages()
  bool operator ==(const Endomorphism& e) const
};
```

———-

14.3 Map/include/MapEnum.h

— **MapEnum.h** —

```
#include "Map.h"
```

———-

14.3.1 class IntTuples

— **class IntTuples** —

```
class IntTuples
{
public:
  IntTuples(const int tupleLength, const int startRadius = 1);
  ~IntTuples( )
  const int* nextTuple( );

private:
  int r;
  int* tuple;
  int* end;
  int* sp;
};
```

———-

14.3.2 class MapEnum

— **class MapEnum** —

```
class MapEnum
{
public:
  MapEnum(const FGGroup& domain, const FGGroup& range, int radius = 1);
  Map nextMap(int jump = 1);
private:
  Word kthWord(int k, int n);
  int domainRank;
  int rangeRank;
  const FGGroup& theDomain;
  const FGGroup& theRange;
  IntTuples theTuples;
};
```

———-

14.4 Map/include/MapParser.h

— **MapParser.h** —

```
#include "WordParser.h"
#include "Map.h"
```

———-

14.4.1 class MapParser

— **class MapParser** —

```
class MapParser : public WordParser {
public:
  MapParser(istream &istr) : WordParser(istr) { }
  MapRep* parseMap(
    const FGGroup& domain,
    const FGGroup& range,
    Chars& errMesg
  );
};
```

———-

14.5 Map/include/RandomAutoInFree.h

— **RandomAutoInFree.h** —

```
#include "RandomNumbers.h"
#include "FreeGroup.h"
#include "Map.h"
```

———-

14.5.1 class RandomAutoInFree

— **class RandomAutoInFree** —

```
class RandomAutoInFree
{
public:
  RandomAutoInFree(const FreeGroup& F, int avgNumGens);
  Map getAutomorphism();
```

```
  Map getFiniteAutomorphism(int& order);
  Trichotomy isFiniteAutomorphism(const Map& m,int& order)const;
private:
  bool isTooLong(const VectorOf<Word>& images )const;
  int sumOfImagesLens(const VectorOf<Word>& images)const;
  static const short MAXLENGTH = 32767;
  FreeGroup theGroup;
  int numberOfGroupGens;
  NormalRandom numGensPicker;
  UniformRandom typeGenPicker;
  int avgNumGens;
};
```

———-

15 The Matrix classes

15.1 Matrix/include/GaussTransformation.h

— **GaussTransformation.h** —

```
#include "Integer.h"
#include "Rational.h"
#include "Matrix.h"

template <class T> class GaussTransformation {
public:
  GaussTransformation(const Matrix<T>& M, bool buildTransMatrix = false, bool buildInvTransMatrix = fa
    matrix( M ),
    bDone( false ), bStart( false ),transformed(false),
    isSingularMatrix( dontknow ), isInvertibleMatrix( dontknow ),
    transMatrix(NULL),invTransMatrix(NULL),buildTransformations(buildTransMatrix),
    buildInverseTransformations(buildInvTransMatrix)
  GaussTransformation(const GaussTransformation&);
  GaussTransformation operator = ( const GaussTransformation& );
  ~GaussTransformation()
  void startComputation( );
  void run()
  void runRow(int rowNum);
  void runUntilDiagHasZero();
  void runWhileDiagHasSingles();
  int getCurrentRow() const
  int getCurrentCol() const
  bool canChange( ) const
  bool done( ) const
  Trichotomy isSingular( ) const
  Trichotomy isInvertible( ) const;
```

```
  const Matrix<T>& getMatrix() const
  const Matrix<T>& getTrMatrix()const
  const Matrix<T>& getInvTrMatrix()const
  Matrix<T>& refMatrix()
  bool isTransformed() const
private:
  bool bDone;
  bool bStart;
  bool transformed;
  Trichotomy isSingularMatrix;
  Trichotomy isInvertibleMatrix;
  Matrix<T> matrix;
  Matrix<T>* transMatrix;
  Matrix<T>* invTransMatrix;
  int untilRow;
  int currentRow;
  int currentCol;
  bool buildTransformations;
  bool buildInverseTransformations;
  static const int  UNTILNOT_0 = 0;
  static const int  UNTIL_1 = 1;
  static const int  RUNROW = 2;
  static const int  RUNALL = 3;
  void continueComputation( int whatDeal);
  void finishComputation( Trichotomy isSingular, Trichotomy isInvertible )
  void addRow( int firstRow, int secondRow, T koef);
  void makeZero( int& row1, int& row2 );
};
```

———-

15.2 Matrix/include/HomomorphismBuilder.h

— HomomorphismBuilder.h —

```
#include "FPGroup.h"
#include "Matrix.h"

template <class R> class HomomorphismBuilder
{
 public:
  HomomorphismBuilder(FPGroup& G, int sizeOfMatrix = 2) : group( G ),
    matrixSize( sizeOfMatrix ), bDone( false ), bStart( false ),
    homomorphism( G.numberOfGenerators() ),
    invertedMatrices( G.numberOfGenerators() )
  void startComputation( )
  void continueComputation( );
  bool done( ) const
```

```
  VectorOf< Matrix<R> > getHomomorphism( ) const
  bool bDone;
  bool bStart;
  VectorOf< Matrix<R> > homomorphism;
  VectorOf< Matrix<R> > invertedMatrices;
  int matrixSize;
  FPGroup& group;
  void finishComputation( )
};
```

———-

15.3 Matrix/include/MatrixComputations.h

— **MatrixComputations.h** —

```
#include "GaussTransformation.h"

template <class R> class MatrixComputations {
public:
  MatrixComputations( const Matrix<R>& matrix) :
    isInvertible(dontknow),
    inverseMatrix(NULL)
  ~MatrixComputations()
   const Matrix<R>& matrix() const
  int size() const
  bool isIdentity() const;
  R getDeterminant();
  bool detKnow() const
  Trichotomy isInvertibleMatrix() const
  void invertMatrix();
  const Matrix<R>& getInverseMatrix() const
  R det() const
  friend ostream& operator < ( ostream& ostr, const MatrixComputations& DA )
  friend istream& operator > ( istream& istr, MatrixComputations& DA)
private:
  void write( ostream& ostr ) const
  void read( istream& istr )
  void abolishCoefficients(Matrix<R>& matrix );
  bool detKnown;
  Trichotomy isInvertible;
  R determinant;
  Matrix<R>* inverseMatrix;
  Matrix<R> theMatrix;
};
```

———-

15.4 Matrix/include/Matrix.h

— **Matrix.h** —

```
#include "DArray.h"
#include "DerivedObjectOf.h"

template <class R> class MatrixRep : public DArrayRep<R> {
public:
  MatrixRep( int height, int width) : DArrayRep<R>(height,width)
  MatrixRep( int n ) : DArrayRep<R>(n)
  MatrixRep* clone( )
  MatrixRep& operator += ( const MatrixRep& );
  MatrixRep& operator -= ( const MatrixRep& );
  MatrixRep& operator *= ( const MatrixRep& );
  MatrixRep operator + ( const MatrixRep& ) const;
  MatrixRep operator - ( const MatrixRep& ) const;
  MatrixRep operator * ( const MatrixRep& ) const;
  MatrixRep operator - ( ) const;
private:
};

template <class R> class Matrix :
  public DerivedObjectOf<DArray<R> ,MatrixRep<R> >{
public:
  Matrix(int n = 0 ) :
    DerivedObjectOf<DArray<R>,MatrixRep<R> >(new MatrixRep<R>(n))
  Matrix(int h, int w) :
    DerivedObjectOf<DArray<R>,MatrixRep<R> >(new MatrixRep<R>(h, w))
  Matrix& operator += ( const Matrix& M )
  Matrix& operator -= ( const Matrix& M )
  Matrix& operator *= ( const Matrix& M )
  Matrix operator + ( const Matrix& M) const
  Matrix operator - ( const Matrix& M ) const
  Matrix operator * ( const Matrix& M ) const
  Matrix operator - ( ) const
protected :
  Matrix( MatrixRep<R> newrep ) :
            DerivedObjectOf<DArray<R>,MatrixRep<R> >(new MatrixRep<R>(newrep))
  Matrix( MatrixRep<R>* newrep ) :
            DerivedObjectOf<DArray<R>,MatrixRep<R> >(newrep)
};
```

———-

15.5 Matrix/include/RandomMatrix.h

— **RandomMatrix.h** —

```
#include "Matrix.h"

template <class R> class RandomMatrix
{
public:
  RandomMatrix( int n = 0 )
  int getSize() const
  Matrix<R> getRandomMatrix( );
private:
  int size;
  int coefficient( );
  Matrix<R> getAtomicMatrix( );
};
```

———-

15.6 Matrix/include/RingParser.h

— **RingParser.h** —

```
#include "RingEltParser.h"
#include "Polynomial.h"

template <class R> class MonomialParser : public RingEltParser<R>
{
  friend class PolynomialParser<R>;
public:
  MonomialParser(istream &istr) : RingEltParser<R>(istr)
  Monomial<R> parseMonomial( Chars& errMesg )
private:
  static const maxNumberOfVariables = 100;
  Monomial<R> getMonomial( Chars& errMesg, bool& isLastMonomial );
};

template <class R> class PolynomialParser : public MonomialParser<R>
{
public:
  PolynomialParser(istream &istr) : MonomialParser<R>(istr)
  Polynomial<R> parsePolynomial( Chars& errMesg );
private:
};
```

———-

16 The Nilpotent Group classes

16.1 NilpotentGroup/include/BasicCommutators.h

— BasicCommutators.h —

```
#include "Vector.h"
#include "Word.h"
#include "Generator.h"
#include "PolyWord.h"
```

———-

16.1.1 class BCEntry

— class BCEntry —

```
class BCEntry {
public:
  BCEntry( )
  BCEntry(const Generator& g) : weight(1), rPart(ord(g))
  BCEntry(int i) : weight(1), rPart(i)
  BCEntry(int left, int right, int aWeight) :
    weight(aWeight), lPart(left), rPart(right)
  BCEntry& operator = ( int i )
  bool operator == ( const BCEntry& v ) const;
  bool operator != ( const BCEntry& v ) const
  bool operator < ( const BCEntry& v ) const;
  bool operator > ( const BCEntry& v ) const
  bool operator <= ( const BCEntry& v ) const
  bool operator >= ( const BCEntry& v ) const
  friend ostream& operator << (ostream& s, const BCEntry& );
  friend ostream& operator < ( ostream& ostr, const BCEntry& BC )
  friend istream& operator > ( istream& istr, const BCEntry& bc)
  friend class BasicCommutators;
private:
  int weight;
  int lPart, rPart;
};
```

———-

16.1.2 class BasicCommutators

— class BasicCommutators —

```
class BasicCommutators {
public:
  BasicCommutators(int numgen, int nilclass, bool initialize = true);
  void initialize() const;
  int theHirschNumber() const
  int numberOfGenerators() const
  int nilpotencyClass() const
  int numberOfWeight(int i) const
  int theFirstOfWeight(int i) const
  bool isInitialized() const
  class NGWordForms wordForm() const;
  int weightOf(int i) const
  int leftIndexOf(int i) const
  int rightIndexOf(int i) const
  virtual Chars commutatorName(int i) const
  VectorOf<Chars> commutatorNames() const;
  bool commutatorIsBasic(int g, int h) const
  int findBC(int g, int h) const;
  bool generatorsCommute(int g, int h) const
  PolyWord commuteLetters(const Letter& left, const Letter& right) const;
  PolyWord findCommutationRelation( Letter Cj, Letter Ci ) const;
  PolyWord decomposeLetter(Letter C) const;
  friend ostream& operator < ( ostream& ostr, const BasicCommutators& BC );
  friend istream& operator > ( istream& istr, const BasicCommutators& BC );
  virtual void print( ostream& BClist, const VectorOf<Chars>& genNames ) const;
  friend class NGWordForms;
private:
  int nilClass;
  int numGens;
  int theNumberOfBC;
  bool initialized;
  VectorOf<int> firstOfWeight;
  VectorOf<BCEntry> entries;
};
```

———-

16.1.3 class NGWordForms

— class NGWordForms —

```
class NGWordForms {
public:
  NGWordForms(const BasicCommutators& bc) : BC(bc)
  Word toWord(int commIndex) const;
  Word toWord(const PolyWord& w) const;
  Word toGroupWord(const Word& basicWord) const;
  Chars commutatorStructure(int commIndex) const;
```

```
  Chars asCommutatorWord(const PolyWord& w) const;
  Chars asCommutatorWord(Letter s) const;
  Chars asBracketedWord(int commIndex, const VectorOf<Chars>& genNames,
      bool embeddingBrackets = true) const;
  Chars asBracketedWord(const PolyWord& w,
      const VectorOf<Chars>& genNames) const;
private:
  const BasicCommutators& BC;
};

inline NGWordForms BasicCommutators::wordForm() const
```

———-

16.2 NilpotentGroup/include/FPNilpotentGroupRep.h

— **FPNilpotentGroupRep.h** —

```
#include "Chars.h"
#include "NilpotentGroupRep.h"
#include "BasicCommutators.h"
#include "LCSQuotients.h"
```

———-

16.2.1 struct FPNilpotentGroupRep

— **struct FPNilpotentGroupRep** —

```
struct FPNilpotentGroupRep : public NilpotentGroupRep {
  FPNilpotentGroupRep(const VectorOf<Chars>& gennames,
          int nilclass,
          const VectorOf<Word>& rels,
          enum NilpotentCollector::Type collectorType
          );
  virtual void initialize() const;
  const BasicCommutators& commutators() const
  int nilpotencyClass() const
  const NGCollector& collector() const
  bool isInitialized() const
  VectorOf<Word> relators() const
  VectorOf<Word> relationSubgroupGenerators() const;
  VectorOf<PolyWord> basis() const;
  VectorOf<Chars> basisNames() const;
  static Type type( );
  int order( ) const;
```

```
  Trichotomy isTrivial( ) const;
  Trichotomy isFinite( ) const;
  Trichotomy isInfinite( ) const;
  Trichotomy isAbelian( ) const;
  Trichotomy isFree( ) const;
  int theHirschNumber() const;
  bool isFreeNilpotent( int* freeRank, int* freeClass ) const;
  int minimalNilpotencyClass( ) const;
  bool isMalcevBasis() const;
  Integer orderOfTheTorsionSubgroup( ) const;
  VectorOf<Word> inverseAuto(const VectorOf<Word>& V) const
  Trichotomy conjugacyProblem( const Word& u, const Word& v ) const;
  Trichotomy wordProblem( const Word& w ) const;
  Trichotomy areEqual(const Elt& e1, const Elt& e2) const;
  virtual PolyWord decompose(const PolyWord& w) const;
  virtual PolyWord toCommutatorWord(const PolyWord& pw) const;
  virtual int weightOf(const Word& w) const;
  virtual int weightOf(const PolyWord& pw) const
  virtual int orderOfBasic(Generator g) const;
  virtual int orderOf(const Word& w) const;
  virtual int orderOf(const PolyWord& w) const;
  VectorOf<Word> centralizer( const Word& w) const;
  virtual void maximalRoot(const PolyWord& pw,
                           PolyWord& root, int& power ) const;
  virtual LCSQuotient getLCSQuotient( int N ) const;
  virtual void printBasis( ostream& BClist ) const;
  virtual void write( ostream& ostr ) const;
  virtual void read( istream& istr );
  void computeLCSQuotients() const;
  void computeQuotient( int theWeight );
  PolyWord decomposeInQuotient(PolyWord& rest, int k ) const;
  Word decomposeInQuotientPresentation(const Word& w, int k ) const;
  PolyWord decomposeFromQuotientPresentation(const PolyWord& w, int k ) const;
  const AbelianGroup& abelianization() const;
  virtual int weightOfDecomposed( const PolyWord& pw) const;
  PureRep* clone( ) const
  static const Type theFPNilpotentGroupType;
  Type actualType( ) const
  int nilClass;
  NGCollector theCollector;
  VectorOf<Word> theRelators;
  VectorOf<LCSQuotient> LCS;
  bool LCScomputed;
  int hirschNumber;
  VectorOf<PolyWord> theBasis;
  VectorOf<Chars> theBasisNames;
  VectorOf<int> theBasisOrders;
};
```

———-

16.3 NilpotentGroup/include/FreeNilpotentGroupRep.h

— **FreeNilpotentGroupRep.h** —

```
#include "NilpotentGroupRep.h"
#include "Chars.h"
#include "LCSQuotients.h"
```

———-

16.3.1 class FreeNilpotentGroupRep

— **class FreeNilpotentGroupRep** —

```
class FreeNilpotentGroupRep : public NilpotentGroupRep {
public:
  FreeNilpotentGroupRep(const VectorOf<Chars>& gennames,
        int nilclass,
        enum NilpotentCollector::Type collectorType
        )
    : NilpotentGroupRep(gennames), nilClass(nilclass),
      theCollector(theNumberOfGenerators, nilclass, collectorType)
  virtual void initialize() const
  virtual bool isInitialized() const
  const BasicCommutators& commutators() const
  int nilpotencyClass() const
  const NGCollector& collector() const
  virtual VectorOf<Word> relators() const
  virtual VectorOf<Word> relationSubgroupGenerators() const
  virtual VectorOf<PolyWord> basis() const;
  virtual VectorOf<Chars> basisNames() const;
  static Type type( )
  int order( ) const;
  Trichotomy isTrivial( ) const;
  Trichotomy isFinite( ) const;
  Trichotomy isInfinite( ) const;
  Trichotomy isAbelian( ) const;
  Trichotomy isFree( ) const;
  int theHirschNumber() const
  bool isFreeNilpotent( int* freeRank, int* freeClass ) const
  int minimalNilpotencyClass( ) const
  bool isMalcevBasis() const
  Integer orderOfTheTorsionSubgroup( ) const
  VectorOf<Word> inverseAuto(const VectorOf<Word>& V) const;
  Trichotomy wordProblem( const Word& w ) const;
```

```
  Trichotomy areEqual(const Elt& e1, const Elt& e2) const;
  VectorOf<Word> centralizer( const Word& w) const;
  virtual PolyWord decompose(const PolyWord& w) const
  virtual PolyWord toCommutatorWord(const PolyWord& pw) const
  virtual int weightOf(const Word& w) const
  virtual int weightOf(const PolyWord& w) const
  virtual int orderOfBasic(Generator g) const
  virtual int orderOf(const Word& w) const
  virtual int orderOf(const PolyWord& w) const
  virtual void maximalRoot(const PolyWord& pw,
                           PolyWord& root, int& power ) const;
  virtual LCSQuotient getLCSQuotient( int N ) const
  virtual void printBasis( ostream& BClist ) const
  virtual void write( ostream& ostr ) const;
  virtual void read( istream& istr );
protected:
  static void findAbelianRoot(const PolyWord& pw, PolyWord& root,
            int& power );
  static int GCDofExponents(const PolyWord& pw);
  int weightOfDecomposed( const PolyWord& pw) const;
  PureRep* clone( ) const
  static const Type theFreeNilpotentGroupType;
  Type actualType( ) const
private:
  int nilClass;
  NGCollector theCollector;
};
```

———-

16.4 NilpotentGroup/include/LCSQuotients.h

— **LCSQuotients.h** —

```
#include "PolyWord.h"
#include "AbelianGroup.h"
#include "MalcevSet.h"
```

———-

16.4.1 struct BasisWord

— **struct BasisWord** —

```
struct BasisWord {
  friend bool operator == (const BasisWord& w1, const BasisWord& w2);
```

```
  friend ostream& operator << (ostream& s, const BasisWord& w);
  friend ostream& operator < ( ostream& s, const BasisWord& w );
  friend istream& operator > ( istream& s, BasisWord& w );
  PolyWord theWord;
  int theWeight;
  int theOrder;
  int theLocalNumber;
  int theGlobalNumber;
  Chars theName;
};
```

———-

16.4.2 class LCSQuotient

— class LCSQuotient —

```
class LCSQuotient {
public:
  LCSQuotient() : abelianization( FPGroup() )
  friend bool operator == (const LCSQuotient& q1, const LCSQuotient& q2);
  friend ostream& operator << (ostream& s, const LCSQuotient& q);
  friend ostream& operator < ( ostream& s, const LCSQuotient& q );
  friend istream& operator > ( istream& s, LCSQuotient& q );
  AbelianGroup abelianization;
  VectorOf<BasisWord> generators;
  int infNumber;
  int numerationShift;
  MalcevSet basis;
};
```

———-

16.5 NilpotentGroup/include/Letter.h

— Letter.h —

```
#include "Word.h"
#include "Generator.h"
```

———-

16.5.1 struct Letter

— struct Letter —

```
struct Letter {
  Letter() : gen(1), power(0)
  Letter(Generator g, int pow = 1)
  Letter(int g, int pow = 1)
  void collectWith (Generator g)
  operator Word() const
  void printOn(ostream& os) const
  void invert()
  int shortLexIndex() const
  friend ostream& operator < (ostream& os, const Letter& let)
  friend istream& operator > (istream& is, Letter& let)
  Generator gen;
  int power;
};

inline ostream& operator<< (ostream& os, const Letter& let)
inline Letter inv( const Letter& let)
inline bool operator == (const Letter& let1, const Letter& let2)
inline bool operator != (const Letter& let1, const Letter& let2)
```

———-

16.6 NilpotentGroup/include/MalcevSet.h

— **MalcevSet.h** —

```
#include "NilpotentCollector.h"
#include "QuickAssociations.h"
#include "AbelianGroup.h"
```

———-

16.6.1 class MalcevSet

— **class MalcevSet** —

```
class MalcevSet {
public:
  MalcevSet();
  MalcevSet(const VectorOf<Word>& v, const NGCollector& nc);
  int cardinality() const
  bool isMalcevBasis() const
  bool isNormalClosure() const;
  void makeFull() const;
  MalcevSet normalClosure() const;
  AbelianGroup mapToQuotient(int weight) const;
```

```
  bool contains(const Word& w) const;
  bool decomposeWord(const PolyWord& w, PolyWord& decomp) const;
  VectorOf<Word> getWords() const;
  VectorOf<Word> getCommutatorWords() const;
  VectorOf<PolyWord> getPolyWords() const;
  void printOn(ostream& s) const;
  friend ostream& operator < (ostream& s, const MalcevSet& b);
  friend istream& operator > (istream& s, MalcevSet& b);
private:
  bool reduceWords(PolyWord& mw1, PolyWord& mw2) const;
  static PolyWord makeCommutator( PolyWord& mw1, PolyWord& mw2 );
  static int power(const PolyWord& pw)
  static int absPower(const PolyWord& pw)
  static Generator leader(const PolyWord& pw)
  static int sign(const PolyWord& pw)
  PolyWord collect(const PolyWord& pw) const
  void checkMembership(PolyWord& w) const;
  bool addWord(const Word& w);
  bool addWord(const PolyWord& w);
  void makeNormalClosure();
private:
  QuickAssociationsOf<Generator, PolyWord> theSet;
  bool isBasis;
  Trichotomy isNormal;
  NGCollector theCollector;
  friend class FPNilpotentGroupRep;
  friend class SGOfFreeNilpotentGroupRep;
};
```

———-

16.7 NilpotentGroup/include/NilpCollectors.h

— NilpCollectors.h —

```
#include "NilpotentCollector.h"
```

———-

16.7.1 class CollectorToTheLeft

— class CollectorToTheLeft —

```
class CollectorToTheLeft: public NilpotentCollector
{
public:
```

```
  CollectorToTheLeft(int numgen, int nilclass, bool initialize = true)
    : NilpotentCollector(numgen, nilclass, initialize)
  CollectorToTheLeft(const BasicCommutators& bc)
    : NilpotentCollector(bc)
  virtual NilpotentCollector::Type type() const
  virtual void collectingProcess( PolyWord& pw ) const;
  virtual NilpotentCollector * clone() const
};
```

———-

16.7.2 class CollectorFromTheLeft

— class CollectorFromTheLeft —

```
class CollectorFromTheLeft: public NilpotentCollector
{
public:
  CollectorFromTheLeft(int numgen, int nilclass, bool initialize = true)
    : NilpotentCollector(numgen, nilclass, initialize)
  CollectorFromTheLeft(const BasicCommutators& bc)
    : NilpotentCollector(bc)
  virtual NilpotentCollector::Type type() const
  virtual void collectingProcess( PolyWord& pw ) const;
  virtual NilpotentCollector * clone() const
};
```

———-

16.8 NilpotentGroup/include/NilpotentCollector.h

— NilpotentCollector.h —

```
#include "BasicCommutators.h"
```

———-

16.8.1 class NilpotentCollector

— class NilpotentCollector —

```
class NilpotentCollector
{
public:
  enum Type { STANDARD = 0, TO_THE_LEFT = 0, FROM_THE_LEFT = 1,
```

```
      FROM_THE_RIGHT = 2, POLYNOMIAL = 3  };
  NilpotentCollector(int numgen, int nilclass, bool initialize = true);
  NilpotentCollector(const BasicCommutators& bc);
  virtual void initialize() const;
  static NilpotentCollector * make(int numgen, int nilclass,
          NilpotentCollector::Type colType,
          bool initialize = true);
  virtual NilpotentCollector * clone() const = 0;
  const BasicCommutators& commutators() const
  virtual NilpotentCollector::Type type() const = 0;
  virtual bool isInitialized() const
  virtual void collectingProcess( PolyWord& pw ) const = 0;
  virtual PolyWord collect(const Word& w) const;
  virtual PolyWord collect(const PolyWord& pw) const;
  virtual bool collectLetter(PolyWord& w, Generator c) const;
  virtual PolyWord multiply(const PolyWord& pw1, const PolyWord& pw2) const;
  virtual PolyWord raiseToPower(const PolyWord& pw, int power) const;
  virtual PolyWord inverse(const PolyWord& pw) const;
  virtual int weightOf(const Word& w) const;
  virtual int weightOf(const PolyWord& w) const;
  virtual ostream& writeIPC(ostream& s) const;
  virtual istream& readIPC(istream& s) const;
protected:
  BasicCommutators BC;
};
```

———-

16.8.2 class NGCollector

— class NGCollector —

```
class NGCollector {
public:
  NGCollector(int numgen, int nilclass,
        NilpotentCollector::Type collType = NilpotentCollector::STANDARD,
        bool initialize = true)
    : NC( NilpotentCollector::make(numgen, nilclass, collType, initialize ) )
  NGCollector(const NGCollector & ngc)
    : NC( ngc.NC->clone() )
  ~NGCollector()
  NGCollector & operator = (const NGCollector& ngc)
  void initialize() const
  const BasicCommutators& commutators() const
  NilpotentCollector::Type type() const
  bool isInitialized() const
  void collectingProcess( PolyWord& pw ) const
  PolyWord collect(const Word& w) const
```

```
  PolyWord collect(const PolyWord& pw) const
  bool collectLetter(PolyWord& w, Generator c) const
  PolyWord multiply(const PolyWord& pw1, const PolyWord& pw2) const
  PolyWord raiseToPower(const PolyWord& pw, int power) const
  virtual PolyWord inverse(const PolyWord& pw) const
  int weightOf(const Word& w) const
  int weightOf(const PolyWord& w) const
  ostream& writeIPC(ostream& s) const;
  istream& readIPC(istream& s) const;
private:
  NilpotentCollector * NC;
};

inline ostream& operator < ( ostream& ostr, const NGCollector& NGC )
inline istream& operator > ( istream& istr, const NGCollector& NGC)
```

———-

16.9 NilpotentGroup/include/NilpotentGroup.h

— **NilpotentGroup.h** —

```
#include "NilpotentGroupRep.h"
#include "FGGroup.h"
#include "LCSQuotients.h"
```

———-

16.9.1 class NilpotentGroup

— **class NilpotentGroup** —

```
class NilpotentGroup : public DerivedObjectOf<FGGroup, NilpotentGroupRep> {
public:
  NilpotentGroup(const VectorOf<Chars>& gennames,
     int nilclass,
     const VectorOf<Word>& rels,
     enum NilpotentCollector::Type colType
                              = NilpotentCollector::STANDARD
     );
  NilpotentGroup(const VectorOf<Chars>& gennames,
     int nilclass,
     enum NilpotentCollector::Type collectorType
                              = NilpotentCollector::STANDARD
     );
  NilpotentGroup(int numOfGen,
```

```
    int nilclass,
    const VectorOf<Word>& rels,
    enum NilpotentCollector::Type colType
                            = NilpotentCollector::STANDARD
    );
  NilpotentGroup(int numOfGen,
    int nilclass,
    enum NilpotentCollector::Type collectorType
                            = NilpotentCollector::STANDARD
    );
  void initialize() const
  static Type type( )
  bool isInitialized() const
  VectorOf<Word> relators() const
  VectorOf<Word> relationSubgroupGenerators() const
  VectorOf<PolyWord> basis() const
  VectorOf<Chars> basisNames() const
  const BasicCommutators& commutators() const
  int nilpotencyClass() const
  const NGCollector collector() const
  virtual int theHirschNumber() const
  bool isFreeNilpotent( int* freeRank, int* freeClass ) const
  int minimalNilpotencyClass( ) const
  bool isMalcevBasis() const
  Integer orderOfTheTorsionSubgroup( ) const
  PresentationForNG makePresentation() const
  VectorOf<Word> getLCSterm(int i) const
  LCSQuotient getLCSQuotient( int N ) const
  VectorOf<Word> inverseAuto(const VectorOf<Word>& V) const
  PolyWord decompose(const Word& w) const
  PolyWord decompose(const PolyWord& w) const
  Word toWord(const PolyWord& pw) const
  PolyWord toCommutatorWord(const PolyWord& pw) const
  Chars asDecomposition(const PolyWord& decomposition) const
  int weightOf(const Word& w) const
  int weightOf(const PolyWord& w) const
  int orderOfBasic(Generator g) const
  int orderOf(const Word& w) const
  int orderOf(const PolyWord& w) const
  VectorOf<Word> centralizer( const Word& w) const
  bool isInCommutatorSubgroup(const Word& w) const
  bool isCentral(const Word& w) const
  void maximalRoot(const Word& w, PolyWord& root, int& power ) const
  void maximalRoot(const PolyWord& pw, PolyWord& root, int& power ) const
  bool isProperPower(const Word& w) const
  bool isProperPower(const PolyWord& pw) const
  void printBasis( ostream& BClist ) const
protected:
  NilpotentGroup( NilpotentGroupRep* newrep )
    : DerivedObjectOf<FGGroup,NilpotentGroupRep>(newrep)
```

```
  static VectorOf<Chars> defaultNames(int num);

};
```

———-

16.10 NilpotentGroup/include/NilpotentGroupRep.h

— NilpotentGroupRep.h —

```
#include "FGGroupRep.h"
#include "NilpotentCollector.h"
#include "Presentation.h"
#include "LCSQuotients.h"
```

———-

16.10.1 struct NilpotentGroupRep

— struct NilpotentGroupRep —

```
struct NilpotentGroupRep : public FGGroupRep {
  NilpotentGroupRep(const VectorOf<Chars>& gennames ) : FGGroupRep(gennames)
  virtual void initialize() const = 0;
  virtual const BasicCommutators& commutators() const = 0;
  virtual int nilpotencyClass() const = 0;
  virtual const NGCollector& collector() const = 0;
  virtual bool isInitialized() const = 0;
  virtual VectorOf<Word> relators() const = 0;
  virtual VectorOf<Word> relationSubgroupGenerators() const = 0;
  virtual VectorOf<PolyWord> basis() const = 0;
  virtual VectorOf<Chars> basisNames() const = 0;
  virtual int theHirschNumber() const = 0;
  virtual bool isFreeNilpotent( int* freeRank, int* freeClass ) const = 0;
  virtual int minimalNilpotencyClass( ) const = 0;
  virtual bool isMalcevBasis() const = 0;
  virtual Integer orderOfTheTorsionSubgroup( ) const = 0;
  class PresentationForNG makePresentation() const;
  VectorOf<Word> getLCSterm(int i) const;
  virtual LCSQuotient getLCSQuotient( int N ) const = 0;
  Elt eval( const Word& w ) const
  Trichotomy conjugacyProblem( const Word& u, const Word& v ) const
  virtual Trichotomy isTrivialElt( const Elt& e ) const
  virtual PolyWord decompose(const Word& w) const
  virtual PolyWord decompose(const PolyWord& w) const = 0;
  Word toWord(const PolyWord& w) const
```

```
  virtual PolyWord toCommutatorWord(const PolyWord& pw) const = 0;
  Chars asDecomposition(const PolyWord& decomposition) const
  virtual VectorOf<Word> inverseAuto(const VectorOf<Word>& V) const = 0;
  virtual int weightOf(const Word& w) const = 0;
  virtual int weightOf(const PolyWord& w) const = 0;
  virtual int orderOfBasic(Generator g) const = 0;
  virtual int orderOf(const Word& w) const = 0;
  virtual int orderOf(const PolyWord& w) const = 0;
  virtual  VectorOf<Word> centralizer( const Word& w) const =0;
  bool isInCommutatorSubgroup(const Word& w) const
  bool isCentral(const Word& w) const;
  void maximalRoot(const Word& w, PolyWord& root, int& power ) const
  virtual void maximalRoot(const PolyWord& pw,
                           PolyWord& root, int& power ) const = 0;
  bool isProperPower(const Word& w) const
  bool isProperPower(const PolyWord& pw) const;
  virtual GroupRep* readFrom(istream&, Chars&) const
  virtual void printOn(ostream&) const
  virtual void printBasis( ostream& BClist ) const = 0;
  virtual void write( ostream& ostr ) const;
  virtual void read( istream& istr );
  static const Type theNilpotentGroupType;
  static Type type( )
  Type actualType( ) const
protected:
  void mapToClass( PolyWord& pw, int theClass ) const;
};
```

———-

16.11 NilpotentGroup/include/PolyWord.h

— PolyWord.h —

```
#include "Letter.h"
#include "PureRep.h"
#include "PolyWordRep.h"
```

———-

16.11.1 class PolyWord

— class PolyWord —

```
class PolyWord : public ObjectOf<PolyWordRep> {
public:
```

```
  PolyWord() : ObjectOf<PolyWordRep>( new PolyWordRep() )
  PolyWord( const Letter& let) :
    ObjectOf<PolyWordRep>( new PolyWordRep(let) )
  PolyWord( const Word& w) :
    ObjectOf<PolyWordRep>( new PolyWordRep(w) )
  PolyWord( const AbelianWord& aw) :
    ObjectOf<PolyWordRep>( new PolyWordRep(aw) )
  PolyWord(PolyWordRep* rep) :
    ObjectOf<PolyWordRep>(rep)
  int length() const
  int numberOfLetters() const
  bool isEmpty() const
  bool isCollected() const
  const Letter& firstLetter() const
  const Letter& lastLetter() const
  friend class PolyWordIterator;
  friend class ConstPolyWordIterator;
  void printOn (ostream& s) const
  friend ostream& operator < (ostream& s, const PolyWord& w)
  friend istream& operator > (istream& s, PolyWord& w)
  void debugInfo() const
  operator Word() const
  Chars toChars( const VectorOf<Chars>& names ) const
  void freelyReduce()
  PolyWord inverse() const
  void append(const PolyWord& w)
  void append(const Letter& w)
  void clear()
  void duplicate(PolyWordNode*& ptrToFirst, PolyWordNode*& ptrToLast) const
  void removeFirstLetter()
  PolyWord raiseToPower( int power ) const
};

inline ostream& operator<<(ostream& s, const PolyWord& w)
inline PolyWord operator * (const PolyWord& p1, const PolyWord& p2)
inline PolyWord commutator(const PolyWord& w1, const PolyWord& w2)
inline PolyWord commutatorOfInverses(const PolyWord& w1, const PolyWord& w2)
PolyWord evaluateWord( const Word& w, const VectorOf<PolyWord>& pw );
PolyWord evaluateWord(const PolyWord& pw, const VectorOf<PolyWord>& v);
```

———-

16.12 NilpotentGroup/include/PolyWordIterator.h

— PolyWordIterator.h —

```
#include "PolyWord.h"
```

———-

16.12.1 class PolyWordIterator

— class PolyWordIterator —

```
class PolyWordIterator {
public:
  PolyWordIterator(PolyWord& w) : theRep( *w.change() ), iter(0)
  PolyWordIterator(PolyWordRep& w) : theRep(w), iter(0)
  void startFromLeft()
  void startFromRight()
  void stepRight()
  void stepLeft()
  bool done() const
  bool searchToLeft( Generator g );
  int position() const;
  Letter& thisLetter() const
  Letter& leftLetter() const
  Letter& rightLetter() const
  bool isFirst() const
  bool isLast() const
  bool collectToRight();
  void splitToLeft();
  void insertRight(const PolyWord& w);
  void insertLeft(const PolyWord& w);
  void insertRight(const Letter& let);
  void insertLeft(const Letter& let);
  void removeThisLetter();
  void removeLeftLetter();
  void removeRightLetter();
  void decreaseLeftLetter();
  void exchangeToLeft();
private:
  PolyWordRep& theRep;
  PolyWordNode *iter;
};
```

———-

16.12.2 class ConstPolyWordIterator

— class ConstPolyWordIterator —

```
class ConstPolyWordIterator : private PolyWordIterator {
public:
  ConstPolyWordIterator(const PolyWord& w)
```

```
    : PolyWordIterator( (PolyWordRep&)*w.look() )
  ConstPolyWordIterator(const PolyWordRep& w)
    : PolyWordIterator( (PolyWordRep&)w )
  PolyWordIterator::startFromLeft;
  PolyWordIterator::startFromRight;
  PolyWordIterator::stepRight;
  PolyWordIterator::stepLeft;
  PolyWordIterator::done;
  PolyWordIterator::searchToLeft;
  PolyWordIterator::position;
  const Letter& thisLetter() const
  const Letter& leftLetter() const
  const Letter& rightLetter() const
  PolyWordIterator::isFirst;
  PolyWordIterator::isLast;
};
```

———-

16.13 NilpotentGroup/include/PolyWordRep.h

— **PolyWordRep.h** —

```
#include "Letter.h"
#include "PureRep.h"
```

———-

16.13.1 struct PolyWordNode

— **struct PolyWordNode** —

```
struct PolyWordNode {
  PolyWordNode(const Letter& e, PolyWordNode *prv = NULL,
     PolyWordNode *nxt = NULL) : value(e), prev(prv), next(nxt)
  PolyWordNode(PolyWordNode *prv = NULL, PolyWordNode *nxt = NULL)
    : prev(prv), next(nxt)
  Letter value;
  PolyWordNode *prev;
  PolyWordNode *next;
};
```

———-

16.13.2 class PolyWordRep

— class PolyWordRep —

```
class PolyWordRep : public PureRep {
public:
  PolyWordRep() : first(NULL), last(NULL), theNumberOfNodes(0)
  PolyWordRep( const Letter& let)
    : first(new PolyWordNode(let) ), last(first), theNumberOfNodes(1)
  PolyWordRep( const Word& w);
  PolyWordRep(const class AbelianWord& aw);
  PolyWordRep(const PolyWordRep& src)
    : theNumberOfNodes(src.theNumberOfNodes)
  ~PolyWordRep()
  int length() const;
  int numberOfLetters() const
  bool isCollected() const;
  const Letter& firstLetter() const
  const Letter& lastLetter() const
  void printOn (ostream& s) const;
  ostream& write(ostream& s) const;
  istream& read(istream& s);
  void debugInfo() const;
  Word toWord() const;
  Chars toChars( const VectorOf<Chars>& name ) const;
  void freelyReduce();
  PolyWord inverse() const;
  void append(const PolyWord& w);
  void append(const Letter& w);
  void clear();
  void removeFirstLetter()
  PolyWord raiseToPower( int power ) const;
  Chars inTermsOf(const VectorOf<Chars>& v) const;
  void duplicate(PolyWordNode*& ptrToFirst, PolyWordNode*& ptrToLast) const;
  static inline void bind(PolyWordNode *leftNode, PolyWordNode *rightNode)
  friend class PolyWordIterator;
  friend class ConstPolyWordIterator;
  virtual PureRep *clone() const
  void removeTheLetter( PolyWordNode *theLetter);
private:
  PolyWordNode *first, *last;
  int theNumberOfNodes;
};
```

———-

16.14 NilpotentGroup/include/Presentation.h

— **Presentation.h** —

```
#include "MalcevSet.h"
#include <iostream.h>
```

———-

16.14.1 struct NilpotentRelator

— **struct NilpotentRelator** —

```
struct NilpotentRelator {
  void setLeft( int s1, int s2 ) {
    left1 = s1;
    left2 = s2;
  }
  bool isCommutation() const
  int left1, left2;
  PolyWord right;
};

inline ostream& operator < ( ostream& s, const NilpotentRelator& rel )
inline istream& operator > ( istream& s, NilpotentRelator& rel )
ostream& operator << ( ostream& s, const NilpotentRelator& rel );
inline bool operator == (const NilpotentRelator& r1,
                         const NilpotentRelator& r2)
```

———-

16.14.2 class NilpotentPresentation

— **class NilpotentPresentation** —

```
class NilpotentPresentation {
public:
  NilpotentPresentation() : built( false ), theCollector(1,1)
  NilpotentPresentation(const NGCollector& coll,
      const VectorOf<Word>& rels)
    : built(false), theGroupRelators(rels), theCollector(coll)
  bool isBuilt() const
  void build();
  void print( ostream& s ) const;
  class FPGroup makeGroup() const;
  virtual ostream& write(ostream& s) const;
  virtual istream& read(istream& s);
protected:
  SetOf<Word> convertRelators() const;
  Chars printRelator(int i) const;
```

```
  void buildKernelRelators();
  void improvePresentation();
  virtual PolyWord decompose( const PolyWord& pw ) const = 0;
  virtual void printGenerators( ostream& s ) const = 0;
protected:
  VectorOf<PolyWord> theGenerators;
  VectorOf<Chars> theNames;
  VectorOf<NilpotentRelator> theRelators;
  bool built;
  VectorOf<Word> theGroupRelators;
  NGCollector theCollector;
};
```

———-

16.14.3 class PresentationForNG

— class PresentationForNG —

```
class PresentationForNG : public NilpotentPresentation {
public:
  PresentationForNG(const NGCollector& coll, const VectorOf<Word>& rels);
protected:
  virtual PolyWord decompose( const PolyWord& pw ) const;
  void printGenerators( ostream& s ) const
};
```

———-

16.14.4 class PresentationForSNG

— class PresentationForSNG —

```
class PresentationForSNG : public NilpotentPresentation {
public:
  PresentationForSNG(const MalcevSet& preimage, const VectorOf<Word>& rels,
         const NGCollector& coll);
protected:
  virtual PolyWord decompose( const PolyWord& pw ) const;
  virtual void printGenerators( ostream& s ) const;
private:
  MalcevSet thePreimageBasis;
};

inline ostream& operator < ( ostream& s, const NilpotentPresentation& pres )
inline istream& operator > ( istream& s, NilpotentPresentation& pres )
```

———-

16.15 NilpotentGroup/include/SGOfFNGRep.h

— **SGOfFNGRep.h** —

```
#include "SGOfNilpotentGroupRep.h"
#include "NilpotentGroup.h"
```

———-

16.15.1 class SGOfFreeNilpotentGroupRep

— **class SGOfFreeNilpotentGroupRep** —

```
class SGOfFreeNilpotentGroupRep : public SGOfNilpotentGroupRep {
public:
  SGOfFreeNilpotentGroupRep(const NilpotentGroup& ng,
                            const VectorOf<Word>& gens);
  virtual void initPreimage( ) const
  virtual void initParent( ) const
  virtual void initBasis( ) const;
  virtual const class NilpotentGroup& parentGroup() const
  virtual const class VectorOf<Word>& generators() const
  virtual const class MalcevSet& preimageBasis() const;
  bool preimageIsInitialized( ) const
  bool parentIsInitialized( ) const
  bool basisIsInitialized( ) const
  VectorOf<PolyWord> basis() const;
  VectorOf<Chars> basisNames() const;
  virtual int theHirschNumber() const
  virtual int order() const;
  virtual VectorOf<Word> normalClosureBasis() const;
  bool decompose(const PolyWord& w, PolyWord& decomp) const;
  virtual void printBasis(ostream&) const;
  virtual ostream& writeIPC(ostream& s) const;
  virtual istream& readIPC(istream& s) const;
  virtual Chars asDecomposition( const PolyWord& p ) const;
  static const Type theSGOfFreeNilpotentGroupType;
  static Type type( )
  virtual Type actualType( ) const
protected:
  virtual GenericRep* clone( ) const;
private:
  NilpotentGroup theParentGroup;
  VectorOf<Word> theGenerators;
  VectorOf<PolyWord> theBasis;
```

```
  VectorOf<Chars> theBasisNames;
  MalcevSet theBasisSet;
};
```

———-

16.16 NilpotentGroup/include/SGOfFPNGRep.h

— **SGOfFPNGRep.h** —

```
#include "SGOfNilpotentGroupRep.h"
#include "NilpotentGroup.h"
#include "SubgroupBasis.h"
```

———-

16.16.1 class SGOfFPNilpotentGroupRep

— **class SGOfFPNilpotentGroupRep** —

```
class SGOfFPNilpotentGroupRep : public SGOfNilpotentGroupRep {
public:
  SGOfFPNilpotentGroupRep(const NilpotentGroup& ng,
          const VectorOf<Word>& gens);
  virtual void initPreimage( ) const;
  virtual void initParent( ) const;
  virtual void initBasis( ) const;
  virtual const class NilpotentGroup& parentGroup() const
  virtual const class VectorOf<Word>& generators() const
  virtual const class MalcevSet& preimageBasis() const;
  bool preimageIsInitialized( ) const
  bool parentIsInitialized( ) const
  bool basisIsInitialized( ) const
  VectorOf<PolyWord> basis() const;
  VectorOf<Chars> basisNames() const;
  virtual int theHirschNumber() const
  virtual int order() const;
  virtual VectorOf<Word> normalClosureBasis() const;
  bool decompose(const PolyWord& w, PolyWord& decomp) const;
  virtual void printBasis(ostream&) const;
  virtual ostream& writeIPC(ostream& s) const;
  virtual istream& readIPC(istream& s) const;
  virtual Chars asDecomposition( const PolyWord& p ) const;
  static const Type theSGOfFPNilpotentGroupType;
  static Type type( )
```

```
  virtual Type actualType( ) const
protected:
  virtual GenericRep* clone( ) const;
private:
  MalcevSet thePreimage;
  bool preimageInitialized;
  SubgroupBasis theBasisSet;
  VectorOf<PolyWord> theBasis;
  VectorOf<Chars> theBasisNames;
};
```

———-

16.17 NilpotentGroup/include/SGOfNilpotentGroup.h

— **SGOfNilpotentGroup.h** —

```
#include "SGOfNilpotentGroupRep.h"
```

———-

16.17.1 class SGOfNilpotentGroup

— **class SGOfNilpotentGroup** —

```
class SGOfNilpotentGroup : public ObjectOf<SGOfNilpotentGroupRep> {
public:
  SGOfNilpotentGroup(const NilpotentGroup& ng, const VectorOf<Word>& gens);
  void initPreimage( ) const
  void initParent( ) const
  void initBasis( ) const
  const class NilpotentGroup& parentGroup() const
  const class VectorOf<Word>& generators() const
  const class MalcevSet& preimageBasis() const
  bool preimageIsInitialized( ) const
  bool parentIsInitialized( ) const
  bool basisIsInitialized( ) const
  static Type type( )
  Type actualType( ) const
  VectorOf<PolyWord> basis() const
  VectorOf<Chars> basisNames() const
  int theHirschNumber() const
  int index() const
  bool isTrivial() const
  bool isCentral() const
  bool isNormal() const
```

```
  bool isAbelian() const
  int subgroupClass() const
  int order() const
  VectorOf<Word> normalClosureBasis() const
  VectorOf<Word> normalClosureGens() const
  PresentationForSNG makePresentation() const
  VectorOf<Word> join(const SGOfNilpotentGroup& SGR) const
  VectorOf<Word> join(const VectorOf<Word>& V) const
  bool contains(const VectorOf<Word>& V) const
  bool contains(const SGOfNilpotentGroup& SG) const
  bool equalTo(const SGOfNilpotentGroup& SG) const
  bool contains(const Word& w) const
  bool decompose(const PolyWord& w, PolyWord& decomp) const
  void printBasis(ostream& s) const
  ostream& writeIPC(ostream& s) const
  istream& readIPC(istream& s) const
  Chars asDecomposition( const PolyWord& p ) const
private:
  static SGOfNilpotentGroupRep* makeRep(const NilpotentGroup& ng,
          const VectorOf<Word>& gens);
};

inline ostream& operator < (ostream& s, const SGOfNilpotentGroup& g)
inline istream& operator > (istream& s, const SGOfNilpotentGroup& g)
```

———-

16.18 NilpotentGroup/include/SGOfNilpotentGroupRep.h

— SGOfNilpotentGroupRep.h —

```
#include "Presentation.h"
```

———-

16.18.1 class SGOfNilpotentGroupRep

— class SGOfNilpotentGroupRep —

```
class SGOfNilpotentGroupRep : public GenericRep {
public:
  virtual void initPreimage( ) const = 0;
  virtual void initParent( ) const = 0;
  virtual void initBasis( ) const = 0;
  virtual const class NilpotentGroup& parentGroup() const = 0;
  virtual const class VectorOf<Word>& generators() const = 0;
```

```
  virtual const class MalcevSet& preimageBasis() const = 0;
  virtual bool preimageIsInitialized( ) const = 0;
  virtual bool parentIsInitialized( ) const = 0;
  virtual bool basisIsInitialized( ) const = 0;
  virtual VectorOf<PolyWord> basis() const = 0;
  virtual VectorOf<Chars> basisNames() const = 0;
  virtual int theHirschNumber() const = 0;
  int index() const;
  bool isTrivial() const;
  bool isCentral() const;
  bool isNormal() const;
  bool isAbelian() const;
  int subgroupClass() const;
  virtual int order() const = 0;
  virtual VectorOf<Word> normalClosureBasis() const = 0;
  VectorOf<Word> normalClosureGens() const;
  PresentationForSNG makePresentation() const;
  VectorOf<Word> join(const class SGOfNilpotentGroup& SGR) const;
  VectorOf<Word> join(const VectorOf<Word>& V) const;
  bool contains(const VectorOf<Word>& V) const;
  bool contains(const Word& w) const;
  virtual bool decompose(const PolyWord& w, PolyWord& decomp) const = 0;
  virtual void printBasis(ostream&) const = 0;
  virtual ostream& writeIPC(ostream& s) const;
  virtual istream& readIPC(istream& s) const;
  virtual Chars asDecomposition( const PolyWord& p ) const = 0;
  static const Type theSGOfNilpotentGroupType;
  static Type type( )
  virtual Type actualType( ) const
  virtual GenericRep* clone( ) const = 0;
};
```

———-

16.19 NilpotentGroup/include/SubgroupBasis.h

— **SubgroupBasis.h** —

```
#include "NilpotentGroup.h"
#include "QuickAssociations.h"
#include "AbelianGroup.h"
```

———-

16.19.1 class SubgroupBasis

— **class SubgroupBasis** —

```
class SubgroupBasis {
public:
  SubgroupBasis(const VectorOf<Word>& v, const NilpotentGroup& ng);
  void initialize() const;
  int cardinality() const
  const NilpotentGroup& parentGroup() const
  const VectorOf<Word>& generators() const
  bool isFull() const
  bool isNormalClosure() const;
  int theHirschNumber() const;
  SubgroupBasis normalClosure() const;
  bool contains(const Word& w) const;
  bool decomposeWord(const PolyWord& w, PolyWord& decomp) const;
  VectorOf<Word> asWords() const;
  VectorOf<PolyWord> asGroupWords() const;
  friend ostream& operator < (ostream& s, const SubgroupBasis& b);
  friend istream& operator > (istream& s, const SubgroupBasis& b);
private:
  PolyWord parentMultiply(const PolyWord& pw1, const PolyWord& pw2) const;
  PolyWord parentInvert(const PolyWord& pw) const;
  PolyWord parentRaiseToPower(const PolyWord& pw, int power) const;
  PolyWord parentCommute(const PolyWord& pw1, const PolyWord& pw2) const;
  int leaderOrder(const PolyWord& pw) const;
  PolyWord toParentBasis(const PolyWord& pw) const;
  PolyWord toBasicCommutators(const PolyWord& pw) const;
  static int power(const PolyWord& pw)
  static int absPower(const PolyWord& pw)
  static Generator leader(const PolyWord& pw)
  static int sign(const PolyWord& pw)
  bool reduceWords(PolyWord& mw1, PolyWord& mw2) const;
  void checkMembership(PolyWord& w) const;
  bool addWord(const Word& w);
  bool addWord(const PolyWord& w);
  void makeFull();
  void makeNormalClosure();
private:
  NilpotentGroup theParent;
  VectorOf<Word> theGenerators;
  QuickAssociationsOf<Generator, PolyWord> theSet;
  bool basisIsFull;
  Trichotomy isNormal;
  int hirschNumber;
};
```

17 The Packages classes

17.1 Packages/include/PackagesData.h

— **PackagesData.h** —

```
#include <fstream.h>
#include "Chars.h"
#include "Menu.h"
#include "SMFPGroup.h"
#include "PackagesObject.h"
#include "Type.h"
```

————-

17.1.1 class Record

— **class Record** —

```
class Record
{
public:
  Record(): id(0)
  virtual void readFrom( istream& in) = 0;
  virtual void writeTo ( ostream& out ) const = 0;
  int getID() const
  void setID(int i) const;
private:
  int id;
 };
```

————-

17.1.2 struct PackageRecord

— **struct PackageRecord** —

```
struct PackageRecord : public Record{
  PackageRecord(): Record(), cs( 0 ), os(0)
  PackageRecord(const Chars& n, const Chars& c, int csel, int osel ):
    Record(), cs(csel), os(osel)
  void readFrom( istream& in);
  void writeTo ( ostream& out ) const;
  Chars getName() const
  Chars getCommand() const
  int getCheckinSelection() const
```

```
  int getObjectSelection() const
  static int size()
private:
  char name[21];
  char comm[129];
  int cs;
  int os;
 };
```

———-

17.1.3 struct ParamRecord

— **struct ParamRecord** —

```
struct ParamRecord : public Record{
  enum PARAM_TYPE { INT, BOOL, STRING };
  ParamRecord(): Record(), packID( 0 )
  ParamRecord(const Chars& n, PARAM_TYPE pt ): Record(), packID(-1), pType(pt)
  void readFrom( istream& in);
  void writeTo ( ostream& out ) const;
  Chars getName() const
  int getPackageID() const
  ParamRecord::PARAM_TYPE getType() const
  void setPID(int i)
  static int size()
private:
  char name[81];
  PARAM_TYPE pType;
  int packID;
 };

template <class T> class DatabaseOf
{
 public:
  DatabaseOf(const Chars& );
  ~DatabaseOf();
  void goFirst();
  bool goTo( int i);
  void skip();
  bool done() const;
  T getRecord();
  void append( const T& );
  void del( int i );
  void change( int i,const T& r );
  int nOfRecords() const;
 private:
  Chars fileName;
```

```
  fstream dataFile;
  fstream cfgFile;
  int lastIDNumber;
};

template <class A1> class Menu1;
```

———-

17.1.4 class Packages

— class Packages —

```
class Packages : protected FEData
{
public:
   enum PObjects { GROUP, WORD, SUBGROUP, MAP, HOMO, SUBGROUP_SUBGROUP,
       WORD_WORD, SUBGROUP_WORD };
  Packages()
  void addPackage( const Chars& theName, const Chars& theCommand,
                   Type o, int t, const ListOf<ParamRecord>& params);
  void deletePackage( PObject* o, int i);
  PackageRecord getPackage( PObject*,int, ListOf<ParamRecord>&);
  int getNOfPackage( PObject* o);
  static Chars getPackagePath(PObject* o);
  static Chars getParamPath(PObject* o);
  void addPackagesToMenu(  );
  static void printObjectNames( ostream& );
  static void printObjectFileNames( ostream& out );
  static void printCheckinTypes( ostream& );
  static SMFPGroup::Checkin_Type getCheckinType( int n );
  static Chars getCheckinTypeName( int n );
};
```

———-

17.2 Packages/include/PackagesMessageParser.h

— PackagesMessageParser.h —

```
#include <fstream.h>
#include "Chars.h"
#include "Menu.h"
#include "SMFPGroup.h"
#include "PackagesObject.h"
#include "Type.h"
```

———-

17.2.1 class PackageDatabase

— class PackageDatabase —

```
class PackageDatabase
{
 public:
  PackageDatabase
};
```

———-

17.3 Packages/include/PackagesObject.h

— PackagesObject.h —

```
#include <fstream.h>
#include "Chars.h"
#include "Menu.h"
#include "FEData.h"
#include "Type.h"
#include "OID.h"
```

———-

17.3.1 class PObject

— class PObject —

```
class PObject
{
 public:
  PObject(  )
  virtual Chars getDBName() const = 0;
  virtual void printObjectName( ostream& )const  = 0;
  virtual Chars getObjectName(  ) const = 0;
  virtual Type getObjectType( )const = 0;
  virtual void writeInitString(ostream& )const = 0;
  virtual void addPackagesToMenu(  )
 private:
  PObject& operator = ( const PObject& );
};
```

———-

17.3.2 class PGroup

— class PGroup —

```
class PGroup : public PObject, protected FEData
{
public:
  PGroup()
  PGroup( const FPGroup& g, OID o, int nc) :
    theGroup( g ),
    oid(o),
    nilpotClass( nc )
  Chars getDBName() const
  void printObjectName( ostream& o ) const;
  Chars getObjectName(  ) const;
  void addPackagesToMenu(  );
  Type getObjectType( )const
  static const Type thePGroupType;
  static Type type( )
  Chars ipcFileName( ) const
  void writeInitString(ostream& o)const
  FPGroup getFPGroup()const
  OID getOID() const
private:
  FPGroup theGroup;
  OID oid;
  int nilpotClass;
};
```

———-

17.3.3 class PWord

— class PWord —

```
class PWord : public PObject, protected FEData
{
public:
  PWord()
  PWord( const PGroup& g, const Word& w,  OID ow) :
    theGroup( g ),
    theWord( w ),
    oidW( ow )
  Chars getDBName() const
  void printObjectName( ostream& o ) const;
  Chars getObjectName(  ) const;
  void addPackagesToMenu(  );
  Type getObjectType( )const
```

```
  static const Type thePGroupType;
  static Type type( )
  Chars ipcFileName( ) const
  void writeInitString(ostream& o)const;
private:
  PGroup theGroup;
  Word theWord;
  OID oidW;
};
```

———-

17.3.4 class PSubgroup

— class PSubgroup —

```
class PSubgroup : public PObject, protected FEData
{
public:
  PSubgroup()
  PSubgroup( const PGroup& g, const VectorOf<Word>& v, OID os) :
    theGroup( g ),
    theSubgroup( v ),
    oidS( os )
  Chars getDBName() const
  void printObjectName( ostream& o ) const;
  Chars getObjectName(  ) const;
  void addPackagesToMenu(  );
  Type getObjectType( )const
  static const Type thePGroupType;
  static Type type( )
  Chars ipcFileName( ) const
  void writeInitString(ostream& o)const;
private:
  PGroup theGroup;
  VectorOf<Word> theSubgroup;
  OID oidS;
};
```

———-

17.3.5 class PMap

— class PMap —

```
class PMap : public PObject, protected FEData
{
```

```
public:
  PMap()
  PMap( const Map& m, const PGroup& d, const PGroup& r, OID om) :
    theMap( m ),
    oidM( om ),
    theDomain( d ),
    theRange( r )
  Chars getDBName() const
  void printObjectName( ostream& o ) const;
  Chars getObjectName(  ) const;
  void addPackagesToMenu(  );
  Type getObjectType( )const
  static const Type thePGroupType;
  static Type type( )
  Chars ipcFileName( )const
  void writeInitString(ostream& o)const;
protected:
  Map theMap;
  PGroup theDomain;
  PGroup theRange;
  OID oidM;
};
```

———-

17.3.6 class PHomo

— class PHomo —

```
class PHomo : public PMap
{
public:
  PHomo(): PMap()
  PHomo( const Map& m, const PGroup& d, const PGroup& r, OID om) :
    PMap( m, d,  r,  om)
  Chars getDBName() const
  void printObjectName( ostream& o ) const;
  Chars getObjectName(  ) const;
  void addPackagesToMenu(  );
  Type getObjectType( )const
  static const Type thePGroupType;
  static Type type( )
  Chars ipcFileName( )const
};
```

———-

17.3.7 class PWordWord

— class PWordWord —

```
class PWordWord : public PObject, protected FEData
{
public:

  PWordWord()
  PWordWord( const PGroup& G, const Word& w1, const Word& w2,
       OID ow1, OID ow2) :
    theGroup( G ),
    theWord1( w1 ),
    theWord2( w2 ),
    oidW1( ow1 ),
    oidW2( ow2 )
  Chars getDBName() const
  void printObjectName( ostream& o ) const;
  Chars getObjectName(  ) const;
  void addPackagesToMenu(  );
  Type getObjectType( )const
  static const Type thePGroupType;
  static Type type( )
  Chars ipcFileName( )const
  void writeInitString(ostream& o)const;
private:
  PGroup theGroup;
  Word theWord1;
  Word theWord2;
  OID oidW1;
  OID oidW2;
};
```

———-

17.3.8 class PSubgroupSubgroup

— class PSubgroupSubgroup —

```
class PSubgroupSubgroup : public PObject, protected FEData
{
public:

  PSubgroupSubgroup() { }
  PSubgroupSubgroup( const PGroup& G, const VectorOf<Word>& s1,
                     const VectorOf<Word>& s2, OID os1, OID os2) :
    theGroup( G ),
    theSubgroup1( s1 ),
```

```
    theSubgroup2( s2 ),
    oidS1( os1 ),
    oidS2( os2 )
  Chars getDBName() const
  void printObjectName( ostream& o ) const;
  Chars getObjectName(  ) const;
  void addPackagesToMenu(  );
  Type getObjectType( )const
  static const Type thePGroupType;
  static Type type( )
  Chars ipcFileName( ) const
  void writeInitString(ostream& o)const;
private:
  PGroup theGroup;
  VectorOf<Word> theSubgroup1;
  VectorOf<Word> theSubgroup2;
  OID oidS1;
  OID oidS2;
};
```

———-

17.3.9 class PSubgroupWord

— class PSubgroupWord —

```
class PSubgroupWord : public PObject, protected FEData
{
public:
  PSubgroupWord()
  PSubgroupWord( const PGroup& G, const VectorOf<Word>& s,  const Word& w,
     OID os, OID ow) :
    theGroup( G ),
    theSubgroup( s ),
    theWord( w ),
    oidS( os ),
    oidW( ow )
  Chars getDBName() const
  void printObjectName( ostream& o ) const;
  Chars getObjectName(  ) const;
  void addPackagesToMenu(  );
  Type getObjectType( )const
  static const Type thePGroupType;
  static Type type( )
  Chars ipcFileName( ) const
  void writeInitString(ostream& o)const;
private:
  PGroup theGroup;
```

```
  VectorOf<Word> theSubgroup;
  Word theWord;
  OID oidS;
  OID oidW;
};

PObject* getPObjectFromType( Type t);
PObject* getPObjectFromInt( int i);
```

———-

17.4 Packages/include/PackagesSMApps.h

— **PackagesSMApps.h** —

```
#include "Menu.h"
#include "fastProblems.h"
#include "SMFPGroup.h"
#include "PackagesData.h"
#include "PackagesObject.h"
```

———-

17.4.1 class PackageBlackBox

— **class PackageBlackBox** —

```
class PackageBlackBox {
public:
  PackageBlackBox( PObject* po, const Chars&, const ListOf<Chars>& pl );
  ~PackageBlackBox()
  bool start();
  bool finished();
  bool getLine( Chars& line );
private:
  friend class RunPackageARCer;
  BlackBox package;
  FPGroup theGroup;
  Chars initString;
  Chars theCommand;
  PObject* theObject;
  ListOf<Chars> paramList;
};
```

———-

17.4.2 class AddPackage

— class AddPackage —

```
class AddPackage : public Supervisor
{
public:
 AddPackage(const Chars& n = Chars(), const Chars& c = Chars(),
      int cs = 0, Type os = Type(Type::notype()),
      const ListOf<ParamRecord>& pl = ListOf<ParamRecord>() );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  Chars theName;
  Chars theCommand;
  int checkinSelect;
  Type objectSelect;
  ListOf<ParamRecord> paramList;
};
```

———-

17.4.3 class EditPackage

— class EditPackage —

```
class EditPackage : public Supervisor
{
public:
 EditPackage( Type id = Type( Type::notype() ), int i = 0,
              Chars action = Chars() );
 ~EditPackage()
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  PObject* theObject;
  int theIndex;
  Chars theAction;
};
```

———-

17.4.4 class RunPackageARCer

— class RunPackageARCer —

```
class RunPackageARCer : public ARCer
{
public:
  RunPackageARCer( ComputationManager& boss, PObject* po, const Chars& com,
       const ListOf<Chars>& pl ) :
    ARCer( boss ),
    package(po,com, pl)
  Chars getFileName() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  PackageBlackBox package;
  File file;
};
```

———-

17.4.5 class RunPackage

— class RunPackage —

```
class RunPackage : public Supervisor
{
public:
  RunPackage( SMFPGroup& G, int id = 0);
  RunPackage( SMWord& w, int id = 0);
  RunPackage( SMSubgroup& s, int  id = 0);
  RunPackage( SMMap& m, int id = 0);
  RunPackage( SMHomomorphism& m, int id = 0);
  RunPackage( SMWord& ,SMWord& , int id = 0);
  RunPackage( SMSubgroup& ,SMSubgroup& , int id = 0);
  RunPackage( SMSubgroup& ,SMWord& , int id = 0);
  void initialize();
  ~RunPackage()
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( )
  void readline(istream& in, char* line, int blength);
private:
  Chars paramToChars(Chars name, PValue value,
                     ParamRecord::PARAM_TYPE t) const ;
  void trimall( char* s, int max = 1000)const;
```

```
  bool actionParser(char* ) const;
  void sendOutput();
  RunPackageARCer* arcer;
  SMFPGroup& theGroup;
  int theID;
  Chars packageName;
  PackageRecord thePackageRecord;
  PObject* packageObject;
  ListOf<ParamRecord> params;
  bool started;
  bool useGlobalLink;
  streampos pos;
  File file;
};
```

———-

18 The Polynomial classes

18.1 Polynomial/include/PBTree.h

— **PBTree.h** —

```
#include "global.h"

template <class Key, class Value> class PBTree;
template <class Key, class Value> class PBTreeIterator;

template <class Key, class Value> class PBTreePage
{
  friend class PBTree<Key,Value>;
  friend class PBTreeIterator<Key,Value>;
public:
  PBTreePage(int order) : theOrder(order), numOfKeys(0), parentLink(NULL)
  ~PBTreePage( )
private:
  int theOrder;
  int numOfKeys;
  Key **keys;
  Value **values;
  PBTreePage **links;
  PBTreePage *parentLink;
  int numOfParentLink;
};

template <class Key, class Value> class PBTree;
```

```
template <class Key, class Value>
ostream& operator < ( ostream& ostr, const PBTree<Key,Value>& T )

template <class Key, class Value>
istream& operator > ( istream& ostr, const PBTree<Key,Value>& T )

template <class Key, class Value> class PBTree
{
  friend class PBTreeIterator<Key,Value>;
public:
  PBTree( int order = 6 ) : theOrder( order )
  PBTree( const PBTree& );
  PBTree& operator = ( const PBTree& )
  ~PBTree( )
  bool remove( const Key& key );
  void insert( const Key& key, const Value& value );
  Value* search( const Key& key );
  friend ostream& operator << ( ostream& ostr, const PBTree& T )
  friend ostream& operator < <Key,Value>( ostream& ostr, const PBTree& T );
  friend istream& operator > <Key,Value>( istream& ostr, const PBTree& T );
  friend bool operator == ( const PBTree& T,const PBTree& T1 )
  void printAll();
protected:
  virtual void theKeyIsFound( const Key& key, Value& value )
  bool search( const Key& key, const PBTreePage<Key,Value>& searchPage,
         PBTreePage<Key,Value> **keyPage, int& position );
  void deleteKey( PBTreePage<Key,Value> *page, int position );
  void deleteAll( )
  void deleteAllPages( PBTreePage<Key,Value> *page );
private:
  int theOrder;
  PBTreePage<Key,Value> *root;
};

template <class Key, class Value> class PBTreeIterator
{
  friend class PBTree<Key,Value>;
public:
  PBTreeIterator( const PBTree<Key,Value>& T ) : tree( T )
  bool done( )
  void reset( );
  Value getValue( )
  Key getKey( )
  bool next( );
private:
  const PBTree<Key,Value>& tree;
  PBTreePage<Key,Value> *page;
  int linkNumber;
  bool bDone;
  Key *key;
```

```
  Value *value;
};
```

———-

18.2 Polynomial/include/Polynomial.h

— **Polynomial.h** —

```
#include "Integer.h"
#include "Rational.h"
#include "RingParser.h"
#include "IStreamPoll.h"
#include "PBTree.h"

template <class R> class Polynomial;
template <class R> class MonomialParser;
template <class R> class PolynomialParser;
template <class R> class Monomial;

template <class R>
ostream& operator << ( ostream& ostr, const Monomial<R>& M )

template <class R>
IStreamPoll operator >> ( istream& istr, Monomial<R>& M )

template <class R>
ostream& operator < ( ostream& ostr, const Monomial<R>& M )

template <class R>
istream& operator > ( istream& istr, Monomial<R>& M )

template <class R> class Monomial
{
  friend class Polynomial<R>;
public:
  Monomial( const char * );
  Monomial( R coef = 0, int numOfVars = 0, const int *powersOfVars = NULL );
  Monomial( const Monomial& );
  ~Monomial( )
  bool operator == ( const Monomial& ) const;
  Monomial& operator = ( const Monomial& );
  Monomial operator - () const;
  Monomial operator * ( const Monomial& ) const;
  Monomial operator + ( const Monomial& ) const;
  Monomial& operator += ( const Monomial& );
  int compare( const Monomial& ) const;
  R getCoefficient( ) const
```

```
  void setCoefficient( const R& c )
  int getNumberOfVariables() const
  int getPowerOfVariable( int i ) const;
  void setPowerOfVariable( int i, int v);
  friend ostream& operator << <R> ( ostream& ostr, const Monomial<R>& M );
  friend IStreamPoll operator >> <R> ( istream& istr, Monomial<R>& M );
  friend ostream& operator < <R> ( ostream& ostr, const Monomial<R>& M );
  friend istream& operator > <R> ( istream& istr, Monomial<R>& M );
private:
  R theCoefficient;
  int numberOfVariables;
  int *powersOfVariables;
  void printOn( ostream& ) const;
  Monomial readFrom( istream& istr, Chars& errMesg )
};

template <class R> class Polynomial;

template <class R>
IStreamPoll operator >> ( istream& istr, Polynomial<R>& P )

template <class R>
class Polynomial : public PBTree< Monomial<R>, Monomial<R> >
{
public:
  Polynomial( ) : PBTree< Monomial<R>, Monomial<R> >( )
  Polynomial( const char*);
  Polynomial( const Monomial<R>& );
  Polynomial( const Polynomial& );
  Polynomial( const R& );
  bool operator == ( const Polynomial& ) const;
  bool operator != ( const Polynomial& ) const;
  Polynomial& operator = ( const Polynomial& );
  Polynomial& operator += ( const Polynomial& );
  Polynomial& operator -= ( const Polynomial& );
  Polynomial& operator *= ( const Polynomial& );
  Polynomial operator + ( const Polynomial& ) const;
  Polynomial operator - ( const Polynomial& ) const;
  Polynomial operator * ( const Polynomial& ) const;
  Polynomial operator - ( ) const;
  bool isIdentity( ) const;
  int degree( ) const;
  int numberOfMonomials( ) const;
  friend ostream& operator << ( ostream& ostr, const Polynomial& P )
  friend IStreamPoll operator >> <R> ( istream& istr, Polynomial<R>& P );
private:
  void printOn( ostream& ) const;
  Polynomial readFrom( istream& istr, Chars& errMesg )
  virtual void theKeyIsFound( const Monomial<R>& key, Monomial<R>& value )
};
```

```
template <class R>
struct PolynomialIterator : public PBTreeIterator< Monomial<R>, Monomial<R> >
```

———-

18.3 Polynomial/include/RingParser.h

— **RingParser.h** —

```
#include "WordParser.h"
#include "Polynomial.h"

template <class R> class PolynomialParser;
template <class R> class Monomial;
template <class R> class Polynomial;

template <class R> class RingEltParser : public WordParser
{
public:
  RingEltParser(istream &istr) : WordParser(istr)
  void parseWhiteSpace( );
  R parseRingElt( Chars& errMesg );
};

template <class R> class MatrixParser : public RingEltParser<R> {
public:
  MatrixParser(istream &istr) : RingEltParser<R>(istr)
  bool parseMatrix( Chars& errMesg, R ***M, int& MSize );
private:
  VectorOf<R> parseMatrixRow( Chars& errMesg );
  void deleteMatrix( R ***M, int MSize );
};

template <class R> class MonomialParser : public RingEltParser<R>
{
  friend class PolynomialParser<R>;
public:
  MonomialParser(istream &istr) : RingEltParser<R>(istr)
  Monomial<R> parseMonomial( Chars& errMesg )
private:
  static const int maxNumberOfVariables = 100;
  Monomial<R> getMonomial( Chars& errMesg, bool& isLastMonomial );
};

template <class R> class PolynomialParser : public MonomialParser<R>
{
public:
```

```
  PolynomialParser(istream &istr) : MonomialParser<R>(istr)
  Polynomial<R> parsePolynomial( Chars& errMesg );
private:
};
```

———-

19 The Session Manager classes

19.1 SessionManager/include/AlgebraicObject.h

— **AlgebraicObject.h** —

```
#include "Chars.h"
#include "SMObject.h"
#include "InformationCenter.h"
```

———-

19.1.1 class AlgebraicObject

— **class AlgebraicObject** —

```
class AlgebraicObject : public SMObject
{
public:
  AlgebraicObject(const Chars heritage) : myRoots( heritage )
  Chars heritage( ) const
  virtual       InformationCenter* infoCenter()
  virtual const InformationCenter* infoCenter() const
private:
  const Chars myRoots;
};
```

———-

19.2 SessionManager/include/ARCer.h

— **ARCer.h** —

```
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <fstream.h>
```

```
#include <iostream.h>
#include <sys/wait.h>
#include "MagnusHome.h"
#include "ComputationManager.h"
```

———-

19.2.1 class ARCer

— class ARCer —

```
class ARCer
{
public:
  ARCer( ComputationManager& boss, const int delay = 1 );
  bool isInterrupted()
  int getPid( )
  int delay( )
  virtual bool takeControl( );
  virtual void terminate( );
  virtual void runComputation( ) = 0;
  virtual void writeResults( ostream& ) = 0;
  virtual void readResults( istream& ) = 0;
  Chars startDir;
protected:
  int pid;
  int theDelay;
  bool bInterrupted;
  ComputationManager& theBoss;
};
```

———-

19.2.2 class ExternalARCer

— class ExternalARCer —

```
class ExternalARCer : public ARCer
{
public:
  ExternalARCer( ComputationManager& boss,
     const char* cmdline,
     const int delay = 1 );
  virtual bool takeControl( );
  virtual void writeArguments( ostream& ) = 0;
  virtual void readResults( istream& ) = 0;
private:
```

```
  void writeResults( ostream& )
  void runComputation( );
  Chars theCmdLine;
  Chars argFN;
  Chars resultFN;
};
```

———-

19.2.3 class ARCer2

— class ARCer2 —

```
class ARCer2 : public ARCer
{
public:
  enum State { RUNNING, READING, FINISHED };
  ARCer2( ComputationManager& boss, const int delay = 1 );
  ~ARCer2( )
  virtual bool takeControl( );
  void terminate( );
  virtual bool readResults2( istream& ) = 0;
private:
  State state;
  ifstream* in;
};
```

———-

19.3 SessionManager/include/ARC.h

— ARC.h —

```
#include <iostream.h>
```

———-

19.3.1 class ARC

— class ARC —

```
class ARC
{
public:
  ARC( ) : theARC( 0 )
```

```
  operator int ( ) const
  ARC operator + ( const ARC& arc ) const
  ARC operator - ( const ARC& arc ) const
  ARC operator += ( const ARC& arc )
  ARC operator -= ( const ARC& arc )
  inline friend ostream& operator << ( ostream& ostr, const ARC& arc )
protected:
  friend class ComputationManager;
  friend class Supervisor;
  ARC( int i ) : theARC( i )
private:
  int theARC;
};
```

———-

19.3.2 struct ZeroARC

— **struct ZeroARC** —

```
struct ZeroARC : public ARC
```

———-

19.3.3 struct OneARC

— **struct OneARC** —

```
struct OneARC : public ARC
```

———-

19.4 SessionManager/include/ARCSlotID.h

— **ARCSlotID.h** —

```
#include <iostream.h>
```

———-

19.4.1 class ARCSlotID

— class ARCSlotID —

```
class ARCSlotID
{
public:
  bool operator == ( const ARCSlotID& asi ) const
  bool operator != ( const ARCSlotID& asi ) const
  inline friend ostream& operator << ( ostream& ostr, const ARCSlotID& asi )
  int unwrap( ) const
protected:
  friend class ComputationManager;
  friend class Supervisor;
  friend class EnumeratorSupervisor;
  ARCSlotID( int i ) : theARCSlotID( i )
  ARCSlotID( );
  int theARCSlotID;
};
```

———-

19.4.2 struct ThisARCSlotID

— struct ThisARCSlotID —

```
struct ThisARCSlotID : public ARCSlotID
};
```

———-

19.5 SessionManager/include/BaseProperties.h

— BaseProperties.h —

```
#include "Property.h"

#include "Associations.h"
#include "QuickAssociations.h"
#include "Chars.h"
#include "APofFreeGroups.h"
#include "FreeByCyclic.h"
#include "MSCGroup.h"
#include "Automorphism.h"
#include "HNNExtOfFreeGroup.h"
#include "NilpotentGroup.h"
```

```
#include "CosetEnumerator.h"
#include "DFSA.h"
#include "DiffMachine.h"
#include "KBMachine.h"
#include "DecomposeInSubgroup.h"
#include "AbelianSGPresentation.h"
#include "SGOfNilpotentGroup.h"
#include "PolyWord.h"
```

———-

19.5.1 class NoData

— class NoData —

```
class NoData {};

inline istream& operator>>( const istream&, const NoData& )
inline ostream& operator<<( const ostream&, const NoData& )
```

———-

19.5.2 class NoDataProperty

— class NoDataProperty —

```
class NoDataProperty : public SimpleProperty<NoData> {
public:
  NoDataProperty( const Chars& descr = Chars() )
    : SimpleProperty<NoData>(NoData(), descr)
};
```

———-

19.5.3 class BoolProperty

— class BoolProperty —

```
class BoolProperty : public SimpleProperty<bool> {
public:
  BoolProperty( const bool& data, const Chars& descr = Chars() )
    : SimpleProperty<bool>(data, descr)
};
```

———-

19.5.4 class IntProperty

— class IntProperty —

```
class IntProperty : public SimpleProperty<int> {
public:
  IntProperty( const int& data, const Chars& descr = Chars() )
    : SimpleProperty<int>(data, descr)
};

inline istream& operator>>( istream& istr, Word& w ) }
```

———-

19.5.5 class WordProperty

— class WordProperty —

```
class WordProperty : public SimpleProperty<Word> {
public:
  WordProperty( const Word& data, const Chars& descr = Chars() )
    : SimpleProperty<Word>(data,descr)
  void readFrom(istream& istr)
  void writeTo(ostream& ostr) const
};

inline istream& operator>>( istream& istr, PolyWord& w )
```

———-

19.5.6 class PolyWordProperty

— class PolyWordProperty —

```
class PolyWordProperty : public SimpleProperty<PolyWord> {
public:
  PolyWordProperty( const PolyWord& data, const Chars& descr = Chars() )
    : SimpleProperty<PolyWord>(data,descr)
  void readFrom(istream& istr)
  void writeTo(ostream& ostr) const
};
```

———-

19.5.7 class IntegerProperty

— class IntegerProperty —

```
class IntegerProperty : public SimpleProperty<Integer> {
public:
  IntegerProperty( const Integer& data, const Chars& descr = Chars() )
    : SimpleProperty<Integer>(data, descr)
};
```

———-

19.5.8 class SMFPCheckinTypeProperty

— class SMFPCheckinTypeProperty —

```
class SMFPCheckinTypeProperty : public SimpleProperty<int> {
public:
  SMFPCheckinTypeProperty( const int& data, const Chars& descr = Chars() )
    : SimpleProperty<int>(data, descr)
};
```

———-

19.5.9 class FPGroupProperty

— class FPGroupProperty —

```
class FPGroupProperty : public SimpleProperty<FPGroup> {
public:
  FPGroupProperty( const FPGroup& data, const Chars& descr = Chars() )
    : SimpleProperty<FPGroup>(data, descr)
};
```

———-

19.5.10 class FreeGroupProperty

— class FreeGroupProperty —

```
class FreeGroupProperty : public SimpleProperty<FreeGroup> {
public:
  FreeGroupProperty( const FreeGroup& data, const Chars& descr = Chars() )
    : SimpleProperty<FreeGroup>(data, descr)
```

```
};
```

———-

19.5.11 class MSCGroupPtr

— class MSCGroupPtr —

```
class MSCGroupPtr
{
public:
  MSCGroupPtr( const MSCGroupPtr& G );
  ~MSCGroupPtr();
  const MSCGroup& getMSCG() const;
private:
  friend class SimpleProperty<MSCGroupPtr>;
  friend class MSCGroupProperty;
  MSCGroupPtr( const FPGroup& G, int lmbda );
  MSCGroupPtr& operator= ( const MSCGroupPtr& G );
  friend istream& operator>>(istream& istr, MSCGroupPtr& G);
  friend ostream& operator<<(ostream& ostr, const MSCGroupPtr& G );
  FPGroup fp;
  int lambda;
  MSCGroup *msc;
private:
  MSCGroupPtr();
};
```

———-

19.5.12 class MSCGroupProperty

— class MSCGroupProperty —

```
class MSCGroupProperty : public SimpleProperty<MSCGroupPtr> {
public:
  MSCGroupProperty(const FPGroup& G, int lambda, const Chars& descr = Chars());
};
```

———-

19.5.13 class FreeByCyclicProperty

— class FreeByCyclicProperty —

```
class FreeByCyclicProperty : public SimpleProperty<FreeByCyclic> {
public:
  FreeByCyclicProperty(const FreeByCyclic& data, const Chars& descr = Chars())
    : SimpleProperty<FreeByCyclic>(data, descr)
};
```

———-

19.5.14 class AmalgProductOfFreeGroupsProperty

— class AmalgProductOfFreeGroupsProperty —

```
class AmalgProductOfFreeGroupsProperty :
  public SimpleProperty<AmalgProductOfFreeGroups> {
public:
  AmalgProductOfFreeGroupsProperty(const AmalgProductOfFreeGroups& data,
           const Chars& descr = Chars() )
    : SimpleProperty<AmalgProductOfFreeGroups>(data, descr)
};
```

———-

19.5.15 class HNNExtOfFreeGroupProperty

— class HNNExtOfFreeGroupProperty —

```
class HNNExtOfFreeGroupProperty : public SimpleProperty<HNNExtOfFreeGroup> {
public:
  HNNExtOfFreeGroupProperty( const HNNExtOfFreeGroup& data,
           const Chars& descr = Chars() )
    : SimpleProperty<HNNExtOfFreeGroup>(data, descr)
};
```

———-

19.5.16 class MapProperty

— class MapProperty —

```
class MapProperty : public SimpleProperty<Map> {
public:
  MapProperty( const Map& data, const Chars& descr = Chars() )
    : SimpleProperty<Map>(data, descr)
};
```

———-

19.5.17 class AbelianGroupProperty

— class AbelianGroupProperty —

```
class AbelianGroupProperty : public ComplexProperty< AbelianGroup > {
public:
  AbelianGroupProperty(const AbelianGroup& data, const Chars& descr = Chars())
    : ComplexProperty<AbelianGroup>(data, descr)
protected:
  void readFrom(istream& istr)
  void writeTo (ostream& ostr) const
};
```

———-

19.5.18 class NilpotentGroupProperty

— class NilpotentGroupProperty —

```
class NilpotentGroupProperty : public ComplexProperty<NilpotentGroup> {
public:
  NilpotentGroupProperty( const NilpotentGroup& data,
        const Chars& descr = Chars() )
    : ComplexProperty<NilpotentGroup>(data, descr)
protected:
  void readFrom(istream& istr);
  void writeTo (ostream& ostr);
};
```

———-

19.5.19 class NilpGroupAssocProperty

— class NilpGroupAssocProperty —

```
class NilpGroupAssocProperty :
  public ComplexProperty< AssociationsOf<int,NilpotentGroup*> >
{
public:
  NilpGroupAssocProperty( const AssociationsOf<int,NilpotentGroup*>& data,
        const Chars& descr = Chars() )
    : ComplexProperty< AssociationsOf<int,NilpotentGroup*> >(data, descr)
  NilpGroupAssocProperty( const NilpGroupAssocProperty& P );
  const NilpGroupAssocProperty& operator=( const NilpGroupAssocProperty& P );
  ~NilpGroupAssocProperty();
```

```
protected:
  void unbindAll();
  void copyAll( const NilpGroupAssocProperty& P );
  void readFrom(istream& istr);
  void writeTo(ostream& ostr) const ;
};
```

———-

19.5.20 class SubgroupGraphProperty

— class SubgroupGraphProperty —

```
class SubgroupGraphProperty : public ComplexProperty<SubgroupGraph> {
public:
  SubgroupGraphProperty( const SubgroupGraph& data,
       const Chars& descr = Chars() )
    : ComplexProperty<SubgroupGraph>(data, descr)
protected:
  void readFrom(istream& istr)
  void writeTo(ostream& ostr) const
};
```

———-

19.5.21 class PermutationRepresentationProperty

— class PermutationRepresentationProperty —

```
class PermutationRepresentationProperty :
  public ComplexProperty<PermutationRepresentation> {
public:
  PermutationRepresentationProperty( const PermutationRepresentation& data,
             const Chars& descr = Chars() )
    : ComplexProperty<PermutationRepresentation>(data, descr)
protected:
  void readFrom(istream& istr)
  void writeTo(ostream& ostr) const
};
```

———-

19.5.22 class GroupDFSAProperty

— class GroupDFSAProperty —

```
class GroupDFSAProperty : public ComplexProperty<GroupDFSA> {
public:
  GroupDFSAProperty( const GroupDFSA& data, const Chars& descr = Chars() )
    : ComplexProperty<GroupDFSA>(data, descr)
protected:
  void readFrom(istream& istr)
  void writeTo(ostream& ostr) const
};
```

———-

19.5.23 class DiffMachineProperty

— class DiffMachineProperty —

```
class DiffMachineProperty : public ComplexProperty<DiffMachine> {
public:
  DiffMachineProperty( const DiffMachine& data, const Chars& descr = Chars() )
    : ComplexProperty<DiffMachine>(data, descr)
protected:
  void readFrom(istream& istr)
  void writeTo(ostream& ostr) const
};
```

———-

19.5.24 class KBMachineProperty

— class KBMachineProperty —

```
class KBMachineProperty : public UnstorableProperty<KBMachine> {
public:
  KBMachineProperty( const KBMachine& data, const Chars& descr = Chars() )
    : UnstorableProperty<KBMachine>( data, descr )
};
```

———-

19.5.25 class DecomposeInSubgroupOfFPGroupProperty

— class DecomposeInSubgroupOfFPGroupProperty —

```
class DecomposeInSubgroupOfFPGroupProperty :
  public ComplexProperty<DecomposeInSubgroupOfFPGroup>
{
public:
  DecomposeInSubgroupOfFPGroupProperty(
    const DecomposeInSubgroupOfFPGroup& data, const Chars& descr = Chars()
  ) : ComplexProperty<DecomposeInSubgroupOfFPGroup>(data, descr)

protected:
  void readFrom(istream& istr)
  void writeTo(ostream& ostr) const
};
```

———-

19.5.26 class ListOfEndomorphismProperty

— class ListOfEndomorphismProperty —

```
class ListOfEndomorphismProperty :
  public ComplexProperty< ListOf<Endomorphism> >
{
public:
  ListOfEndomorphismProperty( const ListOf<Endomorphism> & data,
            const Chars& descr = Chars() )
    : ComplexProperty< ListOf<Endomorphism> >(data, descr)
protected:
  void readFrom(istream& istr);
  void writeTo (ostream& ostr) const;
};
```

———-

19.5.27 class ListOfAutomorphismProperty

— class ListOfAutomorphismProperty —

```
class ListOfAutomorphismProperty :
  public ComplexProperty< ListOf<Automorphism> > {
public:
  ListOfAutomorphismProperty( const ListOf<Automorphism> & data,
            const Chars& descr = Chars() )
```

```
    : ComplexProperty< ListOf<Automorphism> >(data, descr)
protected:
    void readFrom(istream& istr);
    void writeTo (ostream& ostr) const;
};
```

———-

19.5.28 class AbelianSGPresentationProperty

— class AbelianSGPresentationProperty —

```
class AbelianSGPresentationProperty :
  public ComplexProperty<AbelianSGPresentation> {
public:
  AbelianSGPresentationProperty( const AbelianSGPresentation& data,
         const Chars& descr = Chars() )
    : ComplexProperty<AbelianSGPresentation>(data, descr)
protected:
  void readFrom(istream& istr)
  void writeTo(ostream& ostr) const
};
```

———-

19.5.29 class SGOfNilpotentGroupProperty

— class SGOfNilpotentGroupProperty —

```
class SGOfNilpotentGroupProperty : public ComplexProperty<SGOfNilpotentGroup> {
public:
  SGOfNilpotentGroupProperty( const SGOfNilpotentGroup& data,
            const Chars& descr = Chars() )
    : ComplexProperty<SGOfNilpotentGroup>(data, descr)
protected:
  void readFrom(istream& istr);
  void writeTo(ostream& ostr) const;
};
```

———-

19.5.30 class SGNilpGroupAssocProperty

— class SGNilpGroupAssocProperty —

```
class SGNilpGroupAssocProperty :
  public ComplexProperty< AssociationsOf<int,SGOfNilpotentGroup*> >
{
public:
  SGNilpGroupAssocProperty(const AssociationsOf<int,SGOfNilpotentGroup*>& data,
         const Chars& descr = Chars() )
    : ComplexProperty< AssociationsOf<int,SGOfNilpotentGroup*> >(data, descr)
  SGNilpGroupAssocProperty( const SGNilpGroupAssocProperty& P );
  const SGNilpGroupAssocProperty& operator=( const SGNilpGroupAssocProperty& P );
  ~SGNilpGroupAssocProperty();
protected:
  void unbindAll();
  void copyAll( const SGNilpGroupAssocProperty& P );
  void readFrom(istream& istr);
  void writeTo(ostream& ostr) const ;
};
```

———-

19.6 SessionManager/include/ComputationManager.h

— **ComputationManager.h** —

```
#include "SMObject.h"
#include "ResourceManager.h"
#include "TheObjects.h"
#include "ARCer.h"
```

———-

19.6.1 class ComputationManager

— **class ComputationManager** —

```
class ComputationManager : public SMObject, public ResourceManager
{
  friend class ARCer;
  friend class RelatorEnumeratorARCer;
  friend class ExternalARCer;
  friend class Supervisor;
public:
  enum State { UNSTARTED, RUNNING, SUSPENDED, TERMINATED };
  ComputationManager( bool display_in_fe = false );
  ~ComputationManager( );
  State state( ) const;
  bool crashed( ) const
```

```
  virtual Chars getCrashMessage( ) const
  virtual bool isSupervisor() const
  Chars helpID( const Chars& problemName, const SMFPGroup& parent ) const;
  const char* typeID( ) const;
  const IconID iconID( ) const;
  void viewStructure(ostream& ostr) const;
  void printProperties(ostream& ostr) const;
  void printDefinition(ostream& ostr) const;
  bool displayInFE( ) const
  static void printGlobalMessageTemplates(ostream& ostr);
  virtual void takeControl( ) = 0;
  bool fastResult( ) const
  bool checkForStall( );
  void attachClient( const SMObject& client );
  void detachClient( const SMObject& client );
  SetOf<OID> getClients( ) const;
  void adminStart( );
  void adminSuspend( );
  void adminResume( );
  virtual void adminTerminate( );
  virtual void start( )
  virtual void suspend( )
  virtual void resume( )
  virtual void terminate( )
protected:
  void readMessage(istream& istr);
  void resultIsFast( );
private:
  void hireArcer( ARCer* arcer );
  void terminateArcers( );
  void setCrashState( );
  ListOf<ARCer*> theArcers;
  SetOf<OID> theClients;
  bool amFast;
  bool showMe;
  State theState;
  bool isCrashed;
  bool wasStalled;
  enum MessageTag {
   VIEW_REQUEST, ARC_UPDATE, START, SUSPEND, RESUME, TERMINATE, PARAMETER
  };
};
```

———-

19.7 SessionManager/include/DatabaseManager.h

— DatabaseManager.h —

```
#include "Chars.h"
#include "Vector.h"
#include "OID.h"

enum DB2FE_MESSAGE { NO_MESSAGE,        MSG_NEW_FILENAME,
         MSG_NEW_DATABASE,  MSG_OPEN_DATABASE,
         MSG_SAVE_DATABASE, MSG_SAVE_DATABASE_AS,
         MSG_SAVE_ON_CLOSE, MSG_CLOSE_DATABASE,
         MSG_ADD_OBJECTS,   MSG_GET_OBJECTS
};

enum FE2DB_MESSAGE { DB_NO_EVENT,
         DB_NEW, DB_OPEN, DB_SAVE, DB_SAVE_AS, DB_CLOSE,
         DB_ADD_OBJECTS, DB_GET_OBJECTS,
         DB_CANCEL, DB_SUCCESS, DB_FAILURE,
         DB_YES, DB_NO, DB_SELECT_OBJECTS, DB_OBJECT_DEFINITION
};
```

———-

19.7.1 struct DBEvent

— **struct DBEvent** —

```
struct DBEvent {
  FE2DB_MESSAGE key;
  Chars str;
  DBEvent() : key(DB_NO_EVENT), str() {}
};

ostream& operator<<( ostream& ostr, const DBEvent& event );
```

———-

19.7.2 class DBState

— **class DBState** —

```
class DBState
{
public:
  DBState( ) : theNextState(0)
  DBState( const DBState& state );
  const DBState& operator=( const DBState& state );
  virtual ~DBState();
  virtual void init( )
  virtual void handleEvent( const DBEvent& event ) = 0;
```

```
  const DBState* nextState( ) const
  virtual DBState* clone() const = 0;
  virtual bool isFinal( ) const = 0;
  virtual DB2FE_MESSAGE initMessage() const
  virtual void printOn( ostream& ostr ) const
protected:
  DBState *theNextState;
};
```

———-

19.7.3 class MainState

— class MainState —

```
class MainState : public DBState
{
public:
  MainState( ) : DBState()
};

class IntermediateState : public DBState
{
public:
  IntermediateState( ) : theFailure(0), theSecondState(0)
  IntermediateState( DBState *onFailure, DBState *secondState )
    : theFailure(onFailure), theSecondState(secondState)
  IntermediateState( const IntermediateState& state );
  ~IntermediateState();
  const IntermediateState& operator=( const IntermediateState& state );
  bool isFinal( ) const
  void printOn( ostream& ostr ) const
protected:
  void handleFailure();
  void useSecondState( DBState *state );
  DBState *theFailure;
  DBState *theSecondState;
};
```

———-

19.7.4 class DatabaseClosed

— class DatabaseClosed —

```
class DatabaseClosed : public MainState
{
```

```
public:
  DatabaseClosed()
  bool isFinal( ) const
  DB2FE_MESSAGE initMessage() const
  void handleEvent( const DBEvent& event );
  DBState *clone() const
  void printOn( ostream& ostr ) const
};
```

———-

19.7.5 class DatabaseSaved

— class DatabaseSaved —

```
class DatabaseSaved : public MainState
{
public:
  DatabaseSaved()
  bool isFinal( ) const
  void handleEvent( const DBEvent& event );
  DBState *clone() const
  void printOn( ostream& ostr ) const
};
```

———-

19.7.6 class DatabaseModified

— class DatabaseModified —

```
class DatabaseModified : public MainState
{
public:
  DatabaseModified()
  bool isFinal( ) const
  void handleEvent( const DBEvent& event );
  DBState *clone() const
  void printOn( ostream& ostr ) const
};
```

———-

19.7.7 class DatabaseCreating

— class DatabaseCreating —

```
class DatabaseCreating : public IntermediateState
{
public:
  DatabaseCreating()
  DatabaseCreating( DBState *onFailure, DBState *secondState = 0 )
    : IntermediateState(onFailure,secondState)
  DB2FE_MESSAGE initMessage() const
  void handleEvent( const DBEvent& event );
  DBState *clone() const
  void printOn( ostream& ostr ) const
};
```

———-

19.7.8 class DatabaseOpening

— class DatabaseOpening —

```
class DatabaseOpening : public IntermediateState
{
public:
  DatabaseOpening()
  DatabaseOpening( DBState *onFailure, DBState *secondState = 0 )
    : IntermediateState(onFailure,secondState)
  DB2FE_MESSAGE initMessage() const
  void handleEvent( const DBEvent& event );
  DBState *clone() const
  void printOn( ostream& ostr ) const
};
```

———-

19.7.9 class DatabaseSaving

— class DatabaseSaving —

```
class DatabaseSaving : public IntermediateState
{
public:
  DatabaseSaving()
  DatabaseSaving( DBState *onFailure, DBState *secondState = 0 )
    : IntermediateState(onFailure,secondState)
```

```
  DB2FE_MESSAGE initMessage() const
  void handleEvent( const DBEvent& event );
  DBState *clone() const
  void printOn( ostream& ostr ) const
};
```

———-

19.7.10 class DatabaseSavingAs

— class DatabaseSavingAs —

```
class DatabaseSavingAs : public IntermediateState
{
public:
  DatabaseSavingAs()
  DatabaseSavingAs( DBState *onFailure, DBState *secondState = 0 )
    : IntermediateState(onFailure,secondState)
  DB2FE_MESSAGE initMessage() const
  void handleEvent( const DBEvent& event );
  DBState *clone() const
  void printOn( ostream& ostr ) const
};
```

———-

19.7.11 class DatabaseClosing

— class DatabaseClosing —

```
class DatabaseClosing : public IntermediateState
{
public:
  DatabaseClosing()
  DatabaseClosing( DBState *onFailure, DBState *secondState = 0 )
    : IntermediateState(onFailure,secondState)
  DB2FE_MESSAGE initMessage() const
  void handleEvent( const DBEvent& event );
  DBState *clone() const
  void printOn( ostream& ostr ) const
};
```

———-

19.7.12 class DatabaseAddingObjects

— class DatabaseAddingObjects —

```
class DatabaseAddingObjects : public IntermediateState
{
public:
  DatabaseAddingObjects()
  DatabaseAddingObjects( DBState *onFailure, DBState *secondState = 0 )
    : IntermediateState(onFailure,secondState)
  DB2FE_MESSAGE initMessage() const
  void handleEvent( const DBEvent& event );
  DBState *clone() const
  void printOn( ostream& ostr ) const
};
```

———-

19.7.13 class DatabaseGettingObjects

— class DatabaseGettingObjects —

```
class DatabaseGettingObjects : public IntermediateState
{
public:
  DatabaseGettingObjects()
  DatabaseGettingObjects( DBState *onFailure, DBState *secondState = 0)
    : IntermediateState(onFailure,secondState)
  DB2FE_MESSAGE initMessage() const
  void handleEvent( const DBEvent& event );
  DBState *clone() const
  void printOn( ostream& ostr ) const
};

inline DBState *copyState( const DBState *state )
inline ostream& operator<<(ostream& ostr, const DBState& state)
```

———-

19.7.14 class DatabaseObjectCategory

— class DatabaseObjectCategory —

```
class DatabaseObjectCategory {
public:
  enum Category
```

```
  {
    NO_CATEGORY = -1,
    FP, FREE, ABELIAN, NILPOTENT, SMALL_CANCELLATION, ONE_RELATOR,
    AP_FREE, AP_FREE_CYCLIC, FREE_NILPOTENT, HNN_FREE,
    ONE_RELATOR_WITH_TORSION,
    SUBGROUP, WORD, MAP, MAP2, HOMOMORPHISM, HOMOMORPHISM2,
    EQUATION, EQUATION2, SET_OF_WORDS, VECTOR_OF_WORDS,
  };

  DatabaseObjectCategory(Category cath = NO_CATEGORY) :theCategory(cath)
  Chars str() const
  Category id() const
  static Chars str(const Category cath);
  static Chars getSMObjectCategory( class SMObject *smo );
  friend istream& operator>>(istream& istr, DatabaseObjectCategory& cath )
  friend ostream& operator<<(ostream& ostr,
           const DatabaseObjectCategory& cath )
private:
  Category theCategory;
};
```

———-

19.7.15 class DatabaseObjectSmith

— class DatabaseObjectSmith —

```
class DatabaseObjectSmith {
public:
  void checkinObjectFromDatabase(istream& istr);
private:
  void reset();
  void checkinObjectProperties( istream& istr );
  void checkinObjectDefinition( istream& istr );
  SMObject* checkinFPGroup(istream& istr, DatabaseObjectCategory type );
  SMObject* checkinWord(istream& istr);
  SMObject* checkinSubgroup(istream& istr);
  SMObject* checkinSetOfWords(istream& istr);
  SMObject* checkinVectorOfWords(istream& istr);
  SMObject* checkinMap(istream& istr, DatabaseObjectCategory type);
  SMObject* checkinHomomorphism(istream& istr, DatabaseObjectCategory type);
  SMObject* checkinEquation(istream& istr, DatabaseObjectCategory type);
  void parseError(const Chars& str) { theError = str; }
  Chars theError;
  Chars theObjectName;
  VectorOf<SMObject *> theExperiment;
  ListOf<OID> dependencies;
```

```
};
```

———-

19.7.16 class DatabaseManager

— class DatabaseManager —

```
class DatabaseManager
{
public:
  DatabaseManager();
  ~DatabaseManager( );
  void takeControl( );
  enum ManagerState { WORKING, STOPPING, STOPPED };
  ManagerState state() const
  Chars getFileName() const
  Chars getSecondaryFileName() const;
  static void printGlobalMessageTemplate( ostream& ostr );
  void forceToFinish();
  void readMessage( istream& istr );
private:
  friend void DatabaseCreating::handleEvent( const DBEvent& event );
  friend void DatabaseSaving::handleEvent( const DBEvent& event );
  friend void DatabaseSavingAs::handleEvent( const DBEvent& event );
  void setFileName( const Chars& filename );
private:
  DatabaseManager( const DatabaseManager& );
  DatabaseManager& operator=( const DatabaseManager& );
  void postMessage( const DB2FE_MESSAGE message ) const;
  static void printDatabaseObjectCategories( ostream& ostr );
  DBState *theCurrentState;
  Chars  theFileName;
  ManagerState theExternalState;
  DBEvent event;
};

extern DatabaseManager TheDatabaseManager;
```

———-

19.8 SessionManager/include/FEData.h

— FEData.h —

```
#include <iostream.h>
#include "Chars.h"
```

```
#include "OID.h"
#include "ARCSlotID.h"
```

———-

19.8.1 class FEData

— class FEData —

```
class FEData
{
private:
```

———-

19.8.2 struct ExpressionRep

— struct ExpressionRep —

```
struct ExpressionRep
{
 ExpressionRep( ) : refs( 1 )
 ExpressionRep* gimme( )
 void getHence( )
 virtual ~ExpressionRep( )
 virtual void printOn(ostream& ostr) const = 0;
 int refs;
};
```

———-

19.8.3 struct KeyRep

— struct KeyRep —

```
struct KeyRep : public ExpressionRep
{
 KeyRep(int i);
 KeyRep(int i, const char* t);
 KeyRep(OID oid, const char* t);
 KeyRep(ExpressionRep* se, const char* t);
 ~KeyRep( )
 void printOn(ostream& ostr) const;
 const int index;
 const char* text;
 bool isConstant;
```

```
 ExpressionRep* subExpression;
};
```

———-

19.8.4 struct JoinRep

— struct JoinRep —

```
struct JoinRep : public ExpressionRep
{
 JoinRep(ExpressionRep* a1, ExpressionRep* a2, const char* j);
 ~JoinRep( );
 void printOn(ostream& ostr) const;
 ExpressionRep* lhs;
 ExpressionRep* rhs;
 const char* junctor;
};
```

———-

19.8.5 struct NotRep

— struct NotRep —

```
struct NotRep : public ExpressionRep
{
 NotRep(ExpressionRep* a);
 ~NotRep( );
 void printOn(ostream& ostr) const;
 ExpressionRep* expr;
};
```

———-

19.8.6 struct NameRep

— struct NameRep —

```
struct NameRep : public ExpressionRep
{
 NameRep(ExpressionRep* a);
 ~NameRep( );
 void printOn(ostream& ostr) const;
 ExpressionRep* expr;
```

```
  };
protected:
  class Expression
  {
  public:
   Expression(int i) : theRep( new KeyRep( i ) )
   Expression( const Expression& E )
   ~Expression( )
   Expression operator == ( const Expression& E ) const
   Expression operator != ( const Expression& E ) const
   Expression operator && ( const Expression& E ) const
   Expression operator || ( const Expression& E ) const
   Expression operator ! ( ) const
   Expression operator > ( const Expression& E ) const
   Expression operator >= ( const Expression& E ) const
   Expression operator < ( const Expression& E ) const
   Expression operator <= ( const Expression& E ) const
   friend ostream& operator << ( ostream& ostr, const Expression& E);
   Expression( ExpressionRep* rep ) : theRep( rep )
   ExpressionRep* theRep;
  private:
   Expression join( const Expression& E, const char* junctor ) const;
  };
```

———-

19.8.7 class Key

— **class Key** —

```
  class Key : public Expression
  {
  public:
   Key( const Key& K ) : Expression( K.theRep->gimme() )
  protected:
   Key(int i) : Expression( new KeyRep( i ) )
   Key(int i, const char* t) : Expression( new KeyRep( i, t ) )
   Key(OID oid, const char* t) : Expression( new KeyRep( oid, t ) )
   Key(const Key& K, const char* t)
     : Expression( new KeyRep( K.theRep->gimme(), t ) )
  };
```

———-

19.8.8 struct DataPair

— **struct DataPair** —

```
struct DataPair
{
 DataPair( const Key& key, const Expression& datum )
    : theKey( key ), theDatum( datum )
 bool operator == (const DataPair& dp) const
 DataPair& operator = (const DataPair& dp);
  friend ostream& operator << ( ostream& ostr, const DataPair& dp);
private:
 Key theKey;
 Expression theDatum;
};
```

———-

19.8.9 struct Text

— **struct Text** —

```
struct Text
{
 Text(const char* t = "");
 Text(const Expression& E);
 operator Chars( ) const;
 Text& operator + (const Text& t);
 Text& operator + (const Expression& E);
 friend ostream& operator << (ostream& ostr, const Text& t);
private:
 Text(const Text&);
 ostrstream ostrstr;
};
```

———-

19.8.10 struct True

— **struct True** —

```
struct True : public Key
{ True( ) : Key( 1 ) };
```

———-

19.8.11 struct False

— **struct False** —

```
struct False : public Key
{ False( ) : Key( 0 ) };
```

———-

19.8.12 struct Object

— **struct Object** —

```
struct Object : public Key
{
 Object(int i) : Key( i, 0 )
 Object(OID o) : Key( o, 0 )
};
```

———-

19.8.13 struct CheckinType

— **struct CheckinType** —

```
struct CheckinType : public Key
{
 CheckinType(int i) : Key( i, "checkin_type" )
 CheckinType(OID o) : Key( o, "checkin_type" )
 CheckinType(const Key& K) : Key( K, "checkin_type" )
};
```

———-

19.8.14 struct IsHomo

— **struct IsHomo** —

```
struct IsHomo : public Key
{
 IsHomo(int i) : Key( i, "homo" )
 IsHomo(OID o) : Key( o, "homo" )
};
```

———-

19.8.15 struct IsIso

— struct IsIso —

```
struct IsIso : public Key
{
 IsIso(int i) : Key( i, "iso" )
 IsIso(OID o) : Key( o, "iso" )
};
```

———-

19.8.16 struct IsAuto

— struct IsAuto —

```
struct IsAuto : public Key
{
 IsAuto(int i) : Key( i, "auto" )
 IsAuto(OID o) : Key( o, "auto" )
};
```

———-

19.8.17 struct Parent

— struct Parent —

```
struct Parent : public Key
{
 Parent(int i) : Key( i, "parent" )
 Parent(OID o) : Key( o, "parent" )
};
```

———-

19.8.18 struct ParentGroup

— struct ParentGroup —

```
struct ParentGroup : public Key
{
 ParentGroup(int i) : Key( Parent(i), "parent" )
 ParentGroup(OID o) : Key( Parent(o), "parent" )
};
```

———-

19.8.19 struct Domain

— struct Domain —

```
struct Domain : public Key
{
 Domain(int i) : Key( i, "domain" )
 Domain(OID o) : Key( o, "domain" )
};
```

———-

19.8.20 struct Range

— struct Range —

```
struct Range : public Key
{
 Range(int i) : Key( i, "range" )
 Range(OID o) : Key( o, "range" )
};
```

———-

19.8.21 struct Oid

— struct Oid —

```
struct Oid : public Key
{
 Oid(int i) : Key( i, "oid" )
 Oid(OID o) : Key( o, "oid" )
};
```

———-

19.8.22 struct Name

— struct Name —

```
struct Name : public Expression
{
 Name(int i);
 Name(OID o);
 Name(const Expression& E);
};
```

———-

19.8.23 struct Link

— **struct Link** —

```
struct Link : public Text
{
  Link( const Chars& text, const Chars& problemName,
  const Chars& fileName, bool isDynamic = false );
};
```

———-

19.8.24 struct SubProblemName

— **struct SubProblemName** —

```
struct SubProblemName
{
  SubProblemName( OID oid, ARCSlotID asi )
    : theOid( oid.unwrap() ), theAsi( asi.unwrap() )
  friend ostream& operator << ( ostream& ostr, const SubProblemName& n )
private:
  int theOid;
  int theAsi;
};
```

———-

19.8.25 struct Banner

— **struct Banner** —

```
struct Banner
{
  Banner( OID oid ) : theOid( oid.unwrap() )
  friend ostream& operator << ( ostream& ostr, const Banner& b )
```

```
  private:
    int theOid;
  };
};
```

———-

19.9 SessionManager/include/GCM.h

— GCM.h —

```
#include "FPGroup.h"
#include "ComputationManager.h"
#include "Supervisor.h"
#include "GIC.h"
#include "NormalClosure.h"
#include "AbelianInvariants.h"
#include "KBModule.h"
#include "AGModule.h"
#include "ToddCoxeter.h"
#include "HToddCoxeter.h"
#include "NilpotentQuotients.h"
```

———-

19.9.1 class GCM

— class GCM —

```
class GCM : public Supervisor
{
public:
  GCM( class SMFPGroup& G );
  SMFPGroup& getSMFPGroup( ) const
  void viewStructure(ostream& ostr) const;
  Subordinate<GCM,NormalClosure> normalClosure;
  Subordinate<GCM,AbelianInvariants> abelianInvariants;
  Subordinate<GCM,AbelianPrimes> abelianPrimes;
  Subordinate<GCM,AbelianRank> abelianRank;
  Subordinate<GCM,KBSupervisor> kbSupervisor;
  Subordinate<GCM,AGSupervisor> agSupervisor;
  Subordinate<GCM,NilpotentQuotients> nilpotentQuotients;
  Subordinate<GCM,NGcomputeBasis> computeBasis;
  Subordinate<GCM,ToddCoxeter> theToddCoxeter;
  Subordinate<GCM,HToddCoxeter> ghToddCoxeter;
  void takeControl( );
```

```
  void start( );
  void terminate( );
private:
  class SMFPGroup& theSMFPGroup;
  bool didFastChecks;
};
```

———-

19.10 SessionManager/include/GIC.h

— **GIC.h** —

```
#include "FEData.h"
#include "BaseProperties.h"
#include "InformationCenter.h"
#include "OID.h"
```

———-

19.10.1 class GroupOrderProperty

— **class GroupOrderProperty** —

```
class GroupOrderProperty : public IntegerProperty {
public:
  GroupOrderProperty(const Integer& data, const Chars& descr)
    : IntegerProperty(data, descr)
  PropertyType actualType  () const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.2 class SolvedWordProblemProperty

— **class SolvedWordProblemProperty** —

```
class SolvedWordProblemProperty : public NoDataProperty {
public:
  SolvedWordProblemProperty(const Chars& descr) : NoDataProperty(descr)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.3 class FastWordProblemProperty

— class FastWordProblemProperty —

```
class FastWordProblemProperty : public NoDataProperty {
public:
  FastWordProblemProperty(const Chars& descr) : NoDataProperty(descr)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.4 class CompleteCayleyGraphProperty

— class CompleteCayleyGraphProperty —

```
class CompleteCayleyGraphProperty : public SubgroupGraphProperty {
public:
  CompleteCayleyGraphProperty(const SubgroupGraph& graph)
    : SubgroupGraphProperty(graph)
  PropertyType actualType  () const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
```

```
};
```

———-

19.10.5 class ConfluentKBMachineProperty

— class ConfluentKBMachineProperty —

```
class ConfluentKBMachineProperty : public KBMachineProperty {
public:
  ConfluentKBMachineProperty(const KBMachine& data) : KBMachineProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.6 class IsAutomaticProperty

— class IsAutomaticProperty —

```
class IsAutomaticProperty : public NoDataProperty {
public:
  IsAutomaticProperty(const Chars& descr) : NoDataProperty(descr)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.7 class Automatic_GroupDFSAProperty

— class Automatic_GroupDFSAProperty —

```
class Automatic_GroupDFSAProperty : public GroupDFSAProperty {
public:
  Automatic_GroupDFSAProperty(const GroupDFSA& data)
    : GroupDFSAProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.8 class Automatic_DiffMachineProperty

— class Automatic_DiffMachineProperty —

```
class Automatic_DiffMachineProperty : public DiffMachineProperty {
public:
  Automatic_DiffMachineProperty(const DiffMachine& data)
    : DiffMachineProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.9 class OneRelatorProperty

— class OneRelatorProperty —

```
class OneRelatorProperty : public WordProperty {
public:
  OneRelatorProperty(const Word& data) : WordProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
```

```
  static const PropertyType theTypeID;
};
```

———-

19.10.10 class OneRelatorWithTorsionProperty

— class OneRelatorWithTorsionProperty —

```
class OneRelatorWithTorsionProperty : public NoDataProperty {
public:
  OneRelatorWithTorsionProperty() : NoDataProperty()
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.11 class AbelianPresentationProperty

— class AbelianPresentationProperty —

```
class AbelianPresentationProperty : public AbelianGroupProperty {
public:
  AbelianPresentationProperty(const AbelianGroup& data,
            const Chars descr = Chars() )
    : AbelianGroupProperty(data,descr)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.12 class IsAbelianProperty

— class IsAbelianProperty —

```
class IsAbelianProperty : public NoDataProperty {
public:
  IsAbelianProperty() : NoDataProperty()
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.13 class IsFreeProperty

— class IsFreeProperty —

```
class IsFreeProperty : public NoDataProperty {
public:
  IsFreeProperty(const Chars& descr) : NoDataProperty(descr)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.14 class IsFiniteProperty

— class IsFiniteProperty —

```
class IsFiniteProperty : public NoDataProperty {
public:
  IsFiniteProperty(const Chars& descr) : NoDataProperty(descr)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
```

```
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.15 class NilpotencyClassProperty

— class NilpotencyClassProperty —

```
class NilpotencyClassProperty : public IntProperty {
public:
  NilpotencyClassProperty(const int& data) : IntProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.16 class NilpotentQuotientsProperty

— class NilpotentQuotientsProperty —

```
class NilpotentQuotientsProperty : public NilpGroupAssocProperty {
public:
  NilpotentQuotientsProperty(const AssociationsOf<int,NilpotentGroup*>& data)
    : NilpGroupAssocProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.17 class IsFreeNilpotentProperty

— class IsFreeNilpotentProperty —

```
class IsFreeNilpotentProperty : public NoDataProperty {
public:
  IsFreeNilpotentProperty() : NoDataProperty()
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.18 class ActualNilpotencyClassProperty

— class ActualNilpotencyClassProperty —

```
class ActualNilpotencyClassProperty : public IntProperty {
public:
  ActualNilpotencyClassProperty(const int& data) : IntProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.19 class IsFreeByCyclicProperty

— class IsFreeByCyclicProperty —

```
class IsFreeByCyclicProperty : public FreeByCyclicProperty {
public:
  IsFreeByCyclicProperty(const FreeByCyclic& data):FreeByCyclicProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
```

```
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.20 class MSCProperty

— class MSCProperty —

```
class MSCProperty : public MSCGroupProperty {
public:
  MSCProperty(const FPGroup& G, int lambda) : MSCGroupProperty(G,lambda)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.21 class MSCLambdaProperty

— class MSCLambdaProperty —

```
class MSCLambdaProperty : public IntProperty {
public:
  MSCLambdaProperty(const int& data, const Chars& descr = Chars())
    : IntProperty(data,descr)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.22 class APofFreeProperty

— class APofFreeProperty —

```
class APofFreeProperty : public AmalgProductOfFreeGroupsProperty {
public:
  APofFreeProperty(const AmalgProductOfFreeGroups& data)
    : AmalgProductOfFreeGroupsProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.23 class HNNofFreeProperty

— class HNNofFreeProperty —

```
class HNNofFreeProperty : public HNNExtOfFreeGroupProperty {
public:
  HNNofFreeProperty(const HNNExtOfFreeGroup& data)
    : HNNExtOfFreeGroupProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.24 class SchreierTransversalProperty

— class SchreierTransversalProperty —

```
class SchreierTransversalProperty : public PermutationRepresentationProperty {
public:
  SchreierTransversalProperty(const PermutationRepresentation& data)
    : PermutationRepresentationProperty(data)
```

```
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.25 class WordDecomposerProperty

— class WordDecomposerProperty —

```
class WordDecomposerProperty : public DecomposeInSubgroupOfFPGroupProperty {
public:
  WordDecomposerProperty(const DecomposeInSubgroupOfFPGroup& data)
    : DecomposeInSubgroupOfFPGroupProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.10.26 class GIC

— class GIC —

```
class GIC : public InformationCenter, protected FEData
{
public:
 enum AlgorithmID { NONE, AB_INV, NORM_CLOSURE, REWR_SYSTEM,
         AUT_STRUCTURE, GENETIC, FREE };
  GIC(OID group_oid);
  ~GIC( );
  void putHaveOrder( Integer the_order, Chars explanation = "",
         bool showExplanation = true );
  void putHaveSolvedWordProblem( Chars explanation,
         bool bShowExplanation = true );
  void putHaveFastWordProblem( Chars explanation,
             bool bShowExplanation = true );
```

```
void putHaveCompleteCayleyGraph(SubgroupGraph CG);
void putHaveConfluentKBMachine(KBMachine kbm);
void putIsAutomatic( Trichotomy IsAutomatic );
void putHaveAutomatic(GroupDFSA wa, DiffMachine dm);
void putIsOneRelator(const Word& relator);
void putIsOneRelatorWithTorsion(const Word& relator);
void putHaveCyclicDecomposition( const AbelianGroup& );
void putHavePrimaryDecomposition(const AbelianGroup& );
void putHaveCanonicalSmithPresentation( const AbelianGroup& );
void putIsAbelian( );
void putIsFree( bool IsFree = true, Chars explanation = "" );
void putIsFinite( bool IsFinite = true, Chars explanation = "",
      bool showExplanation = true );
void putHaveNilpotentQuotInited(const NilpotentGroup& theQuot, int theClass);
void putHaveNilpotentGroupInited(const NilpotentGroup& theGroup,
                                 Chars filename);
void putIsFreeNilpotent(const FreeGroup& F, int theClass);
void putIsNilpotent(int theClass);
void putIsFreeByCyclic(const FreeGroup& F, const Map& aut);
void putHaveMSC(FPGroup G, int lambda);
void putHaveMSCLambda( int lambda );
void putHaveAPOfFree( const AmalgProductOfFreeGroups& );
void putHaveHNNofFree( const class HNNExtOfFreeGroup& );
void putHaveSchreierTransvl(const PermutationRepresentation&);
void putHaveWordDecomposer( const class DecomposeInSubgroupOfFPGroup& );
void postSecondaryLogMessage(Chars message);
bool haveOrder( ) const;
const Integer& getOrder( ) const;
const Chars    getOrderExplanation( ) const;
const Chars    getOrderMessage( ) const;
bool haveSolvedWordProblem( ) const;
bool haveFastWordProblem( ) const;
bool haveCompleteCayleyGraph( ) const;
SubgroupGraph getCompleteCayleyGraph( ) const;
bool haveConfluentKBMachine( ) const;
const KBMachine& getConfluentKBMachine( ) const;
Trichotomy isAutomatic( ) const;
bool haveAutomatic( ) const;
const Chars getAutomaticMessage( ) const;
const GroupDFSA& getWordAcceptor( ) const;
const DiffMachine& getDiffMachine( ) const;
bool isOneRelatorWithTorsion( ) const;
bool isOneRelator( ) const;
Word getOneRelator( ) const;
bool haveCyclicDecomposition( ) const;
bool havePrimaryDecomposition( ) const;
const AbelianGroup& getCyclicDecomposition( ) const;
const Chars getCyclicDecompositionMessage( ) const;
Trichotomy  isAbelian( ) const;
Trichotomy  isFree( ) const;
```

```
  Trichotomy  isFinite( ) const;
  const Chars getFiniteExplanation( ) const;
  const Chars getFiniteMessage( ) const;
  bool haveNilpotentQuotInited (int theClass) const;
  bool haveNilpotentGroupInited(            ) const;
  const NilpotentGroup& getNilpotentQuotInited (int theClass) const;
  const NilpotentGroup& getNilpotentGroupInited(            ) const;
  const Chars& getBasisFileName() const
  Trichotomy isFreeNilpotent( ) const;
  Trichotomy isNilpotent( ) const;
  int getNilpotentcyClass( ) const;
  void putActualNilpotencyClass(int c );
  bool haveActualNilpotencyClass( ) const;
  int  getActualNilpotencyClass ( ) const;
  bool isFreeByCyclic( ) const;
  const FreeByCyclic& getFreeByCyclic( ) const;
  bool haveMSC( ) const;
  const MSCGroup& getMSC( ) const;
  bool haveMSCLambda( ) const;
  int  getMSCLambda( ) const;
  Chars getMSCMessage() const;
  bool haveAPOfFree( ) const;
  const AmalgProductOfFreeGroups& getAPOfFree( ) const;
  bool haveHNNofFree( ) const;
  const class HNNExtOfFreeGroup& getHNNofFree() const;
  bool haveSchreierTransvl() const;
  const PermutationRepresentation& getSchreierTransvl() const;
  bool haveWordDecomposer( ) const;
  const class DecomposeInSubgroupOfFPGroup& getWordDecomposer( ) const;
private:
  GIC(const GIC&);
  AssociationsOf<int,NilpotentGroup*>& getNilpotentQuotients() const;
  OID groupOID;
  Chars BCfileName;
};
```

———-

19.11 SessionManager/include/InformationCenter.h

— InformationCenter.h —

```
#include "Property.h"
```

———-

19.11.1 class InformationCenter

— class InformationCenter —

```
class InformationCenter {
public:
  InformationCenter()
  Trichotomy haveProperty ( const PropertyType ptype  ) const;
  void  putProperty       ( const GenericProperty& property, Trichotomy have );
  Chars getDescription    ( const PropertyType ptype  ) const;
  void  setUnknownProperty( const PropertyType ptype  );
  void  setHaveNotProperty( const PropertyType ptype, const Chars& expl );
  void  setKnownProperty  ( const GenericProperty& property );
  void  getKnownProperty  ( GenericProperty& property ) const;
  const GenericProperty* getKnownProperty( const PropertyType ptype ) const;
        GenericProperty* getKnownProperty( const PropertyType ptype );
  void read ( istream& istr, bool extraDetails );
  void write( ostream& ostr, bool extraDetails );
private:
  InformationCenter( const InformationCenter& );
  InformationCenter& operator=( const InformationCenter& );
  PropertiesCollection theProperties;
};

inline
Trichotomy InformationCenter::haveProperty ( const PropertyType ptype ) const

inline
void InformationCenter::putProperty ( const GenericProperty& property,
             Trichotomy       have )

inline
Chars InformationCenter::getDescription ( const PropertyType ptype ) const

inline
void InformationCenter::setUnknownProperty( const PropertyType ptype )

inline
void InformationCenter::setHaveNotProperty( const PropertyType ptype,
             const Chars& expl )

inline
void InformationCenter::setKnownProperty  ( const GenericProperty& property )

inline
void InformationCenter::getKnownProperty  ( GenericProperty& property) const

inline
const GenericProperty* InformationCenter::getKnownProperty(
```

```
   const PropertyType ptype  ) const

inline
GenericProperty* InformationCenter::getKnownProperty(const PropertyType ptype)

inline
void InformationCenter::read ( istream& istr, bool extraDetails )

inline
void InformationCenter::write( ostream& ostr, bool extraDetails )
```

———-

19.12 SessionManager/include/Menu.h

— **Menu.h** —

```
#include "FEData.h"
#include "SMObject.h"
#include "SMFPGroup.h"
#include "SMWord.h"
#include "SMEquation.h"
#include "SMSubgroup.h"
#include "SMSetOfWords.h"
#include "SMVectorOfWords.h"
#include "SMMap.h"
#include "SMMagnusBreakdown.h"
#include "SMPermutation.h"
#include "fastProblems.h"
#include "HomologyProblem.h"
#include "FreeProblems.h"
#include "IsNilpotentProblem.h"
#include "SubgroupProblems.h"
#include "OneRelatorProblems.h"
#include "PackagesSMApps.h"
#include "SMEqSystem.h"
```

———-

19.12.1 struct Ctor

— **struct Ctor** —

```
struct Ctor
{
  virtual void readFrom(istream& istr) const = 0;
```

```
  virtual Ctor* copy( ) const = 0;
protected:
  SMObject* get(int oid) const;
  void newDependent(SMObject* smo, OID oid) const;
};
```

———-

19.12.2 struct CtorAux0

— struct CtorAux0 —

```
struct CtorAux0 : public Ctor
{
  void readFrom(istream& istr) const;
  virtual SMObject* construct(Chars&, istream&) const = 0;
};
```

———-

19.12.3 struct CtorAux1

— struct CtorAux1 —

```
struct CtorAux1 : public Ctor
{
  void readFrom(istream& istr) const;
  virtual SMObject* construct(SMObject*, Chars&, istream&) const = 0;
};
```

———-

19.12.4 struct CtorAux2

— struct CtorAux2 —

```
struct CtorAux2 : public Ctor
{
  CtorAux2(bool transpose = false) : transposeARGS( transpose )
  void readFrom(istream& istr) const;
  virtual SMObject* construct(SMObject*, SMObject*, Chars&, istream&) const
  bool transposeARGS;
};
```

———-

19.12.5 struct CtorAux3

— struct CtorAux3 —

```
struct CtorAux3 : public Ctor
{
  void readFrom(istream& istr) const;
  virtual SMObject* construct(SMObject*, SMObject*, SMObject*,
            Chars&, istream&) const = 0;
};
```

————-

19.12.6 struct CtorAux4

— struct CtorAux4 —

```
struct CtorAux4 : public Ctor
{
  void readFrom(istream& istr) const;
  virtual SMObject* construct(SMObject*, SMObject*, SMObject*, SMObject*,
            Chars&, istream&) const = 0;
};
```

————-

19.12.7 struct CtorArgs0

— struct CtorArgs0 —

```
struct CtorArgs0 : public CtorAux0

template <class T>
struct Ctor0 : public CtorArgs0
{
  Ctor* copy( ) const
private:
  SMObject* construct(Chars&, istream&) const
};

template <class A1> struct CtorArgs1 : public CtorAux1

template <class T, class A1>
struct Ctor1 : public CtorArgs1<A1>
{
  Ctor* copy( ) const
```

```
private:
  SMObject* construct(SMObject* a1, Chars&, istream&) const
};

template <class A1, class A2> struct CtorArgs2 : public CtorAux2
{
  CtorArgs2(bool transpose = false) : CtorAux2( transpose )
};

template <class T, class A1, class A2>
struct Ctor2 : public CtorArgs2<A1,A2>
{
  Ctor2(bool transpose = false) : CtorArgs2<A1,A2>( transpose )
  Ctor* copy( ) const
private:
  SMObject* construct(SMObject* a1, SMObject* a2, Chars&, istream&) const
};

template <class A1, class A2, class A3> struct CtorArgs3 : public CtorAux3

template <class T, class A1, class A2, class A3>
struct Ctor3 : public CtorArgs3<A1,A2,A3>
{
  Ctor* copy( ) const
private:
  SMObject* construct(SMObject* a1, SMObject* a2, SMObject* a3,
          Chars&, istream&) const
};

template <class A1, class A2, class A3, class A4> struct CtorArgs4 :
  public CtorAux4

template <class T, class A1, class A2, class A3, class A4>
struct Ctor4 : public CtorArgs4<A1,A2,A3,A4>
{
  Ctor* copy( ) const
private:
  SMObject* construct(SMObject* a1, SMObject* a2, SMObject* a3, SMObject* a4,
          Chars&, istream&) const
};
```

———-

19.12.8 struct ReadSMFPGroup

— struct ReadSMFPGroup —

```
struct ReadSMFPGroup : public Ctor0<SMFPGroup>
```

```
{
  Ctor* copy( ) const
  SMObject* construct(Chars&, istream&) const;
};
```

———-

19.12.9 struct ReadSMFreeGroup

— struct ReadSMFreeGroup —

```
struct ReadSMFreeGroup : public Ctor0<SMFPGroup>
{
  Ctor* copy( ) const
  SMObject* construct(Chars&, istream&) const;
};
```

———-

19.12.10 struct ReadSMAbelianGroup

— struct ReadSMAbelianGroup —

```
struct ReadSMAbelianGroup : public Ctor0<SMFPGroup>
{
  Ctor* copy( ) const
  SMObject* construct(Chars&, istream&) const;
};
```

———-

19.12.11 struct ReadSMNilpotentGroup

— struct ReadSMNilpotentGroup —

```
struct ReadSMNilpotentGroup : public Ctor0<SMFPGroup>
{
  Ctor* copy( ) const
  SMObject* construct(Chars&, istream&) const;
};
```

———-

19.12.12 struct ReadSMFreeNilpotentGroup

— struct ReadSMFreeNilpotentGroup —

```
struct ReadSMFreeNilpotentGroup : public Ctor0<SMFPGroup>
{
  Ctor* copy( ) const
  SMObject* construct(Chars&, istream&) const;
};
```

———-

19.12.13 struct ReadSMORGroup

— struct ReadSMORGroup —

```
struct ReadSMORGroup : public Ctor0<SMFPGroup>
{
  Ctor* copy( ) const
  SMObject* construct(Chars&, istream&) const;
};
```

———-

19.12.14 struct ReadSMSmallCancGroup

— struct ReadSMSmallCancGroup —

```
struct ReadSMSmallCancGroup : public Ctor0<SMFPGroup>
{
  Ctor* copy( ) const
  SMObject* construct(Chars&, istream&) const;
};
```

———-

19.12.15 struct ReadSMPermutation

— struct ReadSMPermutation —

```
struct ReadSMPermutation : public Ctor0<SMPermutation>
{
  Ctor* copy( ) const
  SMObject* construct(Chars&, istream&) const;
};
```

———-

19.12.16 struct ReadSMWord

— struct ReadSMWord —

```
struct ReadSMWord : public Ctor1<SMWord,SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.17 struct ReadSMEquation

— struct ReadSMEquation —

```
struct ReadSMEquation : public Ctor1<SMEquation,SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.18 struct ReadSMEquation2

— struct ReadSMEquation2 —

```
struct ReadSMEquation2 : public Ctor1<SMEquation2,SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.19 struct ReadSMEqSystem

— struct ReadSMEqSystem —

```
struct ReadSMEqSystem : public Ctor1<SMEqSystem,SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.20 struct ReadSMSubgroup

— **struct ReadSMSubgroup** —

```
struct ReadSMSubgroup : public Ctor1<SMSubgroup,SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.21 struct ReadSMSetOfWords

— **struct ReadSMSetOfWords** —

```
struct ReadSMSetOfWords :
  public Ctor1<SMSetOfWords,SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.22 struct ReadSMVectorOfWords

— **struct ReadSMVectorOfWords** —

```
struct ReadSMVectorOfWords :
  public Ctor1<SMVectorOfWords,SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.23 struct ReadSMMap

— struct ReadSMMap —

```
struct ReadSMMap : public Ctor1<SMMap,SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.24 struct ReadSMMap2

— struct ReadSMMap2 —

```
struct ReadSMMap2 : public Ctor2<SMMap,SMFPGroup,SMFPGroup>, protected FEData
{
  ReadSMMap2(bool transpose = false)
    : Ctor2<SMMap,SMFPGroup,SMFPGroup>( transpose )
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, SMObject* a2, Chars&, istream&) const;
};
```

———-

19.12.25 struct ReadSymmetricGroup

— struct ReadSymmetricGroup —

```
struct ReadSymmetricGroup : public Ctor0<SMFPGroup>, protected FEData
{
  ReadSymmetricGroup() : Ctor0<SMFPGroup>( )
  Ctor* copy( ) const
  SMObject* construct(Chars&, istream&) const;
};
```

———-

19.12.26 struct ReadSMMagnusBreakdown

— struct ReadSMMagnusBreakdown —

```
struct ReadSMMagnusBreakdown :
  public Ctor1<SMMagnusBreakdown,SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.27 struct ReadPowerOfMapItem

— **struct ReadPowerOfMapItem** —

```
struct ReadPowerOfMapItem : public Ctor1<PowerOfMap,SMMap>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.28 struct ReadHomologyItem

— **struct ReadHomologyItem** —

```
struct ReadHomologyItem :
  public Ctor1<HomologyProblem,SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.29 struct ReadHomologyItem1

— **struct ReadHomologyItem1** —

```
struct ReadHomologyItem1 :
  public Ctor1<AbelianIntegralHomologyProblem,SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.30 struct ReadAutEnumItem

— struct ReadAutEnumItem —

```
struct ReadAutEnumItem :
  public Ctor1<AutEnumerator,SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.31 struct ReadFinAutEnumItem

— struct ReadFinAutEnumItem —

```
struct ReadFinAutEnumItem :
  public Ctor1<AutEnumerator,SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.32 struct ReadInitialSegmentItem

— struct ReadInitialSegmentItem —

```
struct ReadInitialSegmentItem :
  public Ctor1<InitialSegmentOfWord,SMWord>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.33 struct ReadPHeight

— struct ReadPHeight —

```
struct ReadPHeight :
  public Ctor1<AbelianPHeightOfEltProblem,SMWord>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.34 struct ReadTerminalSegmentItem

— struct ReadTerminalSegmentItem —

```
struct ReadTerminalSegmentItem :
  public Ctor1<TerminalSegmentOfWord,SMWord>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.35 struct ReadSegmentOfWordItem

— struct ReadSegmentOfWordItem —

```
struct ReadSegmentOfWordItem :
  public Ctor1<SegmentOfWord,SMWord>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.36 struct ReadFreeGetN

— struct ReadFreeGetN —

```
struct ReadFreeGetN_thElementItem :
  public Ctor1<FreeGetN_thElement,SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.37 struct ReadFreeGetNextN

— struct ReadFreeGetNextN —

```
struct ReadFreeGetNextN_thElementItem :
  public Ctor1<FreeGetNextN_thElement,SMWord>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.38 struct ReadMakeNilpotentQuotientItem

— struct ReadMakeNilpotentQuotientItem —

```
struct ReadMakeNilpotentQuotientItem :
    public Ctor1<MakeNilpotentQuotient,SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.39 struct ReadMakeQuotientItem

— struct ReadMakeQuotientItem —

```
struct ReadMakeQuotientItem :
  public Ctor1<MakeQuotient, SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.40 struct ReadIsNilpotentProblemItem

— struct ReadIsNilpotentProblemItem —

```
struct ReadIsNilpotentProblemItem :
  public Ctor1<IsNilpotentProblem, SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.41 struct ReadIsSGNilpotentItem

— struct ReadIsSGNilpotentItem —

```
struct ReadIsSGNilpotentItem :
  public Ctor1<IsSGNilpotent, SMSubgroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.42 struct ReadLCStermProblem

— struct ReadLCStermProblem —

```
struct ReadLCStermProblem :
  public Ctor1<NGLCStermGensProblem, SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.43 struct ReadNthPower

— struct ReadNthPower —

```
struct ReadNthPower : public Ctor1<IsWordNthPower, SMWord>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.44 struct ReadNormalApproximation

— struct ReadNormalApproximation —

```
struct ReadNormalApproximation :
  public Ctor1<NormalApproximationProblem, SMSubgroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.45 struct ReadPackage

— struct ReadPackage —

```
struct ReadPackage :
  public Ctor0<AddPackage>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct( Chars&, istream&) const;
};
```

———-

19.12.46 struct ReadEditPackage

— struct ReadEditPackage —

```
struct ReadEditPackage :
  public Ctor0<EditPackage>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct( Chars&, istream&) const;
};
```

———-

19.12.47 struct ReadGroupPackageID

— struct ReadGroupPackageID —

```
struct ReadGroupPackageID :
  public Ctor1<RunPackage, SMFPGroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.48 struct ReadWordPackageID

— struct ReadWordPackageID —

```
struct ReadWordPackageID :
  public Ctor1<RunPackage, SMWord>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.49 struct ReadSubgroupPackageID

— struct ReadSubgroupPackageID —

```
struct ReadSubgroupPackageID :
  public Ctor1<RunPackage, SMSubgroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.50 struct ReadMapPackageID

— struct ReadMapPackageID —

```
struct ReadMapPackageID :
  public Ctor1<RunPackage, SMMap>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.51 struct ReadHomoPackageID

— struct ReadHomoPackageID —

```
struct ReadHomoPackageID :
  public Ctor1<RunPackage, SMHomomorphism>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, Chars&, istream&) const;
};
```

———-

19.12.52 struct ReadWordWordPackageID

— struct ReadWordWordPackageID —

```
struct ReadWordWordPackageID :
  public Ctor2<RunPackage,SMWord,SMWord>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, SMObject* a2, Chars&, istream&) const;
};
```

———-

19.12.53 struct ReadSubgroupSubgroupPackageID

— struct ReadSubgroupSubgroupPackageID —

```
struct ReadSubgroupSubgroupPackageID :
  public Ctor2<RunPackage,SMSubgroup,SMSubgroup>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, SMObject* a2, Chars&, istream&) const;
};
```

———-

19.12.54 struct ReadSubgroupWordPackageID

— struct ReadSubgroupWordPackageID —

```
struct ReadSubgroupWordPackageID :
  public Ctor2<RunPackage,SMSubgroup,SMWord>, protected FEData
{
  Ctor* copy( ) const
  SMObject* construct(SMObject* a1, SMObject* a2, Chars&, istream&) const;
};
```

———-

19.12.55 class Menu

— **class Menu** —

```
class Menu : public FEData
{
public:
  enum MenuKind { CHECKIN, TOOLS, MAKE, PACKAGES, DATABASE, TESTING };
  void startItemGroup( );
  void startItemGroup(const Expression& condition);
  void startCascade(const Text& text);
  void closeCascade( );
  void addDisabled(const Text& text);
  void separator( );
  void done( );
```

———-

19.12.56 struct Action

— **struct Action** —

```
  struct Action
  { virtual void print(ostream& ostr) const  };
```

———-

19.12.57 struct DefineFPGroup

— **struct DefineFPGroup** —

```
  struct DefineFPGroup : public Action
  {  void print(ostream& ostr) const;  };
```

———-

19.12.58 struct DefineFreeGroup

— struct DefineFreeGroup —

```
struct DefineFreeGroup : public Action
{ void print(ostream& ostr) const; };
```

———-

19.12.59 struct DefineAbelianGroup

— struct DefineAbelianGroup —

```
struct DefineAbelianGroup : public Action
{ void print(ostream& ostr) const;  };
```

———-

19.12.60 struct DefineNilpotentGroup

— struct DefineNilpotentGroup —

```
struct DefineNilpotentGroup : public Action
{ void print(ostream& ostr) const;  };
```

———-

19.12.61 struct DefineFreeNilpotentGroup

— struct DefineFreeNilpotentGroup —

```
struct DefineFreeNilpotentGroup : public Action
{ void print(ostream& ostr) const; };
```

———-

19.12.62 struct DefineSmallCancGroup

— struct DefineSmallCancGroup —

```
struct DefineSmallCancGroup : public Action
{ void print(ostream& ostr) const; };
```

———-

19.12.63 struct DefineORGroup

— struct DefineORGroup —

```
struct DefineORGroup : public Action
{ void print(ostream& ostr) const; };
```

————-

19.12.64 struct DefinePermutation

— struct DefinePermutation —

```
struct DefinePermutation : public Action
{ void print(ostream& ostr) const; };
```

————-

19.12.65 struct DefineWord

— struct DefineWord —

```
struct DefineWord : public Action
{ void print(ostream& ostr) const; };
```

————-

19.12.66 struct DefineEquation

— struct DefineEquation —

```
struct DefineEquation : public Action
{ void print(ostream& ostr) const; };
```

————-

19.12.67 struct DefineEquation2

— struct DefineEquation2 —

```
struct DefineEquation2 : public Action
{ void print(ostream& ostr) const; };
```

————-

19.12.68 struct DefineEqSystem

— struct DefineEqSystem —

```
struct DefineEqSystem : public Action
{ void print(ostream& ostr) const; };
```

———-

19.12.69 struct DefineSubgroup

— struct DefineSubgroup —

```
struct DefineSubgroup : public Action
{ void print(ostream& ostr) const;  };
```

———-

19.12.70 struct DefineSetOfWords

— struct DefineSetOfWords —

```
struct DefineSetOfWords : public Action
{ void print(ostream& ostr) const;  };
```

———-

19.12.71 struct DefineVectorOfWords

— struct DefineVectorOfWords —

```
struct DefineVectorOfWords : public Action
{ void print(ostream& ostr) const;  };
```

———-

19.12.72 struct DefineMap

— struct DefineMap —

```
struct DefineMap : public Action
{ void print(ostream& ostr) const; };
```

———-

19.12.73 struct DefineMap2

— **struct DefineMap2** —

```
struct DefineMap2 : public Action
{ void print(ostream& ostr) const; };
```

———-

19.12.74 struct DefineInverseMap2

— **struct DefineInverseMap2** —

```
struct DefineInverseMap2 : public Action
{ void print(ostream& ostr) const; };
```

———-

19.12.75 struct DefineInt

— **struct DefineInt** —

```
struct DefineInt : public Action
{ DefineInt(const char* p, int l)
   : prompt(p), lower(l), suppliedUpper(false)
 DefineInt(const char* p, int l, int u)
   : prompt(p), lower(l), upper(u), suppliedUpper(true)
 void print(ostream& ostr) const;
private:
 const char* prompt;
 int lower, upper;
 bool suppliedUpper;
};
```

———-

19.12.76 struct DefinePackage

— **struct DefinePackage** —

```
struct DefinePackage : public Action
{
  DefinePackage()
  void print(ostream& ostr) const;
};
```

———-

19.12.77 struct DefineEditPackage

— struct DefineEditPackage —

```
struct DefineEditPackage : public Action
{
  DefineEditPackage()
  void print(ostream& ostr) const;
};
```

————-

19.12.78 struct PackageID

— struct PackageID —

```
struct PackageID : public Action
 {
   PackageID( int id):theID( id )
   void print(ostream& ostr) const;
   private:
   int theID;
 };
```

————-

19.12.79 struct BoundedInteger

— struct BoundedInteger —

```
struct BoundedInteger
{
  BoundedInteger()
    : lowerBound(0), upperBound(0),
      haveLowerBound(false), haveUpperBound(false)
  BoundedInteger(int lower)
    : lowerBound(lower), upperBound(0),
      haveLowerBound(true), haveUpperBound(false)
  BoundedInteger(int lower, int upper)
    : lowerBound(lower), upperBound(upper),
      haveLowerBound(true), haveUpperBound(true)
  int lowerBound, upperBound;
  bool haveLowerBound, haveUpperBound;
};
```

————-

19.12.80 struct DefineInt2

— **struct DefineInt2** —

```
  struct DefineInt2 : public Action
  {
    DefineInt2( const char* p0, const BoundedInteger& bi0,
          const char* p1, const BoundedInteger& bi1 )
    {
      prompt[0] = p0;
      prompt[1] = p1;
      boundInt[0] = bi0;
      boundInt[1] = bi1;
    }
    void print(ostream& ostr) const;
  private:
    const char* prompt[2];
    BoundedInteger boundInt[2];
  };
```

———-

19.12.81 struct DefineSetOfRelators

— **struct DefineSetOfRelators** —

```
  struct DefineSetOfRelators : public Action
  { void print(ostream& ostr) const; };
protected:
  Menu(MenuKind);
  void add(Ctor* ctor, const Text& text, const Action& action);
private:
  enum StateType { INIT, ITEM_GROUP, DONE };
  StateType state;
};
```

———-

19.12.82 class Menu0

— **class Menu0** —

```
class Menu0 : public Menu
{
public:
  Menu0(MenuKind mk) : Menu( mk )
```

```
  void add(const CtorArgs0& ctor,
        const Text& text,
        const Action& action = Action()
        )
};

template <class A1>
class Menu1 : public Menu
{
public:
  Menu1(MenuKind mk) : Menu( mk )
  void add(const CtorArgs1<A1>& ctor,
        const Text& text,
        const Action& action = Action()
        )
};

template <class A1, class A2>
class Menu2 : public Menu
{
public:
  Menu2(MenuKind mk) : Menu( mk )
  void add(const CtorArgs2<A1,A2>& ctor,
        const Text& text,
        const Action& action = Action()
        )
};

template <class A1, class A2, class A3>
class Menu3 : public Menu
{
public:
  Menu3(MenuKind mk) : Menu( mk )
  void add( const CtorArgs3<A1,A2,A3>& ctor,
      const Text& text,
      const Action& action = Action()
      )
};

template <class A1, class A2, class A3, class A4>
class Menu4 : public Menu
{
public:
  Menu4(MenuKind mk) : Menu( mk )
  void add( const CtorArgs4<A1,A2,A3,A4>& ctor,
      const Text& text,
      const Action& action = Action()
      )
};
```

———-

19.13 SessionManager/include/MIC.h

— MIC.h —

```
#include "Trichotomy.h"
#include "Chars.h"
#include "FEData.h"
#include "InformationCenter.h"
```

———-

19.13.1 class HomIsMonoProperty

— class HomIsMonoProperty —

```
class HomIsMonoProperty : public NoDataProperty {
public:
  HomIsMonoProperty( ) : NoDataProperty()
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.13.2 class HomIsEpiProperty

— class HomIsEpiProperty —

```
class HomIsEpiProperty : public NoDataProperty {
public:
  HomIsEpiProperty( ) : NoDataProperty()
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.13.3 class ExtendToHomProperty

— class ExtendToHomProperty —

```
class ExtendToHomProperty : public NoDataProperty {
public:
  ExtendToHomProperty( const Chars& descr ) : NoDataProperty(descr)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.13.4 class MIC

— class MIC —

```
class MIC : public InformationCenter, protected FEData
{
public:
  MIC( OID map_oid );
  ~MIC( );
  void putDoesExtendToHom( bool doesExtend = true, Chars explanation = "" );
  Trichotomy doesExtendToHom( ) const;
  Chars getExtendToHomExplanation( ) const;
  Chars getExtendToHomMessage( ) const;
  void putIsMono(bool ismono);
  void putIsEpi(bool isepi);
  Trichotomy isMono() const;
  Trichotomy isEpi() const;
  OID mapOID;
private:
};
```

———-

19.14 SessionManager/include/ObjectFactory.h

— ObjectFactory.h —

```
class ObjectFactory {
public:
  virtual class SMObject* factory() const = 0;
};
```

———-

19.14.1 class SymmetricGroupFactory

— class SymmetricGroupFactory —

```
class SymmetricGroupFactory : public ObjectFactory {
public:
  SymmetricGroupFactory( int num ) : n(num) {}
  class SMObject* factory() const;
private:
  int n;
};
```

———-

19.15 SessionManager/include/ObjectSmith.h

— ObjectSmith.h —

```
#include "Menu.h"
```

———-

19.15.1 class ObjectSmith

— class ObjectSmith —

```
class ObjectSmith : protected Menu
{
public:
  static void outputPackagesMenuDefns(ostream& ostr);
private:
  ObjectSmith( );
  friend class SessionManager;
  friend class Menu;
  static void outputCheckinMenuDefns(ostream& ostr);
  static void outputToolsMenuDefns(ostream& ostr);
  static void outputMakeMenuDefns(ostream& ostr);
  static void outputTestingMenuDefns(ostream& ostr);
```

```
  static void readMessage(istream& istr);
  static int registerCallBack(Ctor* ctor);
  static int theCallBacksLen;
  static Ctor** theCallBacks;
  static Ctor** freeCallBack;
  static const int jumpSize = 64;
};
```

———-

19.16 SessionManager/include/OID.h

— **OID.h** —

```
#include <iostream.h>
#include "IPC.h"
```

———-

19.16.1 class OID

— **class OID** —

```
class OID
{
public:
  bool operator == ( const OID& oid ) const
  bool operator != ( const OID& oid ) const
  inline friend ostream& operator << ( ostream& ostr, const OID& oid )
  int hash( ) const
  int unwrap( ) const
  friend ostream& operator < ( ostream& ostr, const OID& oid )
  friend istream& operator > ( istream& istr, OID& oid )
private:
  friend class Ctor;
  friend class SessionManager;
  friend class TheObjects;
  friend class SMFPGroup;
  friend class RunPackage;
  friend class DatabaseManager;
  OID( int i ) : theOID( i )
  int theOID;
public:
  OID( ) : theOID( 0 )
};
```

———-

19.17 SessionManager/include/OutMessages.h

— **OutMessages.h** —

```
#include <stdio.h>
#include <iostream.h>
#include "Chars.h"
#include "List.h"
#include "ARC.h"
#include "ARCSlotID.h"
#include "FEData.h"
#include "VectorPtr.h"
```

———-

19.17.1 class OutMessage

— **class OutMessage** —

```
class OutMessage
{
public:
  void send( ) const;
protected:
  OutMessage( )
  virtual void print(ostream& ostr) const = 0;
private:
  OutMessage( const OutMessage& );
};
```

———-

19.17.2 class LogMessage

— **class LogMessage** —

```
class LogMessage : public OutMessage, protected FEData, public ostrstream
{
public:
  LogMessage(int message_level = 1);
  LogMessage(OID o, int message_level = 1);
  LogMessage(OID o1, OID o2, int message_level = 1);
  LogMessage(OID o1, OID o2, OID o3, int message_level = 1);
  LogMessage( const char* msg, int message_level = 1);
  LogMessage(OID o, const char* msg, int message_level = 1);
protected:
```

```
  void print(ostream& ostr) const;
private:
  int num_addressees;
  VectorPtrOf<OID> oids;
  int level;
};
```

———-

19.17.3 class ParseErrorMessage

— class ParseErrorMessage —

```
class ParseErrorMessage : public OutMessage
{
public:
  ParseErrorMessage(const Chars name, const Chars diagnosis);
protected:
  void print(ostream& ostr) const;
private:
  const Chars theName;
  const Chars theDiagnosis;
};
```

———-

19.17.4 class PackageInfoMessage

— class PackageInfoMessage —

```
class PackageInfoMessage : public OutMessage
{
public:
  PackageInfoMessage(const Chars& name, const Chars& command,
                     const Chars& ch_type, const Chars& obj);
protected:
  void print(ostream& ostr) const;
private:
  const Chars theChType;
  const Chars theName;
  const Chars theObject;
  const Chars theCommand;
};
```

———-

19.17.5 class ParseParamErrorMessage

— class ParseParamErrorMessage —

```
class ParseParamErrorMessage : public OutMessage
{
public:
  ParseParamErrorMessage(OID o,const Chars& name,
       const Chars& type,
       const Chars& msg);
protected:
  void print(ostream& ostr) const;
private:
  OID theOID;
  const Chars theType;
  const Chars theName;
  const Chars theMsg;
};
```

———-

19.17.6 class ParseParamOk

— class ParseParamOk —

```
class ParseParamOk : public ParseParamErrorMessage
{
public:
  ParseParamOk(OID o):
    ParseParamErrorMessage(o,"Parameters","","none")
};
```

———-

19.17.7 class CheckinMessage

— class CheckinMessage —

```
class CheckinMessage : public OutMessage
{
public:
  CheckinMessage(const class SMObject& smo,
     const Chars name,
     const ListOf<OID> dependencies,
     bool bPrintDefinition = true
     );
```

```
protected:
  void print(ostream& ostr) const;
private:
  const class SMObject& theSMO;
  const Chars theName;
  const ListOf<OID> theDependencies;
  bool BPrintDefinition;
};
```

———-

19.17.8 class FEDataUpdate

— class FEDataUpdate —

```
class FEDataUpdate : public OutMessage, protected FEData
{
public:
  FEDataUpdate( const Key& key, const Expression& datum )
    : theKey( key ), theDatum( datum )
protected:
  void print(ostream& ostr) const;
private:
  const Key theKey;
  const Expression theDatum;
};
```

———-

19.17.9 class StateTransition

— class StateTransition —

```
class StateTransition : public OutMessage
{
public:
  enum Kind { START, SUSPEND, RESUME, TERMINATE, STALLED };
  StateTransition(Kind k, OID o) : kind( k ), oid( o )
protected:
  void print(ostream& ostr) const;
private:
  const Kind kind;
  const OID oid;
};
```

———-

19.17.10 class ARCUpdate

— class ARCUpdate —

```
class ARCUpdate : public OutMessage
{
public:
  ARCUpdate(OID o, ARCSlotID a, ARC v) : oid( o ), asi( a ), value( v )
protected:
  void print(ostream& ostr) const;
private:
  OID oid;
  ARCSlotID asi;
  ARC value;
};
```

———-

19.17.11 class Warning

— class Warning —

```
class Warning : public OutMessage, protected FEData, public ostrstream
{
protected:
  void print(ostream& ostr) const;
};
```

———-

19.17.12 class Message

— class Message —

```
class Message : public OutMessage, protected FEData, public ostrstream
{
public:
  Message( Chars title = "Information" ) : theTitle( title )
protected:
  void print(ostream& ostr) const;
private:
  Chars theTitle;
};
```

———-

19.17.13 class InvokingMessage

— class InvokingMessage —

```
class InvokingMessage : public OutMessage, protected FEData, public ostrstream
{
public:
  InvokingMessage( )
protected:
  void print(ostream& ostr) const;
};
```

———-

19.18 SessionManager/include/Property.h

— Property.h —

```
#include "QuickAssociations.h"
#include "Chars.h"
#include "APofFreeGroups.h"
#include "Automorphism.h"
#include "HNNExtOfFreeGroup.h"
#include "NilpotentGroup.h"
#include "CosetEnumerator.h"
#include "DFSA.h"
#include "DiffMachine.h"
#include "KBMachine.h"
#include "DecomposeInSubgroup.h"
#include "AbelianSGPresentation.h"
#include "SGOfNilpotentGroup.h"
#include "PolyWord.h"
```

———-

19.18.1 class PropertyType

— class PropertyType —

```
class PropertyType {
public:
  PropertyType( int t ) : type(t)
  int unwrap() const
  int operator++()
  friend ostream& operator<<(ostream& ostr, const PropertyType& p)
  friend istream& operator>>(istream& istr, PropertyType& p)
```

```
private:
  int type;
};

inline bool operator==( const PropertyType a, const PropertyType b )
inline bool operator!=( const PropertyType a, const PropertyType b )
```

———-

19.18.2 class PropertiesManager

— class PropertiesManager —

```
class PropertiesManager {
public:
  static PropertiesManager* instance();
  PropertyType typeOf( const Chars& propertyName ) const;
  Chars nameOf( const PropertyType& ptype ) const;
  const GenericProperty* property( const PropertyType& ptype ) const;
  bool isRegisteredProperty( const PropertyType ptype ) const;
  void registerProperty( const GenericProperty& property );
  void unregisterProperty( const GenericProperty& property );
  void unregisterProperty( const PropertyType ptype );
  GenericProperty* readProperty(istream& istr, bool extraDetails);
  GenericProperty* readProperty(istream& istr, const PropertyType ptype,
        bool extraDetails);
private:
  PropertiesManager()
  PropertiesManager( const PropertiesManager& );
  PropertiesManager& operator=( const PropertiesManager& );
  class SingletonKiller {
  public:
    ~SingletonKiller()
  };
  friend class SingletonKiller;
  static PropertiesManager *theInstance;
  static SingletonKiller    theKiller;
  QuickAssociationsOf<PropertyType, GenericProperty*> theRegisteredProperties;
  QuickAssociationsOf<Chars, PropertyType>            theRegisteredTypes;
};

void registerProperties();
extern const PropertyType unregisteredProperty;
```

———-

19.18.3 class PropertiesCollection

— class PropertiesCollection —

```
class PropertiesCollection
{
public:
  PropertiesCollection( )
  Trichotomy haveProperty  ( const PropertyType ptype  ) const;
  void  putProperty       ( const GenericProperty& property, Trichotomy have );
  Chars getDescription    ( const PropertyType ptype  ) const;
  void  setUnknownProperty( const PropertyType ptype  );
  void  setHaveNotProperty( const PropertyType ptype, const Chars& expl );
  void  setKnownProperty  ( const GenericProperty& property );
  void  getKnownProperty  ( GenericProperty& property ) const;
  const GenericProperty* getKnownProperty( const PropertyType ptype ) const;
        GenericProperty* getKnownProperty( const PropertyType ptype );
  void read ( istream& istr, bool extraDetails );
  void write( ostream& ostr, bool extraDetails );
private:
  QuickAssociationsOf<PropertyType, GenericProperty*> knownProperties;
  QuickAssociationsOf<PropertyType, Chars>            haveNotProperties;
};
```

———-

19.18.4 class PropertiesParser

— class PropertiesParser —

```
class PropertiesParser
{
public:
  PropertiesParser(istream &i) : istr(i)
  Chars parsePropertiesCollection(
    QuickAssociationsOf<PropertyType, GenericProperty*>& known,
    QuickAssociationsOf<PropertyType, Chars>&            unknown,
    bool extraDetails );
  Chars parseRawWord(Word& w);
  bool skipWhiteSpaces();
  bool skipWhiteSpacesAndGivenChar(char c);
private:
  Chars parseKnownProperties(
    QuickAssociationsOf<PropertyType,GenericProperty*>& known,
    bool extraDetails );
  Chars parseUnknownProperties(
    QuickAssociationsOf<PropertyType,Chars>& unknown );
  char peekCh()
```

```
  char getCh()
  istream& istr;
};
```

———-

19.18.5 class GenericProperty

— class GenericProperty —

```
class GenericProperty {
public:
  GenericProperty( const Chars& descr ) : theDescription( descr )
  virtual ~GenericProperty()
  virtual PropertyType actualType  () const = 0;
  virtual Chars        propertyName() const = 0;
  virtual void         putProperty( GenericProperty* property ) = 0;
  Chars getDescription() const
  virtual bool isStorable() const
  friend  istream& operator>>( istream& istr, GenericProperty& p )
  friend ostream& operator<<( ostream& ostr, const GenericProperty& p )
  virtual void read (istream& istr, bool extraDetails = false)  = 0;
  virtual void write(ostream& ostr, bool extraDetails,
         const Chars& extrasDetailsFileName ) const = 0;
protected:
  friend class PropertiesCollection;
  friend class PropertiesManager;
  virtual GenericProperty* clone() const = 0;
protected:
  virtual void readFrom(istream& istr);
  virtual void writeTo (ostream& ostr) const;
  static PropertyType unique();
private:
  static PropertyType theTypeCounter;
  Chars theDescription;
};
```

———-

19.18.6 class GenericUnstorableProperty

— class GenericUnstorableProperty —

```
class GenericUnstorableProperty : public GenericProperty {
public:
  GenericUnstorableProperty( const Chars& descr ) : GenericProperty(descr)
  bool isStorable() const
```

```
protected:
  void read (istream& istr, bool extraDetails = false)   }
  void write(ostream& ostr, bool extraDetails,
       const Chars& extrasDetailsFileName) const
};
```

———-

19.18.7 class GenericSimpleProperty

— class GenericSimpleProperty —

```
class GenericSimpleProperty : public GenericProperty {
public:
  GenericSimpleProperty( const Chars& descr ) : GenericProperty(descr)
protected:
  void read (istream& istr, bool extraDetails = false);
  void write(ostream& ostr, bool extraDetails,
       const Chars& extrasDetailsFileName) const;
};
```

———-

19.18.8 class GenericComplexProperty

— class GenericComplexProperty —

```
class GenericComplexProperty : public GenericProperty {
public:
  GenericComplexProperty( const Chars& descr ) : GenericProperty(descr)
protected:
  void read (istream& istr, bool extraDetails = false);
  void write(ostream& ostr, bool extraDetails,
       const Chars& extrasDetailsFileName) const;
private:
  static void completeHeader( char *header, const char *title,
            const char *name, int start, int length );
  static const int header_size;
};

template <class T>
class UnstorableProperty : public GenericUnstorableProperty {
public:
  UnstorableProperty(const T& data, const Chars descr)
    : GenericUnstorableProperty(descr), theData(data)
  const T& value() const
        T& ref  ()
```

```
  void putProperty( GenericProperty* property )
  void readFrom(istream& istr)
  void writeTo (ostream& ostr) const
protected:
  T theData;
};

template <class T>
class SimpleProperty : public GenericSimpleProperty {
public:
  SimpleProperty(const T& data, const Chars descr)
    : GenericSimpleProperty(descr), theData(data)
  const T& value() const
        T& ref  ()
  void putProperty( GenericProperty* property )
  void readFrom(istream& istr)
  void writeTo (ostream& ostr) const
protected:
  T theData;
};

template <class T>
class ComplexProperty : public GenericComplexProperty {
public:
  ComplexProperty(const T& data, const Chars descr)
    : GenericComplexProperty(descr), theData(data)
  const T& value() const
        T& ref  ()
  void putProperty( GenericProperty* property )
protected:
  T theData;
};
```

———-

19.19 SessionManager/include/RandomDefinitionsGenerator.h

— **RandomDefinitionsGenerator.h** —

```
#include "RandomNumbers.h"
#include "FPGroup.h"
#include "Chars.h"
#include "ViewContents.h"
#include <iostream.h>
```

———-

19.19.1 class RandomDefinitionGenerate

— class RandomDefinitionGenerate —

```
class RandomDefinitionGenerate
{
public:
  RandomDefinitionGenerate() : gw("Parameters")
  virtual void readParameters(istream& istr);
  virtual void requireParameters() = 0;
  virtual void printDefinition() = 0;
protected:
  GroupWindow gw;
};
```

———-

19.19.2 class RandomWordGenerate

— class RandomWordGenerate —

```
class RandomWordGenerate : public RandomDefinitionGenerate
{
public:
  RandomWordGenerate(istream& is, const FPGroup& p, Chars id);
  void requireParameters();
  void printDefinition();
private:
  RandomWordGenerate(const RandomWordGenerate &);
  Chars generateWord( );
  istream& istr;
  FPGroup parent;
  Chars theID;
  UniformRandom ur;
  int averagelength;
};
```

———-

19.19.3 class RandomCollectionOfWordsGenerate

— class RandomCollectionOfWordsGenerate —

```
class RandomCollectionOfWordsGenerate : public RandomDefinitionGenerate
{
public:
```

```
  RandomCollectionOfWordsGenerate(istream& is, const FPGroup& p,
                                  Chars id, bool isS = false);
  void requireParameters();
  void printDefinition();
private:
  RandomCollectionOfWordsGenerate(const RandomCollectionOfWordsGenerate &);
  Chars generateCollection( );
  istream& istr;
  FPGroup parent;
  Chars theID;
  UniformRandom ur;
  int averagelength;
  int numberOfWords;
  bool isSet;
};
```

———-

19.19.4 class RandomMapGenerate

— class RandomMapGenerate —

```
class RandomMapGenerate : public RandomDefinitionGenerate
{
public:
  RandomMapGenerate(istream& is, const FPGroup& p, Chars id);
  void requireParameters();
  void printDefinition();
private:
  RandomMapGenerate(const RandomMapGenerate &);
  Chars generateMap( );
  istream& istr;
  FPGroup parent;
  Chars theID;
  UniformRandom ur;
  int averagelength;
};
```

———-

19.19.5 class RandomGroupGenerate

— class RandomGroupGenerate —

```
class RandomGroupGenerate : public RandomDefinitionGenerate
{
public:
```

```
  RandomGroupGenerate(istream& is, Chars id, bool iF = false,
                      bool iN = false, bool iOR = false);
  void requireParameters();
  void printDefinition();
private:
  RandomGroupGenerate(const RandomGroupGenerate &);
  Chars generateGroup( );
  istream& istr;
  Chars theID;
  int maxGens;
  int maxRels;
  int averageRels;
  int nilpClass;
  bool isFree;
  bool isNilpotent;
  bool isOR;
};
```

———-

19.19.6 class RandomSCGroupGenerate

— class RandomSCGroupGenerate —

```
class RandomSCGroupGenerate : public RandomDefinitionGenerate
{
public:
  RandomSCGroupGenerate(istream& is, Chars id);
  void requireParameters();
  void printDefinition();
private:
  RandomSCGroupGenerate(const RandomSCGroupGenerate &);
  Chars generateSCGroup( );
  istream& istr;
  Chars theID;
  int maxGens;
  int maxRels;
  int averageRels;
};
```

———-

19.19.7 class RandomDefinitionsGenerator

— class RandomDefinitionsGenerator —

```
class RandomDefinitionsGenerator
{
 private:
  RandomDefinitionsGenerator( );
  friend class SessionManager;
  static void readMessage(istream& istr);
  static Chars generateFPGroupPresentation();
  static RandomDefinitionGenerate* defGenerator;
};
```

———-

19.20 SessionManager/include/ResourceManager.h

— **ResourceManager.h** —

```
#include "ARC.h"
#include "ARCSlotID.h"
#include "OID.h"
```

———-

19.20.1 class ResourceManager

— **class ResourceManager** —

```
class ResourceManager
{
public:
  ResourceManager(OID oid);
  ~ResourceManager( );
  OID getOID( ) const;
  ARC freeARCs( ) const;
  bool workingFor(OID oid) const;
  bool isNeeded() const;
  void freeze( );
  void liquefy( );
  void allocate(ResourceManager& recipient, ARC arcs);
  void acceptAllocation(OID benefactor,
                ARCSlotID asi,
                ARC arcs,
                bool overrides = false
                );
  void usedARCs(ARC arcs);
  void usedOneARC( );
  void usedMemory(int kilobytes)
```

```
private:
  bool isLiquid(OID oid) const;
  ResourceManager(const ResourceManager&);
  ResourceManager& operator = (const ResourceManager&);
```

———-

19.20.2 struct Resources

— struct Resources —

```
  struct Resources
  {
   Resources(OID oid, ARCSlotID asi, ARC arcs, Resources* n) :
     benefactor(oid),
    theARCs(arcs),
    theARCSlotID(asi),
    memory(-1),
    next(n)
   ~Resources( )
   OID benefactor;
   ARCSlotID theARCSlotID;
   ARC theARCs;
   int memory;
   Resources* next;
  };
  Resources* toUse;
  Resources* lastDrawnFrom;
  OID theOID;
};
```

———-

19.21 SessionManager/include/SessionManager.h

— SessionManager.h —

```
#include <iostream.h>
#include "ObjectSmith.h"
#include "OutMessages.h"
```

———-

19.21.1 class SessionManager

— class SessionManager —

```
class SessionManager
{
public:
  static OID getObjectSmithOid( )
private:
  friend int main(int argc, char* argv[]);
  static void start( );
  static void initializeFrontEnd( );
  static bool areMessages( );
  static void readMessages( );
  static void readMessage(istream& istr);
  static void takeControl( );
  static const int sessionManagerOid = -1;
  static const int objectSmithOid = -2;
  static const int databaseManagerOid = -3;
  static const int randomDefnGeneratorOid = -4;
  enum MessageTag { QUIT, DELETE, SET_NAME };
  static const long magicCookie;
  static bool quitRequested;
};
```

———-

19.22 SessionManager/include/SMEnumerator.h

— SMEnumerator.h —

```
#include "AlgebraicObject.h"
#include "SMFPGroup.h"
#include "Word.h"
#include "Associations.h"
#include "Subgroup.h"

template <class T> class EnumeratorProblem;
```

———-

19.22.1 class NoType

— class NoType —

```
class NoType {
 public:
  static Type type( )
 private:
  static const Type theNoType;
```

```
};
```

19.22.2 class SMListData

— class SMListData —

```
class SMListData
{
public:
  SMListData( );
  SMListData( const SMListData& sd );
  void doHardCopy( const SMListData& );
  ~SMListData()
  SMListData& operator << (const class WriteEnumeratorElement& e);
  SMListData& operator << (const Chars& c);
  void append( SMListData& sd );
  void setWriteMode();
  void setReadMode();
  void closeCurrentMode();
  Chars getElementOf( int i );
  int numberOfElements() const
  Chars getDataFileName() const
  int getMode() const
  friend ostream& operator < ( ostream& ostr, const SMListData& s )
  friend istream& operator > ( istream& istr, SMListData& s )
private:
  void openFiles( int );
  int getBegPosOf( int i);
  int getLengthOf( int i);
  fstream refFile;
  Chars dataFileName;
  Chars refFileName;
  int number_of_elements;
  int mode;
};
```

19.22.3 class EnumeratorARCer

— class EnumeratorARCer —

```
class EnumeratorARCer : public ARCer
{
public:
```

```
  EnumeratorARCer( ComputationManager& boss, SMListData& d )
    : ARCer( boss ), theData( d )
  void submitSignal( );
  void runComputation( );
 protected:
  virtual void enumerate() = 0;
  bool submitSignalRecieved()const;
  SMListData& theData;
 private:
};

template <class T> class EnumeratorProblem;

template <class T>  class EnProbType {
public:
  EnProbType()
private:
  friend class EnumeratorProblem<T>;
  char s[100];
};

template <class T> class EnumeratorProblem  : public Supervisor
{
public:
  EnumeratorProblem( AlgebraicObject& b ):
    Supervisor(true),
    theParent(b)
  ~EnumeratorProblem()
  AlgebraicObject& getParentObject() const
  const IconID iconID( ) const
  const char* typeID( ) const
  static const char* type( )
  SMListData& getData()
protected:
  SMListData theData;
  Chars theDataFile;
  static EnProbType<T> theType;
  AlgebraicObject& theParent;
};
```

———-

19.23 SessionManager/include/SMEqSystem.h

— SMEqSystem.h —

```
#include "AlgebraicObject.h"
#include "SMFPGroup.h"
```

```
#include "Word.h"
```

———-

19.23.1 class SEIC

— class SEIC —

```
class SEIC : protected FEData
{
public:
  SEIC(OID eqs_oid);
  ~SEIC( );
  void putIsSolvedInAbelianization()
  void putHaveSolutionsInAbelianization( bool sol )
  void putIsSolved()
  void putHaveSolutions( bool sol )
  bool isSolvedInAbelianization() const
  bool haveSolutionsInAbelianization( ) const
  bool isSolved() const
  bool haveSolutions( ) const
private:
  SEIC(const SEIC&);
  OID eqSystemOID;
  bool solvedInAbelian;
  bool haveSolInAbelian;
  bool solved;
  bool haveSol;
};
```

———-

19.23.2 class SMEqSystem

— class SMEqSystem —

```
class SMEqSystem : public AlgebraicObject
{
public:
  SMEqSystem( SMFPGroup& G, FreeGroup F, const VectorOf<Word> v, int nvar,
        const Chars heritage);
  SMEqSystem( SMFPGroup& G )
    : AlgebraicObject(""),
      theGroup(G),
      seic( oid() )
  VectorOf<Word> getSystem( ) const
  SMFPGroup& getParent( ) const
```

```
  const FreeGroup& getFreeGroup( ) const
  int numberOfVariables( ) const
  static const char* type( )
  const char* typeID( ) const
  const IconID iconID( ) const
  void viewStructure(ostream& ostr) const;
  void printProperties(ostream& ostr) const;
  void printDefinition(ostream& ostr) const;
public:
  SEIC seic;
protected:
  void readMessage(istream&)
private:
  SMFPGroup& theGroup;
  FreeGroup theFreeGroup;
  VectorOf<Word> theSystem;
  int numOfVar;
};
```

———-

19.24 SessionManager/include/SMEquation.h

— **SMEquation.h** —

```
#include "AlgebraicObject.h"
#include "SMFPGroup.h"
#include "Word.h"
#include "QEqnSolutions.h"
#include "BaseProperties.h"
#include "InformationCenter.h"
```

———-

19.24.1 class AllBasicSolutionsProperty

— **class AllBasicSolutionsProperty** —

```
class AllBasicSolutionsProperty : public NoDataProperty {
public:
  AllBasicSolutionsProperty() : NoDataProperty()
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
```

```
private:
  static const PropertyType theTypeID;
};
```

———-

19.24.2 class AllRegStabGeneratorsProperty

— class AllRegStabGeneratorsProperty —

```
class AllRegStabGeneratorsProperty : public NoDataProperty {
public:
  AllRegStabGeneratorsProperty() : NoDataProperty()
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.24.3 class BasicSolutionsProperty

— class BasicSolutionsProperty —

```
class BasicSolutionsProperty : public ListOfEndomorphismProperty {
public:
  BasicSolutionsProperty(const ListOf<Endomorphism>& data,
       const Chars& descr = "" )
    : ListOfEndomorphismProperty(data, descr)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.24.4 class RegStabGeneratorsProperty

— class RegStabGeneratorsProperty —

```
class RegStabGeneratorsProperty : public ListOfAutomorphismProperty {
public:
  RegStabGeneratorsProperty(const ListOf<Automorphism>& data,
          const Chars& descr = "" )
    : ListOfAutomorphismProperty(data, descr)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.24.5 class EIC

— class EIC —

```
class EIC : public InformationCenter, protected FEData
{
public:
  EIC(OID group_oid);
  ~EIC( );
  void putHaveAllBasicSolutions( bool found );
  void putHaveAllRegStabGenerators( bool found );
  void addBasicSolutions( const ListOf<Endomorphism>& basicSolutions );
  void addRegStabGenerators( const ListOf<Automorphism>& regStabGens );
  void putIsSolvedInAbelianization()
  void putHaveSolutionsInAbelianization( bool sol )
  bool isSolved() const
  bool haveAllBasicSolutions( ) const;
  bool haveAllRegStabGenerators( ) const;
  Trichotomy haveBasicSolutions() const;
  bool isSolvedInAbelianization() const
  bool haveSolutionsInAbelianization( ) const
  ListOf<Endomorphism> getBasicSolutions( ) const;
  Trichotomy haveRegStabGenerators() const;
  ListOf<Automorphism> getRegStabGenerators( ) const;
private:
  EIC(const EIC&);
  OID equationOID;
  ListOf<Endomorphism> theBasicSolutions;
```

```
  ListOf<Automorphism> theRegStabGenerators;
  bool foundAllBasicSolutions;
  bool foundAllRegStabGenerators;
  bool solvedInAbelian;
  bool haveSolInAbelian;
};
```

———-

19.24.6 class SMEquation

— class SMEquation —

```
class SMEquation : public AlgebraicObject
{
public:
  SMEquation( SMFPGroup& G, FreeGroup F, const Word w, int nvar,
        const Chars heritage);
  SMEquation(SMFPGroup& G) : AlgebraicObject(""), theGroup(G), eic( oid() )
  Word getWord( ) const
  SMFPGroup& getParent( ) const
  const FreeGroup& getFreeGroup( ) const
  int numberOfVariables( ) const
        InformationCenter* infoCenter()
  const InformationCenter* infoCenter() const
  static const char* type( )
  const char* typeID( ) const
  const IconID iconID( ) const
  void viewStructure(ostream& ostr) const;
  void printProperties(ostream& ostr) const;
  void printDefinition(ostream& ostr) const;
public:
  EIC eic;
protected:
  void readMessage(istream&)
private:
  SMFPGroup& theGroup;
  FreeGroup theFreeGroup;
  Word theWord;
  int numOfVar;
};
```

———-

19.24.7 class SMEquation2

— class SMEquation2 —

```
class SMEquation2 : public AlgebraicObject
{
public:
  SMEquation2( SMFPGroup& G, FreeGroup F, const Word w, int nvar,
         const Chars heritage);
  SMEquation2(SMFPGroup& G) : AlgebraicObject(""), theGroup(G), eic( oid() )
  Word getWord( ) const
  SMFPGroup& getParent( ) const
  const FreeGroup& getFreeGroup( ) const
  int numberOfVariables( ) const
  static const char* type( )
  const char* typeID( ) const
  const IconID iconID( ) const
  void viewStructure(ostream& ostr) const;
  void printProperties(ostream& ostr) const;
  void printDefinition(ostream& ostr) const;
public:
  EIC eic;
protected:
  void readMessage(istream&)
private:
  SMFPGroup& theGroup;
  FreeGroup theFreeGroup;
  Word theWord;
  int numOfVar;
};
```

———-

19.25 SessionManager/include/SMFPGroup.h

— SMFPGroup.h —

```
#include "AlgebraicObject.h"
#include "FPGroup.h"
#include "GIC.h"
#include "GCM.h"
#include "Supervisor.h"
```

———-

19.25.1 class SMFPGroup

— class SMFPGroup —

```
class SMFPGroup : public AlgebraicObject
```

```
{
public:
  enum Checkin_Type {
   FP, FREE, ABELIAN, NILPOTENT, SMALL_CANCELLATION, ONE_RELATOR,
   AP_FREE, AP_FREE_CYCLIC, FREE_NILPOTENT, HNN_FREE, ONE_RELATOR_WITH_TORSION
  };
  SMFPGroup(const FPGroup& G, const Chars heritage, Checkin_Type t = FP,
      Chars def = Chars());
  SMFPGroup( ) : AlgebraicObject( "" ), gic( oid() ), theGCM( 0 )
  FPGroup getFPGroup( ) const
  FreeGroup getFreePreimage( ) const
  GCM& gcm() const
  Checkin_Type getCheckinType( ) const
  Chars getCheckinTypeStr( ) const;
        InformationCenter* infoCenter()
  const InformationCenter* infoCenter() const
  static const char* type( )
  const IconID iconID( ) const
  const char* typeID( ) const
  void viewStructure(ostream& ostr) const;
  void printProperties(ostream& ostr) const;
  void printDefinition(ostream& ostr) const;
  static void printGlobalMessageTemplates(ostream& ostr)
protected:
  void readMessage(istream&);
public:
  GIC gic;
private:
  enum MessageTag { GENS_MAP_REQUEST };
  GCM* theGCM;
  const FPGroup theGroup;
  const FreeGroup theFreePreimage;
  Checkin_Type  checkin_type;
  Chars definition;
};
```

———-

19.26 SessionManager/include/SMHomomorphism.h

— **SMHomomorphism.h** —

```
#include "SMMap.h"
```

———-

19.26.1 class SMHomomorphism

— class SMHomomorphism —

```
class SMHomomorphism : public SMMap
{
public:
  SMHomomorphism( const class SMMap& map, const Chars& heritage ) :
    SMMap( map.getDomain(), map.getRange(), map.getMap(), heritage )
  SMHomomorphism( SMFPGroup& G, Map m, const Chars heritage) :
    SMMap( G, m, heritage )
  SMHomomorphism( SMFPGroup& G, SMFPGroup& H, Map m, const Chars heritage) :
    SMMap( G, H, m, heritage )
  static const char* type( )
  const char* typeID( ) const
  const IconID iconID( ) const
protected:
  void readMessage(istream&)
private:
};
```

———-

19.26.2 class SMHomomorphism2

— class SMHomomorphism2 —

```
class SMHomomorphism2 : public SMHomomorphism
{
public:
  SMHomomorphism2( const class SMMap& map, const Chars& heritage ) :
    SMHomomorphism( map, heritage )
  SMHomomorphism2( SMFPGroup& G, SMFPGroup& H, Map m, const Chars heritage) :
    SMHomomorphism( G, H, m, heritage )
  static const char* type( )
  const char* typeID( ) const
protected:
private:
};
```

———-

19.27 SessionManager/include/SMList.h

— SMList.h —

```
#include "AlgebraicObject.h"
#include "SMFPGroup.h"
#include "Word.h"
#include "Associations.h"
#include "SMEnumerator.h"

template <class T> class SMList;

template <class T> class SMList;

template <class T>  class SMListType {
public:
  SMListType()
private:
  friend class SMList<T>;
  char s[100];
};
```

———-

19.27.1 class LIC

— class LIC —

```
class LIC : protected FEData
{
public:
  LIC( OID, int );
  ~LIC( );
  Trichotomy getTrivialStatus( int, GIC::AlgorithmID&, Chars& ) const;
  Trichotomy getTrivialStatus( int i ) const
  void setTrivialStatus( int i, Trichotomy is_trivial,
       GIC::AlgorithmID al = GIC::NONE  );
  bool IsTrivialChecked() const;
  bool hasTrivialStatusBuffer() const;
  void putIsTrivialFiles( const Chars& is_trivial,
      const Chars& is_not_trivial )
  Chars getListOfTrivial() const
  Chars getListOfNonTrivial() const
  Trichotomy getAbelianStatus( int, GIC::AlgorithmID&, Chars& ) const;
  Trichotomy getAbelianStatus( int i ) const
  void setAbelianStatus( int i, Trichotomy is_abelian,
       GIC::AlgorithmID al = GIC::NONE  );
  bool IsAbelianChecked() const;
  bool hasAbelianStatusBuffer( ) const;
  void putAbelianFiles( const Chars& is_abelian,
      const Chars& is_not_abelian )
  Chars getListOfAbelian() const
```

```
  Chars getListOfNonAbelian() const
  Trichotomy getIsCentralStatus( int, GIC::AlgorithmID&, Chars& ) const;
  Trichotomy getIsCentralStatus( int i ) const
  void setIsCentralStatus( int i, Trichotomy is_abelian,
       GIC::AlgorithmID al = GIC::NONE  );
  bool IsCentralChecked() const;
  bool hasIsCentralStatusBuffer( ) const;
  void putIsCentralFiles( const Chars& is_central,
        const Chars& is_not_central )
  Chars getListOfCentral() const
  Chars getListOfNonCentral() const
  int numberOfElements() const
private:
```

———-

19.27.2 struct status

— **struct status** —

```
  struct status_type {
    status_type( int );
    ~status_type();
    Trichotomy getStatus( int, GIC::AlgorithmID& ) const;
    void setStatus( int i, Trichotomy is_abelian,
        GIC::AlgorithmID al = GIC::NONE  );
    bool isChecked() const;
  private:
    char* status_buffer;
    int status_file;
    int size;
  };
  Chars explanation( GIC::AlgorithmID ) const;
  OID listOID;
  int number_of_elements;
  status_type* abStatus;
  status_type* wpStatus;
  status_type* centStatus;
  Chars isAbelianFName;
  Chars isNonAbelianFName;
  Chars isTrivialFName;
  Chars isNonTrivialFName;
  Chars isCentralFName;
  Chars isNonCentralFName;
};

template <class T> class SMList : public AlgebraicObject
{
public:
```

```
  SMList( class EnumeratorProblem< T >& P, const Chars heritage);
  SMList( const SMFPGroup& ao, const SMListData& d, const Chars heritage );
  ~SMList();
  const SMListData& getData() const
  const SMFPGroup& getGroup() const;
  const AlgebraicObject& getParent()const
  LIC& lic()
  static const char* type( )
  const char* typeID( ) const
  const IconID iconID( ) const
  void printProperties(ostream& ostr) const;
  void viewStructure(ostream& ostr) const;
  void printDefinition(ostream& ostr) const;
public:
protected:
  void readMessage(istream&)
  const AlgebraicObject& theParent;
  static SMListType<T> theType;
  SMListData theData;
  LIC* theLic;
};
```

———-

19.28 SessionManager/include/SMMap.h

— **SMMap.h** —

```
#include "AlgebraicObject.h"
#include "SMFPGroup.h"
#include "MIC.h"
#include "Map.h"
#include "AbelianProblems.h"
```

———-

19.28.1 class MCM

— **class MCM** —

```
class MCM : public Supervisor
{
public:
  MCM( class SMMap& M );
  SMMap& getMap( ) const
  void viewStructure(ostream& ostr) const { }
```

```
  Subordinate<MCM,AbelianHomIsMonoComp> abelianHomIsMono;
  Subordinate<MCM,AbelianHomIsEpiComp> abelianHomIsEpi;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMMap& theMap;
  bool didFastChecks;
};
```

———-

19.28.2 class SMMap

— class SMMap —

```
class SMMap : public AlgebraicObject
{
public:
  SMMap(SMFPGroup& G, Map m, const Chars heritage);
  SMMap(SMFPGroup& G, SMFPGroup& H, Map m, const Chars heritage);
  SMMap(SMFPGroup& G);
  SMMap(SMFPGroup& G, SMFPGroup& H);
  Map getMap( ) const
  SMFPGroup& getDomain( ) const
  SMFPGroup& getRange( ) const
  MCM& mcm() const
        InformationCenter* infoCenter()
  const InformationCenter* infoCenter() const
  static const char* type( )
  const char* typeID( ) const
  const IconID iconID( ) const
  void viewStructure(ostream& ostr) const;
  void printProperties(ostream& ostr) const;
  void printDefinition(ostream& ostr) const;
protected:
  void readMessage(istream&)
public:
  MIC mic;
private:
  const Map theMap;
  SMFPGroup& theDomain;
  SMFPGroup& theRange;
  MCM* theMCM;
};
```

———-

19.29 SessionManager/include/SMObject.h

— SMObject.h —

```
#include "List.h"
#include "FEData.h"
```

———-

19.29.1 class IconID

— class IconID —

```
class IconID
{
public:
  static const IconID group;
  static const IconID subgroup;
  static const IconID SetOfWords;
  static const IconID VectorOfWords;
  static const IconID elt;
  static const IconID equation;
  static const IconID systemOfEquations;
  static const IconID map;
  static const IconID homomorphism;
  static const IconID permutation;
  static const IconID enumerator_object;
  static const IconID enumerator_problem;
  static const IconID list_object;
  static const IconID problem;
  static const IconID none;
  static const IconID do_not_display;
  bool operator == ( const IconID& ) const;
  bool operator != ( const IconID& ) const;
  friend inline ostream& operator << ( ostream& ostr, const IconID& iid )
private:
  IconID( const char* name ) : theName( name )
  const char* theName;
};
```

———-

19.29.2 class SMObject

— class SMObject —

```
class SMObject : protected FEData
{
public:
  operator OID ( ) const
  OID oid( ) const
  virtual const char* typeID( ) const = 0;
  virtual const IconID iconID( ) const = 0;
  virtual void viewStructure(ostream& ostr) const = 0;
  virtual void printProperties(ostream& ostr) const = 0;
  virtual void printDefinition(ostream& ostr) const = 0;
  virtual bool displayInFE( ) const { return true; }
  bool isComputationManager( ) const { return isCM; }
protected:
  SMObject(bool is_cm = false);
  virtual ~SMObject( )
  friend class SessionManager;
  friend class TheObjects;
  friend class Ctor;
  friend class SubordinateBase;
  friend class SMFPGroup;
  friend class SMSubgroup;
  friend class SMWord;
  friend class SMMap;
  friend class SMMagnusBreakdown;
  friend class UnboundedSupervisor;
  friend class CheckinMessage;
  virtual void readMessage(istream&) = 0;
private:
  ListOf<OID> getDependents( )
  void addDependent(OID oid)
  SMObject(const SMObject&);
  const OID theOid;
  bool isCM;
  ListOf<OID> dependents;
};
```

———-

19.30 SessionManager/include/SMPermutation.h

— **SMPermutation.h** —

```
#include "Permutation.h"
#include "AlgebraicObject.h"
```

———-

19.30.1 class SMPermutation

— class SMPermutation —

```
class SMPermutation : public AlgebraicObject
{
public:
  SMPermutation( Permutation perm, const Chars heritage );
  SMPermutation( ) : AlgebraicObject("")
  Permutation getPermutation( ) const
  static const char* type( )
  const char* typeID( ) const
  const IconID iconID( ) const
  void viewStructure(ostream& ostr) const;
  void printProperties(ostream& ostr) const;
  void printDefinition(ostream& ostr) const;
protected:
  void readMessage(istream&)
private:
  Permutation thePermutation;
};
```

———-

19.31 SessionManager/include/SMSetOfWords.h

— SMSetOfWords.h —

```
#include "AlgebraicObject.h"
#include "SMFPGroup.h"
#include "conversions.h"
```

———-

19.31.1 class SMSetOfWords

— class SMSetOfWords —

```
class SMSetOfWords : public AlgebraicObject
{
public:
  SMSetOfWords(SMFPGroup& G, const SetOf<Word>& V, const Chars heritage);
  SMSetOfWords(SMFPGroup& G)
    : AlgebraicObject(""), theGroup(G), theWords(0)
  VectorOf<Word> getWords( ) const
  SMFPGroup& getParent( ) const
```

```
  static const char* type( )
  const char* typeID( ) const
  const IconID iconID( ) const
  void viewStructure(ostream& ostr) const;
  void printProperties(ostream& ostr) const;
  void printDefinition(ostream& ostr) const;
protected:
  void readMessage(istream&)
private:
  const SetOf<Word> theWords;
  SMFPGroup& theGroup;
};
```

———-

19.32 SessionManager/include/SMSubgroup.h

— **SMSubgroup.h** —

```
#include "AlgebraicObject.h"
#include "SMFPGroup.h"
#include "SGofFreeGroup.h"
#include "NGSubgroupProblems.h"
#include "AbelianSGPresentation.h"
#include "AbelianProblems.h"
#include "ToddCoxeter.h"
#include "HToddCoxeter.h"
#include "SGNilpotentQuotients.h"
```

———-

19.32.1 class CyclicDecompositionOfFactorProperty

— **class CyclicDecompositionOfFactorProperty** —

```
class CyclicDecompositionOfFactorProperty : public AbelianGroupProperty {
public:
  CyclicDecompositionOfFactorProperty( const AbelianGroup& data )
    : AbelianGroupProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
```

```
};
```

———-

19.32.2 class AbelianSubgroupPresentationProperty

— class AbelianSubgroupPresentationProperty —

```
class AbelianSubgroupPresentationProperty
  : public AbelianSGPresentationProperty {
public:
  AbelianSubgroupPresentationProperty(const AbelianSGPresentation& data)
    : AbelianSGPresentationProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.32.3 class IsPureSubgroupProperty

— class IsPureSubgroupProperty —

```
class IsPureSubgroupProperty : public NoDataProperty {
public:
  IsPureSubgroupProperty() : NoDataProperty()
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.32.4 class IsCentralSubgroupProperty

— class IsCentralSubgroupProperty —

```
class IsCentralSubgroupProperty : public NoDataProperty {
public:
  IsCentralSubgroupProperty() : NoDataProperty()
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.32.5 class IsNormalSubgroupProperty

— class IsNormalSubgroupProperty —

```
class IsNormalSubgroupProperty : public NoDataProperty {
public:
  IsNormalSubgroupProperty() : NoDataProperty()
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.32.6 class IsAbelianSubgroupProperty

— class IsAbelianSubgroupProperty —

```
class IsAbelianSubgroupProperty : public NoDataProperty {
public:
  IsAbelianSubgroupProperty() : NoDataProperty()
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.32.7 class IsTrivialSubgroupProperty

— class IsTrivialSubgroupProperty —

```
class IsTrivialSubgroupProperty : public NoDataProperty {
public:
  IsTrivialSubgroupProperty() : NoDataProperty()
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.32.8 class IndexOfSubgroupProperty

— class IndexOfSubgroupProperty —

```
class IndexOfSubgroupProperty : public IntProperty {
public:
  IndexOfSubgroupProperty(const int& data) : IntProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.32.9 class SGNilpotentQuotientsProperty

— class SGNilpotentQuotientsProperty —

```
class SGNilpotentQuotientsProperty : public SGNilpGroupAssocProperty {
public:
  SGNilpotentQuotientsProperty(
    const AssociationsOf<int,SGOfNilpotentGroup*>& data
```

```
    ) : SGNilpGroupAssocProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.32.10 class SubgroupOfNilpotentGroupProperty

— class SubgroupOfNilpotentGroupProperty —

```
class SubgroupOfNilpotentGroupProperty : public SGOfNilpotentGroupProperty {
public:
  SubgroupOfNilpotentGroupProperty( const SGOfNilpotentGroup& data )
    : SGOfNilpotentGroupProperty( data )
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.32.11 class SIC

— class SIC —

```
class SIC : public InformationCenter, protected FEData
{
public:
  SIC( OID subgroup_oid );
  ~SIC( );
  void putHaveCyclicDecompositionOfFactor( const AbelianGroup& );
  bool haveCyclicDecompositionOfFactor( ) const;
  const AbelianGroup& getCyclicDecompositionOfFactor( ) const;
  void putHaveCyclicDecomposition( const AbelianSGPresentation& );
  void putHavePrimaryDecomposition( const AbelianSGPresentation& );
  bool haveCyclicDecomposition( ) const;
  bool havePrimaryDecomposition( ) const;
```

```
  const AbelianSGPresentation& getSGPresentation( ) const;
  void putHaveSGOfNGPreimageInitialized( const SGOfNilpotentGroup& );
  bool haveSGOfNGPreimageInitialized( ) const;
  void putHaveSGOfNGinitialized( const SGOfNilpotentGroup& );
  bool haveSGOfNGinitialized( ) const;
  const SGOfNilpotentGroup& getSGOfNilpotentGroup( ) const;
  void putHaveSGNilpotentQuotInited( const SGOfNilpotentGroup& theQuot,
             int theClass );
  bool haveSGNilpotentQuotInited(int theClass) const;
  const SGOfNilpotentGroup& getSGNilpotentQuotInited(int theClass) const;
  void putHaveCompleteCayleyGraph( const SubgroupGraph& S );
  bool haveCompleteCayleyGraph( ) const;
  const SubgroupGraph& getCompleteCayleyGraph( ) const;
  void putHaveWordDecomposer( const DecomposeInSubgroupOfFPGroup& D );
  bool haveWordDecomposer( ) const;
  const DecomposeInSubgroupOfFPGroup& getWordDecomposer( ) const;
  void putHaveSchreierTransvl(const PermutationRepresentation&);
  bool haveSchreierTransvl()const;
  const PermutationRepresentation& getSchreierTransvl()const;
  Trichotomy isPure() const;
  void putIsPure(bool ispure);
  Trichotomy isCentral() const;
  void putIsCentral( bool iscentral);
  Trichotomy isNormal() const;
  void putIsNormal( bool isnormal);
  Trichotomy isAbelian() const;
  void putIsAbelian( bool isabelian);
  Trichotomy isTrivial() const;
  void putIsTrivial( bool istrivial);
  int index() const;
  bool haveIndex() const;
  void putIndex(const int& index);
private:
  OID subgroupOID;
  AssociationsOf<int,SGOfNilpotentGroup*>& getSGNilpotentQuotients() const;
};
```

———-

19.32.12 class SCM

— class SCM —

```
class SCM : public Supervisor
{
public:
  SCM( class SMSubgroup& S );
  SMSubgroup& getSubgroup( ) const
```

```
  void viewStructure(ostream& ostr) const
  Subordinate<SCM,AbelianInvariantsOfFactor> abelianInvariantsOfFactor;
  Subordinate<SCM,AbelianSGInvariants> abelianSGInvariants;
  Subordinate<SCM,SGOfNGinitializeProblem> initializeSGOfNG;
  Subordinate<SCM,SGOfNGinitPreimageProblem> initPreimageSGOfNG;
  Subordinate<SCM,SGIndexToddCoxeter> sgToddCoxeter;
  Subordinate<SCM,SGIndexToddCoxeter> sgGHToddCoxeter;
  Subordinate<SCM,SGNilpotentQuotients> sgNilpotentQuotients;
  Subordinate<SCM,NormalClosure> normalClosure;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMSubgroup& theSubgroup;
  bool didFastChecks;
};
```

———-

19.32.13 class SMSubgroup

— **class SMSubgroup** —

```
class SMSubgroup : public AlgebraicObject
{
public:
  SMSubgroup(SMFPGroup& G, const SGofFreeGroup& S, const Chars heritage,
      Chars def = Chars());
  SMSubgroup(SMFPGroup& G)
    : AlgebraicObject(""), theGroup(G), theSubgroup( *((SGofFreeGroup*)0) ),
      sic( oid() ), theSCM( 0 )
  SGofFreeGroup getSubgroup( ) const
  SMFPGroup& getParent( ) const
  SCM& scm() const
        InformationCenter* infoCenter()
  const InformationCenter* infoCenter() const
  static const char* type( )
  const char* typeID( ) const
  const IconID iconID( ) const
  void viewStructure(ostream& ostr) const;
  void printProperties(ostream& ostr) const;
  void printDefinition(ostream& ostr) const;
protected:
  void readMessage(istream&)
public:
  SIC sic;
private:
  const SGofFreeGroup theSubgroup;
```

```
  SMFPGroup& theGroup;
  SCM* theSCM;
  Chars definition;
};
```

———-

19.33 SessionManager/include/SMVectorOfWords.h

— SMVectorOfWords.h —

```
#include "AlgebraicObject.h"
#include "SMFPGroup.h"
#include "conversions.h"
```

———-

19.33.1 class SMVectorOfWords

— class SMVectorOfWords —

```
class SMVectorOfWords : public AlgebraicObject
{
public:
  SMVectorOfWords(SMFPGroup& G, const VectorOf<Word>& V, const Chars heritage);
  SMVectorOfWords(SMFPGroup& G)
    : AlgebraicObject(""), theGroup(G), theWords(0)
  VectorOf<Word> getWords( ) const
  SMFPGroup& getParent( ) const
  static const char* type( )
  const char* typeID( ) const
  const IconID iconID( ) const
  void viewStructure(ostream& ostr) const;
  void printProperties(ostream& ostr) const;
  void printDefinition(ostream& ostr) const;
protected:
  void readMessage(istream&)
private:
  VectorOf<Word> theWords;
  SMFPGroup& theGroup;
};
```

———-

19.34 SessionManager/include/SMWord.h

— **SMWord.h** —

```
#include "AlgebraicObject.h"
#include "SMFPGroup.h"
#include "Word.h"
#include "InformationCenter.h"
#include "BaseProperties.h"
```

———-

19.34.1 class IsTrivialProperty

— **class IsTrivialProperty** —

```
class IsTrivialProperty : public NoDataProperty {
public:
  IsTrivialProperty(const Chars& descr) : NoDataProperty(descr)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.34.2 class WordOrderProperty

— **class WordOrderProperty** —

```
class WordOrderProperty : public IntegerProperty {
public:
  WordOrderProperty(const Integer& data) : IntegerProperty(data)
  PropertyType actualType  () const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.34.3 class MaximalRootProperty

— class MaximalRootProperty —

```
class MaximalRootProperty : public WordProperty {
public:
  MaximalRootProperty(const Word& data) : WordProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.34.4 class MaximalPowerProperty

— class MaximalPowerProperty —

```
class MaximalPowerProperty : public IntegerProperty {
public:
  MaximalPowerProperty(const Integer& data) : IntegerProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.34.5 class CollectedFormProperty

— class CollectedFormProperty —

```
class CollectedFormProperty : public PolyWordProperty {
public:
  CollectedFormProperty(const PolyWord& data) : PolyWordProperty(data)
  PropertyType actualType()   const
  static PropertyType type()
  Chars propertyName() const
```

```
protected:
  GenericProperty* clone() const
private:
  static const PropertyType theTypeID;
};
```

———-

19.34.6 class WIC

— class WIC —

```
class WIC : public InformationCenter, protected FEData
{
public:
  WIC( OID word_oid );
  ~WIC( );
  Trichotomy isTrivial() const;
  Trichotomy isTrivial(Chars& expl);
  void putIsTrivial(bool istrivial, const Chars& expl);
  bool haveCollectedForm() const;
  void putHaveCollectedForm( const PolyWord& p );
  const PolyWord& getCollectedForm() const;
  bool haveOrder( ) const;
  void putHaveOrder (const Integer& o);
  Integer getOrder( ) const;
  bool haveMaximalRoot() const;
  void putHaveMaximalRoot(const Word& ,const Integer&);
  void getMaximalRoot(Word& , Integer& ) const;
  Trichotomy isPower() const;
  Integer getMaximalExponent() const;
  OID wordOID;
};
```

———-

19.34.7 class WCM

— class WCM —

```
class WCM : public Supervisor
{
public:
  WCM( class SMWord& W );
  SMWord& getWord( )
  void viewStructure(ostream& ostr) const
  Subordinate<WCM,NGdecomposeWord> ngDecomposeWord;
```

```
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMWord& theWord;
  bool didFastChecks;
};
```

———-

19.34.8 class SMWord

— **class SMWord** —

```
class SMWord : public AlgebraicObject
{
public:
  SMWord(SMFPGroup& G, const Word w, const Chars heritage);
  SMWord(SMFPGroup& G) : AlgebraicObject(""), theGroup(G),
    wic( oid() ),theWCM ( 0 )
  Word getWord( ) const
  SMFPGroup& getParent( ) const
  WCM& wcm() const
        InformationCenter* infoCenter()
  const InformationCenter* infoCenter() const
  static const char* type( )
  const char* typeID( ) const
  const IconID iconID( ) const
  void viewStructure(ostream& ostr) const;
  void printProperties(ostream& ostr) const;
  void printDefinition(ostream& ostr) const;
  WIC wic;
protected:
  void readMessage(istream&)
private:
  SMFPGroup& theGroup;
  const Word theWord;
  WCM* theWCM;
};
```

———-

19.35 SessionManager/include/Supervisor.h

— **Supervisor.h** —

```
#include "ARC.h"
#include "ARCSlotID.h"
#include "OID.h"
#include "Associations.h"
#include "viewStructure.h"
#include "ComputationManager.h"
#include "TheObjects.h"
```

———-

19.35.1 class SubordinateBase

— class SubordinateBase —

```
class SubordinateBase
{
public:
  SubordinateBase(class Supervisor& boss);
  ARCSlotID arcSlotID( ) const
  virtual Supervisor& getBoss( ) const = 0;
  virtual ComputationManager* getWorker( ) const = 0;
  virtual void terminateWorker( ) = 0;
  virtual void deleteWorker() = 0;
  virtual bool exists( ) const = 0;
protected:
  friend class Supervisor;
  virtual void acceptAllocation(OID oid, ARCSlotID, ARC, bool overrides=true)
  void addDependent( SMObject& smo, OID oid ) const;
private:
  SubordinateBase( const SubordinateBase& );
  ARCSlotID asi;
};

template <class B, class W> class Subordinate : public SubordinateBase
{
friend class MirrorSubordinate;
public:
  Subordinate(B& boss)
    : SubordinateBase( boss ), theBoss( boss ), theWorker( 0 )
  W* operator -> ( )
  Supervisor& getBoss( ) const
  ComputationManager* getWorker( ) const
  void terminateWorker( )
  bool exists( ) const
protected:
  void acceptAllocation(OID oid, ARCSlotID asi, ARC arc, bool overrides)
private:
  void deleteWorker()
```

```
  B& theBoss;
  W* theWorker;
};
```

———-

19.35.2 class MirrorSubordinate

— class MirrorSubordinate —

```
class MirrorSubordinate : public SubordinateBase
{
public:
  MirrorSubordinate(class Supervisor& boss, SubordinateBase&);
  Supervisor& getBoss( ) const
  bool exists( ) const
  ComputationManager* getWorker( ) const;
  void terminateWorker( );
  void deleteWorker();
protected:
  void acceptAllocation(OID oid, ARCSlotID asi, ARC arc, bool overrides);
private:
  class Supervisor& theBoss;
  SubordinateBase& mirror;
  bool firstAccessToWorker;
};
```

———-

19.35.3 class Supervisor

— class Supervisor —

```
class Supervisor : public ComputationManager
{
public:
  bool isSupervisor( ) const
  void supervise( );
protected:
  friend class MirrorSubordinate;
  Supervisor( bool display_in_fe )
    : ComputationManager( display_in_fe ), ARCSlotID_Counter( 1 ),
      theParameters( NULL )
  void readMessage(istream& istr);
  void forwardAllocation(OID oid, ARCSlotID to, ARCSlotID from, ARC arc,
       bool overrides);
  void adminTerminate( );
```

```
  void submit(ostream& ostr,const EnumertatorProblemView& pv) const
  const ParameterStructure& getParameters()const
private:
  SubordinateBase* dereference(ARCSlotID asi) const;
  friend class SubordinateBase;
  ARCSlotID hire( SubordinateBase* worker );
  AssociationsOf<int,SubordinateBase*> theWorkers;
  AssociationsOf<int,bool> theCrashedWorkers;
  int ARCSlotID_Counter;
  ParameterStructure* theParameters;
};
```

———-

19.35.4 class UnboundedSupervisor

— class UnboundedSupervisor —

```
class UnboundedSupervisor : public ComputationManager
{
protected:
  UnboundedSupervisor( bool display_in_fe )
    : ComputationManager( display_in_fe )
  void remove(OID oid)
  void addDependent(OID oid)
};
```

———-

19.36 SessionManager/include/TheObjects.h

— TheObjects.h —

```
#include "Set.h"
#include "Associations.h"
#include "Chars.h"
```

———-

19.36.1 class TheObjects

— class TheObjects —

```
class TheObjects
{
public:
  static bool isRunning(OID oid);
  static bool isStalled(OID oid);
  static SetOf<OID> remove(OID oid);
  static SMObject* get(OID oid);
  static SetOf<OID> dependencyClosure(OID oid);
  static Chars name(OID oid);
private:
  friend class SMObject;
  friend class SessionManager;
  friend class Ctor;
  friend class UnboundedSupervisor;
  friend class ResourceManager;
  friend class ExtendToHomProblem;
  static OID reserveOid( );
  static void enroll(SMObject* newObject);
  static void passControl( );
  static void setName(OID oid, Chars name);
  static void resize(int newLength);
  static const int minimumLength = 16;
  static int theObjectsLength;
  static SMObject** theObjects;
  static int CMToGetControl;
  static AssociationsOf<OID,Chars>* theNames;
};
```

———-

19.37 SessionManager/include/ViewContents.h

— **ViewContents.h** —

```
#include "global.h"
#include "values.h"
#include "Chars.h"
#include "Word.h"
#include "iostream.h"
#include "ARCSlotID.h"
#include "FreeGroup.h"
#include "OID.h"
#include "FEData.h"
```

———-

19.37.1 class PValue

— class PValue —

```
class PValue
{
public:
  PValue(int p): ivalue(p), theType(INT)
  PValue(Chars p): cvalue(p), theType(CHARS)
  PValue(Word p): wvalue(p), theType(WORD)
  PValue(const PValue& p):
    ivalue(p.ivalue),
    cvalue(p.cvalue),
    wvalue(p.wvalue),
    theType(p.theType)
  operator int( )
  operator Chars( )
  operator Word( )
private:
  enum VType { INT, CHARS, WORD };
  int ivalue;
  Chars cvalue;
  Word wvalue;
  VType theType;
};
```

———-

19.37.2 class ViewContent

— class ViewContent —

```
class ViewContent
{
public:
  ViewContent(const Chars& name): theName(name)
  ViewContent(const ViewContent& v):  theName(v.theName)
  virtual const char* type( ) const = 0;
  virtual ViewContent* clone( ) const = 0;
  const Chars& name()const
  virtual void submit(ostream& ostr)const=0;
  virtual PValue getValue(Chars name ) const = 0;
  virtual void setValue( istream& is,Chars& errMsg,Chars name) = 0;
  virtual const ViewContent* contentOf( const Chars& name )const = 0;
private:
  Chars theName;
};
```

———-

19.37.3 struct ViewContentCell

— **struct ViewContentCell** —

```
struct ViewContentCell
{
  ViewContentCell( const ViewContent& content): c(content.clone()), n(NULL)
  const ViewContent& content() const
  ViewContentCell* next()
  void setNext(ViewContentCell* next)
  ViewContent* pcontent()
private:
  ViewContent* c;
  ViewContentCell* n;
};
```

———-

19.37.4 class EditInteger

— **class EditInteger** —

```
class EditInteger : public  ViewContent
{
public:
  EditInteger( const Chars& name,const Chars& text,int change = 0,int def = 0,
         int minV = 0,int maxV = MAXINT):
    ViewContent(name),
    theText( text ),
    changeable( change ),
    defaultValue( def ),
    minValue(minV),
    maxValue(maxV)
  EditInteger(const EditInteger& i):
    ViewContent(i),
    theText( i.theText ),
    changeable( i.changeable ),
    defaultValue( i.defaultValue ),
    minValue(i.minValue),
    maxValue(i.maxValue)
  EditInteger* clone() const
  const char* type( ) const
  PValue getValue(Chars name = Chars() ) const
  void setValue( istream& is,Chars& errMsg,Chars name = Chars())
  const ViewContent* contentOf( const Chars& n ) const
```

```
  void submit(ostream& ostr)const;
private:
  int changeable;
  int defaultValue;
  int minValue;
  int maxValue;
  Chars theText;
};
```

———-

19.37.5 class EditBool

— class EditBool —

```
class EditBool : public  ViewContent
{
public:
  EditBool( const Chars& name,const Chars& text,int change = 0,int def = 0):
    ViewContent(name),
    theText( text ),
    changeable( change ),
    defaultValue( def )
  EditBool(const EditBool& i):
    ViewContent(i),
    theText( i.theText ),
    changeable( i.changeable ),
    defaultValue( i.defaultValue )
  EditBool* clone() const
  const char* type( ) const
  PValue getValue(Chars name = Chars() ) const
  void setValue( istream& is,Chars& errMsg,Chars name = Chars())
  const ViewContent* contentOf( const Chars& n ) const
  void submit(ostream& ostr)const;
private:
  int changeable;
  int defaultValue;
  Chars theText;
};
```

———-

19.37.6 class EditText

— class EditText —

```
class EditText : public  ViewContent
{
public:
  EditText( const Chars& name,const Chars& text,int change = 0,Chars def = ""):
    ViewContent(name),
    theText( text ),
    changeable( change ),
    defaultValue( def )
  EditText(const EditText& c):
    ViewContent(c),
    theText( c.theText ),
    changeable( c.changeable ),
    defaultValue( c.defaultValue )
  virtual EditText* clone() const
  const char* type( ) const
  virtual PValue getValue(Chars name = Chars() ) const
  virtual void setValue( istream& is, Chars& errMsg, Chars name = Chars());
  const ViewContent* contentOf( const Chars& n ) const
  void submit(ostream& ostr)const;
protected:
   Chars defaultValue;
   Chars theText;
private:
  int changeable;
};
```

———-

19.37.7 class EditWord

— class EditWord —

```
class EditWord : public  EditText
{
public:
  EditWord( const Chars& name,const Chars& text,int change = 0,
      const FreeGroup& f = FreeGroup(), const Word& w = Word() ):
    EditText( name, text, change),
      theGroup( f ),
      theWord( w )
  EditWord(const EditWord& c):
    EditText(c),
    theGroup( c.theGroup ),
    theWord( c.theWord )
  EditWord* clone() const
  PValue getValue(Chars name = Chars() ) const
  void setValue( istream& is, Chars& errMsg, Chars name = Chars());
private:
```

```
  Word theWord;
  FreeGroup theGroup;
};
```

———-

19.37.8 class GroupWindow

— class GroupWindow —

```
class GroupWindow : public  ViewContent
{
public:
  GroupWindow(const Chars& name):
    ViewContent(name),
    children(NULL),
    last(NULL)
  GroupWindow(const GroupWindow& gw);
  ~GroupWindow();
  GroupWindow* clone() const
  virtual const char* type( ) const
  virtual Chars properties() const
  void submit(ostream& ostr) const;
  PValue getValue(Chars name) const
  void setValue(istream& is, Chars& errMsg,Chars name)
  void add(const ViewContent& c);
  const ViewContent* contentOf( const Chars& name ) const;
protected:
  ViewContentCell* children;
  ViewContentCell* last;
};
```

———-

19.37.9 class RadioButtonGroup

— class RadioButtonGroup —

```
class RadioButtonGroup :  public GroupWindow
{
public:
  RadioButtonGroup(const Chars& name): GroupWindow( name )
  RadioButtonGroup(const RadioButtonGroup& bg):
    GroupWindow( bg ),
    selButton( bg.selButton)
  RadioButtonGroup* clone() const
  const char* type( ) const
```

```
  void add(const ViewContent& c);
  PValue getValue(Chars name) const
  void setValue( istream& is, Chars& errMsg, Chars name)
private:
  Chars selButton;
};
```

———-

19.37.10 class RadioButton

— class RadioButton —

```
class RadioButton :  public GroupWindow
{
public:
  RadioButton(const Chars& name, const Chars& text, int change = 0)
    : GroupWindow( name ), theText(text),changeable( change )
  RadioButton(const RadioButton& bg):
    GroupWindow( bg ),
    theText(bg.theText),
    changeable( bg.changeable )
  RadioButton* clone() const
  const char* type( ) const
  Chars properties() const
private:
  int changeable;
  Chars theText;
};
```

———-

19.37.11 class Subproblem

— class Subproblem —

```
class Subproblem : public ViewContent, public FEData
{
public:
  Subproblem(const Chars& name, ARCSlotID asi, const Text& d, int value = 0)
    : ViewContent( name ),
      numOfARCs( value ),
      description( d ),
      theASI( asi )
  Subproblem(const Subproblem& s)
    : ViewContent(s),
      numOfARCs( s.numOfARCs),
```

```
      description( s.description ),
      theASI( s.theASI )
  const char* type( ) const
  Subproblem* clone( ) const
  PValue getValue(class Chars) const
  void setValue(istream& is, Chars& errMsg, class Chars)
  void submit(ostream& ostr)const;
  const ViewContent* contentOf(const class Chars & )const
private:
  ARCSlotID theASI;
  int numOfARCs;
  Chars description;
};
```

———-

19.37.12 class ParameterStructure

— class ParameterStructure —

```
class ParameterStructure
{
public:
  ParameterStructure(OID o,const GroupWindow& p)
    : theParameters( p ), bossOID( o )
  ParameterStructure(const ParameterStructure& p):
    theParameters( p.theParameters ), bossOID( p.bossOID )
  void read( istream& istr );
  PValue getValue(const Chars& name)const
private:
  GroupWindow theParameters;
  OID bossOID;
};
```

———-

19.38 SessionManager/include/viewStructure.h

— viewStructure.h —

```
#include "FEData.h"
#include "ARCSlotID.h"
#include "ViewContents.h"
```

———-

19.38.1 class ObjectView

— class ObjectView —

```
class ObjectView
{
public:
  ObjectView(ostream& ostr, OID oid, Chars heritage);
};
```

———-

19.38.2 class ProblemView

— class ProblemView —

```
class ProblemView : public FEData
{
public:
  ProblemView(ostream& o, OID oid, const Text& title, const Text& banner,
        const char* helpID = "none", const Chars abbreviation = "none");
  void startItemGroup( );
  void startItemGroup(const Expression& E);
  void add(const Text& name,ARCSlotID asi,int value = 0, bool enabled = true);
  void done( );
private:
  enum StateType { INIT, ITEM_GROUP, DONE };
  StateType state;
  ostream& ostr;
};
```

———-

19.38.3 class EnumertatorProblemView

— class EnumertatorProblemView —

```
class EnumertatorProblemView : public FEData
{
 public:
  EnumertatorProblemView(OID oid,const Chars& title, const Chars& banner,
       const char* helpID = "none",
       const Chars abbreviation = "none")
    : theOID( oid ),
    theTitle( title ),
    theBanner( banner ),
```

```
    theHelpID( helpID ),
    theAbbreviation( abbreviation ),
    theParameters("Parameters"),
    theSubproblems("Subproblems")
  void addProblem(const Subproblem& sp)
  void addParameter(const ViewContent& c)
  void done(ostream& ostr) const;
  ParameterStructure extractParameters()const
 private:
  OID theOID;
  Chars theTitle;
  Chars theBanner;
  Chars theHelpID;
  Chars theAbbreviation;
  GroupWindow theParameters;
  GroupWindow theSubproblems;
};
```

———-

20 The SM Apps classes

20.1 SMApps/include/AbelianInvariants.h

— **AbelianInvariants.h** —

```
#include "Supervisor.h"
#include "AbelianGroup.h"
#include "ARCer.h"
```

———-

20.1.1 class AbelianInvariantsARCer

— **class AbelianInvariantsARCer** —

```
class AbelianInvariantsARCer : public ARCer
{
public:
  AbelianInvariantsARCer( ComputationManager& boss )
    : ARCer( boss ), AG( 0 )
  ~AbelianInvariantsARCer( )
  void setArguments( const FPGroup&, bool );
  AbelianGroup getCyclicDecomposition();
  void runComputation( );
  void writeResults( ostream& );
```

```
  void readResults( istream& );
private:
  FPGroup G;
  bool bMakeFile;
  AbelianGroup* AG;
};
```

———-

20.1.2 class AbelianInvariants

— class AbelianInvariants —

```
class AbelianInvariants : public ComputationManager
{
public:
  AbelianInvariants(class GCM& gcm);
  void takeControl( );
  void start( )
  void terminate( )
private:
  class GCM& theGCM;
  AbelianInvariantsARCer arcer;
};
```

———-

20.1.3 class AbelianInvariantsOfFactor

— class AbelianInvariantsOfFactor —

```
class AbelianInvariantsOfFactor : public ComputationManager
{
public:
  AbelianInvariantsOfFactor(class SCM& scm);
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SCM& theSCM;
  AbelianInvariantsARCer arcer;
};
```

———-

20.1.4 class AbelInvariantsProb

— class AbelInvariantsProb —

```
class AbelInvariantsProb : public Supervisor
{
public:
  AbelInvariantsProb(class SMFPGroup& G);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMFPGroup& theGroup;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.1.5 class AbelianRank

— class AbelianRank —

```
class AbelianRank : public ComputationManager
{
public:
  AbelianRank(class GCM& gcm);
  ~AbelianRank( );
  void takeControl( );
  void start( );
  void terminate( )
private:
  SMFPGroup& theGroup;
  class AbelianInfinitenessProblem* theProb;
};
```

———-

20.1.6 class AbelianPrimesARCer

— class AbelianPrimesARCer —

```
class AbelianPrimesARCer : public ARCer
{
public:
  AbelianPrimesARCer( ComputationManager& boss )
```

```
    : ARCer( boss ), AG( 0 )
  ~AbelianPrimesARCer( )
  void setArguments( const AbelianGroup& );
  AbelianGroup getPrimaryDecomposition();
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  FPGroup G;
  bool bMakeFile;
  AbelianGroup* AG;
};
```

———-

20.1.7 class AbelianPrimes

— class AbelianPrimes —

```
class AbelianPrimes : public ComputationManager
{
public:
   AbelianPrimes(class GCM& gcm);
  void takeControl( );
  void start( )
  void terminate( )
private:
   class GCM& theGCM;
   AbelianPrimesARCer A;
   bool primesStarted;
};
```

———-

20.1.8 class AbelianPrimeDecomp

— class AbelianPrimeDecomp —

```
class AbelianPrimeDecomp : public Supervisor
{
public:
  AbelianPrimeDecomp(class SMFPGroup& smg);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
```

```
  SMFPGroup& theGroup;
  MirrorSubordinate abelianPrimes;
  MirrorSubordinate abelianInvariants;
  bool abDone;
};
```

———-

20.2 SMApps/include/AbelianProblems.h

— **AbelianProblems.h** —

```
#include "ComputationManager.h"
#include "Supervisor.h"
#include "AbelianInvariants.h"
#include "AbelianSGPresentation.h"
```

———-

20.2.1 class AbelianWordProblem

— **class AbelianWordProblem** —

```
class AbelianWordProblem : public Supervisor
{
public:
  AbelianWordProblem(class SMWord& w);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMWord& theWord;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.2 class AbelianSGWordProblem

— **class AbelianSGWordProblem** —

```
class AbelianSGWordProblem : public Supervisor
{
public:
```

```
  AbelianSGWordProblem(class SMSubgroup& S, class SMWord& w);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  SMWord& theWord;
  MirrorSubordinate abelianSGInvariants;
};
```

———-

20.2.3 class AbelianQuotientWordProblem

— class AbelianQuotientWordProblem —

```
class AbelianQuotientWordProblem : public Supervisor
{
public:
  AbelianQuotientWordProblem(class SMSubgroup& S, class SMWord& w);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  SMWord& theWord;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.4 class IsWordNthPower

— class IsWordNthPower —

```
class IsWordNthPower : public Supervisor
{
public:
  IsWordNthPower(class SMWord& w, int p = 1);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  Integer thePower;
```

```
  SMFPGroup& theGroup;
  SMWord& theWord;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.5 class AbelianIsomProblem

— class AbelianIsomProblem —

```
class AbelianIsomProblem : public Supervisor
{
public:
  AbelianIsomProblem(class SMFPGroup& g1, class SMFPGroup& g2);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMFPGroup& G1;
  SMFPGroup& G2;
  MirrorSubordinate abelianInvariants1;
  MirrorSubordinate abelianInvariants2;
};
```

———-

20.2.6 class AbelianTorsionFreeRankProblem

— class AbelianTorsionFreeRankProblem —

```
class AbelianTorsionFreeRankProblem : public Supervisor
{
public:
  AbelianTorsionFreeRankProblem(class SMFPGroup& G);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMFPGroup& theGroup;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.7 class AbelianOrderOfTheTorsionSubgroupProblem

— class AbelianOrderOfTheTorsionSubgroupProblem —

```
class AbelianOrderOfTheTorsionSubgroupProblem : public Supervisor
{
public:
  AbelianOrderOfTheTorsionSubgroupProblem(class SMFPGroup& G);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMFPGroup& theGroup;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.8 class AbelianOrderProblem

— class AbelianOrderProblem —

```
class AbelianOrderProblem : public Supervisor
{
public:
  AbelianOrderProblem(class SMFPGroup& G);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMFPGroup& theGroup;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.9 class AbelianOrderOfEltProblem

— class AbelianOrderOfEltProblem —

```
class AbelianOrderOfEltProblem : public Supervisor
{
public:
  AbelianOrderOfEltProblem(class SMWord& w);
```

```
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMWord& theWord;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.10 class AbelianEqualityProblem

— class AbelianEqualityProblem —

```
class AbelianEqualityProblem : public Supervisor
{
public:
  AbelianEqualityProblem(class SMWord& w1, class SMWord& w2);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMWord& word1;
  SMWord& word2;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.11 class WordInSGOfAbelian

— class WordInSGOfAbelian —

```
class WordInSGOfAbelian : public Supervisor
{
public:
  WordInSGOfAbelian(class SMSubgroup& S, class SMWord& w);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  SMWord& theWord;
  MirrorSubordinate abelianInvariants;
```

```
};
```

———-

20.2.12 class WordPowerInSGOfAbelian

— class WordPowerInSGOfAbelian —

```
class WordPowerInSGOfAbelian : public Supervisor
{
public:
  WordPowerInSGOfAbelian(class SMSubgroup& S, class SMWord& w);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  SMWord& theWord;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.13 class WordInSGBasisOfAbelian

— class WordInSGBasisOfAbelian —

```
class WordInSGBasisOfAbelian : public Supervisor
{
public:
  WordInSGBasisOfAbelian(class SMSubgroup& S, class SMWord& w);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  SMWord& theWord;
  MirrorSubordinate abelianSGInvariants;
};
```

———-

20.2.14 class AbelianIsSGEqualToTheGroup

— class AbelianIsSGEqualToTheGroup —

```
class AbelianIsSGEqualToTheGroup : public Supervisor
{
public:
  AbelianIsSGEqualToTheGroup(class SMSubgroup& S);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.15 class AbelianSGIndexProblem

— class AbelianSGIndexProblem —

```
class AbelianSGIndexProblem : public Supervisor
{
public:
  AbelianSGIndexProblem(class SMSubgroup& S);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.16 class AbelianIsSGIsolatedProblem

— class AbelianIsSGIsolatedProblem —

```
class AbelianIsSGIsolatedProblem : public Supervisor
{
public:
  AbelianIsSGIsolatedProblem(class SMSubgroup& S);
```

```
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.17 class AbelianSGContainmentProblem

— class AbelianSGContainmentProblem —

```
class AbelianSGContainmentProblem : public Supervisor
{
public:
  AbelianSGContainmentProblem(class SMSubgroup& S1, class SMSubgroup& S2);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& subgroup1;
  SMSubgroup& subgroup2;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.18 class AbelianSGDirectFactorProblem

— class AbelianSGDirectFactorProblem —

```
class AbelianSGDirectFactorProblem : public Supervisor
{
public:
  AbelianSGDirectFactorProblem(SMSubgroup& S1, SMSubgroup& S2);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& subgroup1;
  SMSubgroup& subgroup2;
  MirrorSubordinate abelianSGInvariants1;
```

```
  MirrorSubordinate abelianSGInvariants2;
  bool sgDone1, sgDone2;
};
```

———-

20.2.19 class FindCanonicalSmithPresentation

— class FindCanonicalSmithPresentation —

```
class FindCanonicalSmithPresentation : public Supervisor
{
public:
  FindCanonicalSmithPresentation(class SMFPGroup& G);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMFPGroup& theGroup;
  MirrorSubordinate abelianInvariants;
  void makePresentation();
};
```

———-

20.2.20 class AbelianPHeightOfEltProblem

— class AbelianPHeightOfEltProblem —

```
class AbelianPHeightOfEltProblem : public Supervisor
{
public:
  AbelianPHeightOfEltProblem(class SMWord& w, int p = 1);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMWord& theWord;
  bool notPrime;
  bool abDone;
  Integer P;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate abelianPrimes;
};
```

———-

20.2.21 class AbelianComputeTorsionSubgroup

— class AbelianComputeTorsionSubgroup —

```
class AbelianComputeTorsionSubgroup : public Supervisor
{
public:
  AbelianComputeTorsionSubgroup(class SMFPGroup& G);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMFPGroup& theGroup;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.22 class AbelianEltPowerSubgrARCer

— class AbelianEltPowerSubgrARCer —

```
class AbelianEltPowerSubgrARCer : public ARCer
{
public:
  AbelianEltPowerSubgrARCer( ComputationManager& boss )
    : ARCer( boss ), AG( 0 ), result(0)
  ~AbelianEltPowerSubgrARCer( )
  void setArguments( const AbelianGroup& group, const Word& w,
                     const VectorOf<Word>& sG);
  Integer getPowerOfEltSubgrup();
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  AbelianGroup* AG;
  Word theWord;
  VectorOf<Word> theSubgroup;
  Integer result;
};
```

———-

20.2.23 class AbelianEltPowerSubgrComp

— class AbelianEltPowerSubgrComp —

```
class AbelianEltPowerSubgrComp : public ComputationManager
{
public:
   AbelianEltPowerSubgrComp(class AbelianEltPowerSubgr& gcm);
  void takeControl( );
  bool isDone() const
  void start( )
  void terminate( )
private:
   class AbelianEltPowerSubgr& theSupervisor;
   AbelianEltPowerSubgrARCer A;
   bool finished;
};
```

———-

20.2.24 class AbelianEltPowerSubgr

— class AbelianEltPowerSubgr —

```
class AbelianEltPowerSubgr : public Supervisor
{
public:
  AbelianEltPowerSubgr(const class SMSubgroup& smg,const class SMWord& smw);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
  const SMWord& getWord() const
  const SMSubgroup& getSubgroup() const
private:
  const SMWord& theWord;
  const SMSubgroup& theSubgroup;
  Subordinate<AbelianEltPowerSubgr,AbelianEltPowerSubgrComp>
    abelianEltPowerSubgr;
};
```

———-

20.2.25 class AbelianPowerProblem

— class AbelianPowerProblem —

```
class AbelianPowerProblem : public Supervisor
{
public:
  AbelianPowerProblem(class SMWord& w);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMWord& theWord;
  bool abDone;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate abelianPrimes;
};
```

———-

20.2.26 class AbelianInvariantsOfSGARCer

— class AbelianInvariantsOfSGARCer —

```
class AbelianInvariantsOfSGARCer : public ARCer
{
public:
  AbelianInvariantsOfSGARCer( ComputationManager& boss )
    : ARCer( boss ), AG( NULL ), ASG( NULL ), presentationFound( false )
  ~AbelianInvariantsOfSGARCer( )
  void setArguments( const AbelianGroup&, const VectorOf<Word>&);
  AbelianSGPresentation getSGCyclicDecomposition();
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  VectorOf<Word> theSGGens;
  bool presentationFound;
  AbelianGroup* AG;
  AbelianSGPresentation* ASG;
};
```

———-

20.2.27 class AbelianSGInvariants

— class AbelianSGInvariants —

```
class AbelianSGInvariants : public ComputationManager
{
```

```
public:
  AbelianSGInvariants(class SCM& scm);
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SCM& theSCM;
  bool presentationStarted;
  AbelianInvariantsOfSGARCer arcer;
};
```

———-

20.2.28 class AbelianSGCyclicDecomposition

— class AbelianSGCyclicDecomposition —

```
class AbelianSGCyclicDecomposition : public Supervisor
{
public:
  AbelianSGCyclicDecomposition(class SMSubgroup& smg);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  MirrorSubordinate abelianSGInvariants;
};
```

———-

20.2.29 class AbelianPrimesOfSGARCer

— class AbelianPrimesOfSGARCer —

```
class AbelianPrimesOfSGARCer : public ARCer
{
public:
  AbelianPrimesOfSGARCer( ComputationManager& boss )
    : ARCer( boss ), ASG( NULL )
  ~AbelianPrimesOfSGARCer( )
  void setArguments( const AbelianSGPresentation&);
  AbelianSGPresentation getSGPrimaryDecomposition();
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
```

```
private:
  AbelianSGPresentation* ASG;
};
```

———-

20.2.30 class AbelianSGPrimes

— class AbelianSGPrimes —

```
class AbelianSGPrimes : public ComputationManager
{
public:
  AbelianSGPrimes(class AbelianSGPrimesDecomposition& sup);
  void takeControl( );
  void start( )
  void terminate( )
private:
  class AbelianSGPrimesDecomposition& theSupervisor;
  bool primesStarted;
  AbelianPrimesOfSGARCer arcer;
};
```

———-

20.2.31 class AbelianSGPrimesDecomposition

— class AbelianSGPrimesDecomposition —

```
class AbelianSGPrimesDecomposition : public Supervisor
{
public:
  AbelianSGPrimesDecomposition(class SMSubgroup& smg);
  void viewStructure(ostream& ostr) const;
  SMSubgroup& getSubgroup() const
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  MirrorSubordinate abelianSGInvariants;
  Subordinate<AbelianSGPrimesDecomposition,AbelianSGPrimes> abelianSGPrimes;
  bool abSGDone;
};
```

———-

20.2.32 class AbelianSGOrder

— class AbelianSGOrder —

```
class AbelianSGOrder : public Supervisor
{
public:
  AbelianSGOrder(class SMSubgroup& smg);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  MirrorSubordinate abelianSGInvariants;
};
```

———-

20.2.33 class AbelianSGMinGens

— class AbelianSGMinGens —

```
class AbelianSGMinGens : public Supervisor
{
public:
  AbelianSGMinGens(SMSubgroup& S);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMFPGroup& theGroup;
  SMSubgroup& theSubgroup;
  MirrorSubordinate abelianSGInvariants;
};
```

———-

20.2.34 class AbelianMaximalRootARCer

— class AbelianMaximalRootARCer —

```
class AbelianMaximalRootARCer : public ARCer
{
public:
```

```
  AbelianMaximalRootARCer( ComputationManager& boss )
    : ARCer( boss ), AG( 0 ), thePower(0)
  ~AbelianMaximalRootARCer( )
  void setArguments( const AbelianGroup& group, const Word& w);
  Integer getPower() const;
  Word getRoot() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  AbelianGroup* AG;
  Integer thePower;
  Word theRoot;
  Word theWord;
};
```

———-

20.2.35 class AbelianMaximalRootComp

— class AbelianMaximalRootComp —

```
class AbelianMaximalRootComp : public ComputationManager
{
public:
   AbelianMaximalRootComp(class AbelianMaximalRoot& sup);
  void takeControl( );
  void start( )
  void terminate( )
private:
   class AbelianMaximalRoot& theSupervisor;
   AbelianMaximalRootARCer A;
   bool rootStarted;
};
```

———-

20.2.36 class AbelianMaximalRoot

— class AbelianMaximalRoot —

```
class AbelianMaximalRoot : public Supervisor
{
public:
  AbelianMaximalRoot(class SMWord& smw);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
```

```
  void start( )
  void terminate( )
  SMWord& getWord() const
private:
  SMWord& theWord;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate abelianPrimes;
  bool abDone;
  bool abPDone;
  Subordinate<AbelianMaximalRoot,AbelianMaximalRootComp> abelianMaximalRoot;
};
```

———-

20.2.37 class AbelianIsIsomorphicSG

— class AbelianIsIsomorphicSG —

```
class AbelianIsIsomorphicSG : public Supervisor
{
public:
  AbelianIsIsomorphicSG(class SMSubgroup& smg1, class SMSubgroup& smg2);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup1;
  SMSubgroup& theSubgroup2;
  MirrorSubordinate abelianSG1Invariants;
  MirrorSubordinate abelianSG2Invariants;
  bool sg1abDone;
  bool sg2abDone;
};
```

———-

20.2.38 class AbelianTorsionFreeRankOfSG

— class AbelianTorsionFreeRankOfSG —

```
class AbelianTorsionFreeRankOfSG : public Supervisor
{
public:
  AbelianTorsionFreeRankOfSG(class SMSubgroup& SG);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
```

```
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  MirrorSubordinate abelianSGInvariants;
};
```

———-

20.2.39 class AbelianOrderOfTheTorsionSubgroupOfSG

— class AbelianOrderOfTheTorsionSubgroupOfSG —

```
class AbelianOrderOfTheTorsionSubgroupOfSG : public Supervisor
{
public:
  AbelianOrderOfTheTorsionSubgroupOfSG(class SMSubgroup& SG);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  MirrorSubordinate abelianSGInvariants;
};
```

———-

20.2.40 class EltPrimeForm

— class EltPrimeForm —

```
class EltPrimeForm : public Supervisor
{
public:
  EltPrimeForm(class SMWord& smw);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMWord& theWord;
  MirrorSubordinate abelianPrimes;
  MirrorSubordinate abelianInvariants;
  bool abDone;
};
```

———-

20.2.41 class virtualFreeCompARCer

— class virtualFreeCompARCer —

```
class virtualFreeCompARCer : public ARCer
{
public:
  virtualFreeCompARCer( ComputationManager& boss )
    : ARCer( boss ), AG( 0 )
  ~virtualFreeCompARCer( )
  void setArguments( const AbelianGroup& group, const VectorOf<Word>& sG);
  VectorOf<Word> getVFComplement()const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  AbelianGroup* AG;
  VectorOf<Word> theSubgroup;
  VectorOf<Word> result;
};
```

———-

20.2.42 class virtualFreeComp

— class virtualFreeComp —

```
class virtualFreeComp  : public Supervisor
{
public:
  virtualFreeComp(class SMSubgroup& sms);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  virtualFreeCompARCer arcer;
  bool started;
};
```

———-

20.2.43 class SubgroupIsolatorARCer

— class SubgroupIsolatorARCer —

```
class SubgroupIsolatorARCer : public ARCer
{
public:
  SubgroupIsolatorARCer( ComputationManager& boss )
    : ARCer( boss ), AG( 0 )
  ~SubgroupIsolatorARCer( )
  void setArguments( const AbelianGroup& group, const VectorOf<Word>& sG);
  VectorOf<Word> getIsolator()const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  AbelianGroup* AG;
  VectorOf<Word> theSubgroup;
  VectorOf<Word> result;
};
```

———-

20.2.44 class SubgroupIsolator

— class SubgroupIsolator —

```
class SubgroupIsolator  : public Supervisor
{
public:
  SubgroupIsolator(class SMSubgroup& sms);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  SubgroupIsolatorARCer arcer;
  bool started;
};
```

———-

20.2.45 class AbelianSGPurityARCer

— class AbelianSGPurityARCer —

```
class AbelianSGPurityARCer : public ARCer
{
public:
  AbelianSGPurityARCer( ComputationManager& boss, class SMSubgroup& subgroup)
    : ARCer( boss ), isPure( false ), theSubgroup( subgroup )
  bool isPureSubgroup() const;
  bool computePurity() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  bool isPure;
  const SMSubgroup& theSubgroup;
};
```

———-

20.2.46 class AbelianSGPurityProblem

— class AbelianSGPurityProblem —

```
class AbelianSGPurityProblem  : public Supervisor
{
public:
  AbelianSGPurityProblem(class SMSubgroup& sms);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate abelianPrimes;
  MirrorSubordinate abelianSGInvariants;
  bool abDone;
  bool PDDone;
  bool sgAbDone;
  AbelianSGPurityARCer arcer;
};
```

———-

20.2.47 class AbelianSGGenedByWordPurityProblem

— class AbelianSGGenedByWordPurityProblem —

```
class AbelianSGGenedByWordPurityProblem  : public Supervisor
{
public:
  AbelianSGGenedByWordPurityProblem(class SMWord& smw);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMWord& theWord;
  SMFPGroup& theGroup;
  MirrorSubordinate abelianPrimes;
  MirrorSubordinate abelianInvariants;
  bool abDone;
};
```

———-

20.2.48 class AbelianDoesGensSummand

— class AbelianDoesGensSummand —

```
class AbelianDoesGensSummand  : public Supervisor
{
public:
  AbelianDoesGensSummand(class SMWord& smw);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMWord& theWord;
  SMFPGroup& theGroup;
  MirrorSubordinate abelianPrimes;
  MirrorSubordinate abelianInvariants;
  bool abDone;
};
```

———-

20.2.49 class AbelianSGEqualityProblem

— class AbelianSGEqualityProblem —

```
class AbelianSGEqualityProblem : public Supervisor
{
public:
```

```
  AbelianSGEqualityProblem(class SMSubgroup& S1, class SMSubgroup& S2);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& subgroup1;
  SMSubgroup& subgroup2;
  bool s1AbDone;
  bool s2AbDone;
  MirrorSubordinate s1AbelianInvariantsOfFactor;
  MirrorSubordinate s2AbelianInvariantsOfFactor;
};
```

———-

20.2.50 class IsAbelianWordPowerOfSecondArcer

— class IsAbelianWordPowerOfSecondArcer —

```
class IsAbelianWordPowerOfSecondArcer : public ARCer
{
public:
  IsAbelianWordPowerOfSecondArcer( ComputationManager& boss)
    : ARCer( boss ), power( 0 ), AG(NULL)
   ~IsAbelianWordPowerOfSecondArcer( )
void IsAbelianWordPowerOfSecondArcer::setArguments(const AbelianGroup& abGroup,
                                const SMWord& w1,const SMWord& w2);
  int getPower() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  int power;
  AbelianGroup* AG;
  Word theWord1;
  Word theWord2;
};
```

———-

20.2.51 class IsAbelianWordPowerOfSecond

— class IsAbelianWordPowerOfSecond —

```
class IsAbelianWordPowerOfSecond  : public Supervisor
{
```

```
public:
  IsAbelianWordPowerOfSecond(class SMWord& w1,class SMWord& w2);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMWord& theWord1;
  SMWord& theWord2;
  bool arcerSet;
  MirrorSubordinate abelianInvariants;
  bool abDone;
  IsAbelianWordPowerOfSecondArcer arcer;
};
```

———-

20.2.52 class AbelianHomIsEpiComp

— class AbelianHomIsEpiComp —

```
class AbelianHomIsEpiComp : public ComputationManager
{
public:
  AbelianHomIsEpiComp(class MCM& mcm);
  void takeControl( );
  void start( )
  void terminate( )
private:
   class MCM& theMCM;
   AbelianInvariantsARCer arcer;
};
```

———-

20.2.53 class AbelianHomIsEpi

— class AbelianHomIsEpi —

```
class AbelianHomIsEpi  : public Supervisor
{
public:
  AbelianHomIsEpi(class SMMap& map);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
```

```
private:
  SMMap& theMap;
  MirrorSubordinate abelianHomIsEpi;
};
```

———-

20.2.54 class AbelianHomIsMonoComp

— class AbelianHomIsMonoComp —

```
class AbelianHomIsMonoComp : public ComputationManager
{
public:
   AbelianHomIsMonoComp(class MCM& mcm);
   ~AbelianHomIsMonoComp()
  void takeControl( );
  void start( )
  void terminate( )
private:
   class MCM& theMCM;
   AbelianInvariantsOfSGARCer arcer;
   AbelianSGPresentation* abSG;
   bool sgInvStarted;
   bool sgInvFinished;
};
```

———-

20.2.55 class AbelianHomIsMono

— class AbelianHomIsMono —

```
class AbelianHomIsMono  : public Supervisor
{
public:
  AbelianHomIsMono(class SMMap& map);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMMap& theMap;
  MirrorSubordinate abelianInvariants1;
  MirrorSubordinate abelianInvariants2;
  MirrorSubordinate abelianHomIsMono;
  MirrorSubordinate abelianHomIsEpi;
```

```
  bool  abRangeDone;
  bool  abDomainDone;
};
```

———-

20.2.56 class AbelianHomIsAuto

— class AbelianHomIsAuto —

```
class AbelianHomIsAuto  : public Supervisor
{
public:
  AbelianHomIsAuto(class SMHomomorphism& homo);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMHomomorphism& theHomo;
  MirrorSubordinate abelianHomIsEpi;
};
```

———-

20.2.57 class AbelianHomIsIso

— class AbelianHomIsIso —

```
class AbelianHomIsIso  : public Supervisor
{
public:
  AbelianHomIsIso(class SMHomomorphism2& homo);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMHomomorphism2& theHomo;
  bool abRangeDone;
  bool abDomainDone;
  MirrorSubordinate abelianHomIsMono;
  MirrorSubordinate abelianHomIsEpi;
  MirrorSubordinate abelianInvariants1;
  MirrorSubordinate abelianInvariants2;
};
```

———-

20.2.58 class AbelianOrderOfAutoARCer

— class AbelianOrderOfAutoARCer —

```
class AbelianOrderOfAutoARCer : public ARCer
{
public:
  AbelianOrderOfAutoARCer( ComputationManager& boss )
    : ARCer( boss ), AG( 0 ), result(0)
  ~AbelianOrderOfAutoARCer( )
  void setArguments( const AbelianGroup& group, const VectorOf<Word>& image);
  int getOrder() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  AbelianGroup* AG;
  VectorOf<Word> theImages;
  int result;
};
```

———-

20.2.59 class AbelianOrderOfAuto

— class AbelianOrderOfAuto —

```
class AbelianOrderOfAuto  : public Supervisor
{
public:
  AbelianOrderOfAuto(const class SMHomomorphism& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class SMHomomorphism& theAuto;
  MirrorSubordinate abelianInvariants;
  AbelianOrderOfAutoARCer arcer;
  bool started;
};
```

———-

20.2.60 class AbelianInverseAutoARCer

— class AbelianInverseAutoARCer —

```
class AbelianInverseAutoARCer : public ARCer
{
public:
  AbelianInverseAutoARCer( ComputationManager& boss )
    : ARCer( boss ), AG( 0 ), result(0)
  ~AbelianInverseAutoARCer( )
  void setArguments( const AbelianGroup& group, const VectorOf<Word>& image);
  const VectorOf<Word> getInverse() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  AbelianGroup* AG;
  VectorOf<Word> theImages;
  VectorOf<Word> result;
};
```

———-

20.2.61 class AbelianInverseAuto

— class AbelianInverseAuto —

```
class AbelianInverseAuto  : public Supervisor
{
public:
  AbelianInverseAuto(const class SMHomomorphism& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class SMHomomorphism& theAuto;
  AbelianInverseAutoARCer arcer;
};
```

———-

20.2.62 class AbelianFixedPointsOfAutoARCer

— class AbelianFixedPointsOfAutoARCer —

```
class AbelianFixedPointsOfAutoARCer : public ARCer
{
public:
  AbelianFixedPointsOfAutoARCer( ComputationManager& boss )
    : ARCer( boss ),
      A( FPGroup() ),
      result( )
  void setArguments( const AbelianGroup& , const VectorOf<Word>& );
  const VectorOf<Word> getResult( ) const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  AbelianGroup A;
  VectorOf<Word> theImages;
  VectorOf<Word> result;
};
```

———-

20.2.63 class AbelianFixedPointsOfAutoProblem

— class AbelianFixedPointsOfAutoProblem —

```
class AbelianFixedPointsOfAutoProblem  : public Supervisor
{
public:
  AbelianFixedPointsOfAutoProblem(const class SMHomomorphism& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class SMHomomorphism& theAuto;
  AbelianFixedPointsOfAutoARCer arcer;
};
```

———-

20.2.64 class AbelianSGIntersectionARCer

— class AbelianSGIntersectionARCer —

```
class AbelianSGIntersectionARCer : public ARCer
{
public:
  AbelianSGIntersectionARCer( ComputationManager& boss )
```

```
    : ARCer( boss ),
      A( FPGroup() ),
      s1( ),
      s2( ),
      result( ),
      file( )
  void setArguments( const AbelianGroup& g,
         const VectorOf<Word>& v1,
         const VectorOf<Word>& v2);
  VectorOf<Word> getResult() const
  Chars getFileName() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  AbelianGroup A;
  VectorOf<Word> s1;
  VectorOf<Word> s2;
  VectorOf<Word> result;
  File file;
};
```

———-

20.2.65 class AbelianSGIntersectionProblem

— class AbelianSGIntersectionProblem —

```
class AbelianSGIntersectionProblem : public Supervisor
{
public:
  AbelianSGIntersectionProblem(class SMSubgroup& S1, class SMSubgroup& S2);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& subgroup1;
  SMSubgroup& subgroup2;
  bool init;
  MirrorSubordinate abelianInvariants;
  AbelianSGIntersectionARCer arcer;
};
```

———-

20.2.66 class AbelianIntegralHomologyARCer

— class AbelianIntegralHomologyARCer —

```
class AbelianIntegralHomologyARCer : public ARCer
{
public:
  AbelianIntegralHomologyARCer( ComputationManager& boss )
    : ARCer( boss ),
      A( FPGroup() ),
      d( 0 ),
      result( FPGroup() )
  void setArguments( const AbelianGroup& g, int n );
  AbelianGroup getResult() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  AbelianGroup A;
  int d;
  AbelianGroup result;
};
```

———-

20.2.67 class AbelianIntegralHomologyProblem

— class AbelianIntegralHomologyProblem —

```
class AbelianIntegralHomologyProblem : public Supervisor
{
public:
  AbelianIntegralHomologyProblem( class SMFPGroup& G, int n = 2 );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMFPGroup& theGroup;
  bool init;
  int d;
  MirrorSubordinate abelianInvariants;
  AbelianIntegralHomologyARCer arcer;
};
```

———-

20.2.68 class IsAbelianHyperbolic

— class IsAbelianHyperbolic —

```
class IsAbelianHyperbolic : public Supervisor
{
public:
  IsAbelianHyperbolic(SMFPGroup& G);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMFPGroup& theGroup;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.69 class AbelianIsSGFinite

— class AbelianIsSGFinite —

```
class AbelianIsSGFinite : public Supervisor
{
public:
  AbelianIsSGFinite(SMSubgroup& g);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMFPGroup& theGroup;
  SMSubgroup& theSubgroup;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.2.70 class AbelianIsSGFreeAbelian

— class AbelianIsSGFreeAbelian —

```
class AbelianIsSGFreeAbelian : public Supervisor
{
public:
```

```
  AbelianIsSGFreeAbelian(SMSubgroup& g);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  MirrorSubordinate abelianSGInvariants;
};
```

———-

20.2.71 class AbelianIsSGHyperbolic

— class AbelianIsSGHyperbolic —

```
class AbelianIsSGHyperbolic : public Supervisor
{
public:
  AbelianIsSGHyperbolic(SMSubgroup& g);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  MirrorSubordinate abelianSGInvariants;
};
```

———-

20.2.72 class AbelianComputeTorsionSubgroupOfSG

— class AbelianComputeTorsionSubgroupOfSG —

```
class AbelianComputeTorsionSubgroupOfSG : public Supervisor
{
public:
  AbelianComputeTorsionSubgroupOfSG(class SMSubgroup& g);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  MirrorSubordinate abelianSGInvariants;
};
```

———-

20.3 SMApps/include/ACConjecture.h

— ACConjecture.h —

```
#include "Supervisor.h"
#include "ACGA.h"
#include "ACConfig.h"
#include "SMVectorOfWords.h"
```

———-

20.3.1 class ACConjectureARCer

— class ACConjectureARCer —

```
class ACConjectureARCer : public ARCer
{
public:
  ACConjectureARCer( ComputationManager& );
  ~ACConjectureARCer( )
  void setArguments( FPGroup );
  Chars getComputationFileName() const
  Chars getResultFileName() const
  bool getResult() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  FPGroup theGroup;
  File compFile;
  File resultFile;
  bool result;
};
```

———-

20.3.2 class ACConjectureProblem

— class ACConjectureProblem —

```
class ACConjectureProblem : public Supervisor
{
public:
  ACConjectureProblem( const class SMVectorOfWords& );
```

```
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( )
private:
  const SMVectorOfWords& theTuple;
  bool linkHasBeenSent;
  ACConjectureARCer arcer;
};
```

———-

20.4 SMApps/include/ACE.h

— **ACE.h** —

```
#include "Supervisor.h"
#include "SMSubgroup.h"
```

———-

20.4.1 class ACEArcer

— **class ACEArcer** —

```
class ACEArcer : public ARCer
{
public:
  ACEArcer( ComputationManager& );
  ~ACEArcer( )
  void setArguments( FPGroup g, VectorOf<Word> sg );
  void setTime( int t )
  void setWorkspace( int w )
  void setDifficulty( Chars d )
  void setExtra( Chars e )
  Chars getFileName( )
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  FPGroup theGroup;
  VectorOf<Word> theSubgroup;
  File in;
  int iTime, iWS;
  Chars rDiff, cExtra;
};
```

———-

20.4.2 class ACECM

— class ACECM —

```
class ACECM : public ComputationManager
{
public:
  ACECM( class ACEProblem& );
  bool isFinished() const
  void takeControl( );
  ACEArcer& getArcer( )
  void start( );
  void terminate( );
private:
  ACEArcer arcer;
  const SMSubgroup& theSubgroup;
  bool linkHasBeenSent;
  bool bFinished;
};
```

———-

20.4.3 class ACEProblem

— class ACEProblem —

```
class ACEProblem : public Supervisor
{
public:
  ACEProblem(const class SMSubgroup& sg);
  ACEProblem(class SMFPGroup& g);
  const class SMFPGroup& getGroup( ) const
  const class SMSubgroup& getSubgroup( ) const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( )
private:
  const SMFPGroup& theGroup;
  const SMSubgroup& theSubgroup;
  Subordinate<ACEProblem, ACECM> ace;
};
```

———-

20.5 SMApps/include/AGModule.h

— **AGModule.h** —

```
#include "Supervisor.h"
#include "KBmagPackage.h"
#include "List.h"
#include "ARCer.h"
#include "FPGroup.h"
#include "CommutatorIterator.h"
```

———-

20.5.1 class AutGroupARCer

— **class AutGroupARCer** —

```
class AutGroupARCer : public ARCer
{
public:
  AutGroupARCer( ComputationManager& boss )
    : ARCer( boss ), retValue(dontknow)
  void setArguments( const FPGroup );
  Trichotomy isAutomatic()
  DiffMachine getDiffMachine1( );
  DiffMachine getDiffMachine2( );
  GroupDFSA getWordAcceptor( );
  GenMult getGeneralMultiplier( );
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  FPGroup G;
  Trichotomy retValue;
  DiffMachine DM1, DM2;
  GroupDFSA WA;
  GenMult GM;
  KBmagPackage* KBM;
};
```

———-

20.5.2 class AGSupervisor

— **class AGSupervisor** —

```
class AGSupervisor : public ComputationManager
{
public:
  AGSupervisor( class GCM& gcm );
  void viewStructure(ostream& ostr) const
  void takeControl( );
  void start( );
  void terminate( )
private:
  SMFPGroup& theGroup;
  AutGroupARCer arcer;
};
```

———-

20.5.3 class AGProblem

— **class AGProblem** —

```
class AGProblem : public Supervisor
{
public:
  AGProblem(class SMFPGroup& G);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
private:
  SMFPGroup& theGroup;
  MirrorSubordinate agSupervisor;
};
```

———-

20.6 SMApps/include/AreEltsEqual.h

— **AreEltsEqual.h** —

```
#include "Supervisor.h"
#include "SetOfWordsChecker.h"
#include "NilpotentProblems.h"
#include "GeneticProblems.h"
```

———-

20.6.1 class AreEltsEqual

— class AreEltsEqual —

```
class AreEltsEqual : public Supervisor
{
public:
  AreEltsEqual( const class SMWord&, const class SMWord& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const SMWord& word1;
  const SMWord& word2;
  SetOfWordsChecker theChecker;
  MirrorSubordinate normalClosure;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate nilpotentQuotients;
  Subordinate<AreEltsEqual,NilpotentWPInQuotients> nilpotentWPInQuotients;
  MirrorSubordinate computeBasis;
  Subordinate<AreEltsEqual,NilpotentWP> nilpotentWP;
  Subordinate<AreEltsEqual, GeneticWPCM> genetic;
};
```

———-

20.7 SMApps/include/CommutatorIterator.h

— CommutatorIterator.h —

———-

20.7.1 class CommutatorIterator

— class CommutatorIterator —

```
class CommutatorIterator {
public:
  CommutatorIterator( int numberOfGenerators, int length ) :
    theLength( length ), numbers( length ), bDone( false ),
    theWords( numberOfGenerators )
  CommutatorIterator( const VectorOf<Word>& V, int length ) :
```

```
    theLength( length ), numbers( length ), bDone( false ), theWords( V )
  Word value( ) const
  bool next( )
  bool done( ) const
  void reset( )
  VectorOf<int> components()
private:
  Word makeCommutator( )
  bool bDone;
  int theLength;
  Word current;
  VectorOf<int> numbers;
  VectorOf<Word> theWords;
};
```

———-

20.8 SMApps/include/CommutatorsChecker.h

— **CommutatorsChecker.h** —

```
#include "SMFPGroup.h"
#include "ARCer.h"
```

———-

20.8.1 class CommutatorsChecker

— **class CommutatorsChecker** —

```
class CommutatorsChecker
{
public:
  CommutatorsChecker( class SMFPGroup& group, int length );
  CommutatorsChecker( class SMFPGroup& group, int length,
          const VectorOf<Word>& V );
  Trichotomy areTrivial( );
  Chars getExplanation( )
  friend ostream& operator < ( ostream& ostr, const CommutatorsChecker& o )
  friend istream& operator > ( istream& istr, CommutatorsChecker& o )
private:
  FPGroup G;
  class GIC& gic;
  class GCM& gcm;
  int theLength;
  VectorOf<Word> generators;
```

```
  bool triedAbelianization;
  Chars explanation;
};
```

———-

20.8.2 class CommutatorsCheckerARCer

— class CommutatorsCheckerARCer —

```
class CommutatorsCheckerARCer : public ARCer
{
public:
  CommutatorsCheckerARCer( ComputationManager& boss )
    : ARCer( boss ), retValue( dontknow )
  void setArguments( CommutatorsChecker* );
  Trichotomy getRetValue()
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  CommutatorsChecker* theChecker;
  Trichotomy retValue;
};
```

———-

20.9 SMApps/include/ConjugacyProblem.h

— ConjugacyProblem.h —

```
#include "Word.h"
#include "FPGroup.h"
#include "Supervisor.h"
#include "GAConjProblemForORGroup.h"
```

———-

20.9.1 class FPConjugacyARCer

— class FPConjugacyARCer —

```
class FPConjugacyARCer : public ARCer
{
public:
```

```
  FPConjugacyARCer( ComputationManager& boss );
  void setArguments(
        const class SMFPGroup& G,
        const class Word& u,
        const class Word& w
        );
  Trichotomy answer( ) const
  Word conjugator( ) const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  const SMFPGroup *theGroup;
  Word firstWord, secondWord;
  Trichotomy theAnswer;
  Word theConjugator;
};
```

———-

20.9.2 class FPConjugacyWrapper

— class FPConjugacyWrapper —

```
class FPConjugacyWrapper : public ComputationManager
{
public:
  FPConjugacyWrapper(class ConjugacyProblem& CP);
  Trichotomy answer( ) const
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class ConjugacyProblem& problem;
  FPConjugacyARCer arcer;
  Trichotomy theAnswer;
};
```

———-

20.9.3 class MSCConjugacyARCer

— class MSCConjugacyARCer —

```
class MSCConjugacyARCer : public ARCer
{
public:
```

```
  MSCConjugacyARCer( ComputationManager& );
  void setArguments( const FPGroup& G, const Word& u, const Word& v );
  Trichotomy answer( ) const
  Word conjugator( ) const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  FPGroup theGroup;
  Word firstWord, secondWord;
  Trichotomy theAnswer;
  Word theConjugator;
};
```

———-

20.9.4 class MSCConjugacyWrapper

— class MSCConjugacyWrapper —

```
class MSCConjugacyWrapper : public ComputationManager
{
public:
  MSCConjugacyWrapper(class ConjugacyProblem& CP);
  Trichotomy answer( ) const
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class ConjugacyProblem& problem;
  MSCConjugacyARCer arcer;
  Trichotomy theAnswer;
};
```

———-

20.9.5 class ConjugacyProblem

— class ConjugacyProblem —

```
class ConjugacyProblem : public Supervisor
{
public:
  ConjugacyProblem(const class SMWord& w1, const class SMWord& w2);
  const class SMWord& firstWord( ) const
  const class SMWord& secondWord( ) const
  void viewStructure(ostream& ostr) const;
```

```
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class SMWord& u;
  const class SMWord& v;
  bool scMethodFailed;
  bool geneticLinkHasBeenSent;
  Subordinate<ConjugacyProblem,FPConjugacyWrapper> fpConjugacyWrapper;
  Subordinate<ConjugacyProblem,MSCConjugacyWrapper> mscConjugacyWrapper;
  Subordinate<ConjugacyProblem,GAConjugacyForORGroup> GAConjugacy;
};
```

———-

20.10 SMApps/include/EquationsInFPProblem.h

— EquationsInFPProblem.h —

```
#include "SMEqSystem.h"
#include "AbelianEquations.h"
```

———-

20.10.1 class EqSystemInAbelianARCer

— class EqSystemInAbelianARCer —

```
class EqSystemInAbelianARCer : public ARCer
{
public:
  EqSystemInAbelianARCer( ComputationManager& boss )
    : ARCer( boss ),
      retValue( 0 ),
      file()
  void setArguments( const AbelianEquationsSolver& );
  ~EqSystemInAbelianARCer()
  Chars getFileName() const
  bool getRetValue() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  File file;
  bool retValue;
  AbelianEquationsSolver AES;
```

```
};
```

———-

20.10.2 class EqSystemInAbelianCM

— class EqSystemInAbelianCM —

```
class EqSystemInAbelianCM : public ComputationManager
{
public:
  EqSystemInAbelianCM( class EqSystemInFPProblem& p );
  Chars getFileName() const
  Trichotomy haveSolutions() const
  void viewStructure(ostream& ostr) const
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMEqSystem& theSystem;
  EqSystemInAbelianARCer arcer;
  bool answer;
};
```

———-

20.10.3 class EqSystemInFPProblem

— class EqSystemInFPProblem —

```
class EqSystemInFPProblem : public Supervisor
{
public:
  EqSystemInFPProblem( SMEqSystem& s );
  SMEqSystem& getSystem() const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
private:
  SMEqSystem& theSystem;
  Subordinate<EqSystemInFPProblem , EqSystemInAbelianCM> abelianCM;
};
```

———-

20.10.4 class EquationInAbelianCM

— class EquationInAbelianCM —

```
class EquationInAbelianCM : public ComputationManager
{
public:
  EquationInAbelianCM( class EquationInFPProblem& p );
  Chars getFileName() const
  Trichotomy haveSolutions() const
  void viewStructure(ostream& ostr) const
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMEquation2& theEquation;
  EqSystemInAbelianARCer arcer;
  bool answer;
};
```

———-

20.10.5 class EquationInFPProblem

— class EquationInFPProblem —

```
class EquationInFPProblem : public Supervisor
{
public:
  EquationInFPProblem( SMEquation2& s );
  SMEquation2& getEquation() const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
private:
  SMEquation2& theEquation;
  Subordinate<EquationInFPProblem , EquationInAbelianCM> abelianCM;
};
```

———-

20.11 SMApps/include/ExtendToHomProblem.h

— ExtendToHomProblem.h —

```
#include "Supervisor.h"
#include "FPGroup.h"
```

```
#include "SMMap.h"
#include "CommutatorIterator.h"
#include "NilpotentProblems.h"
#include "GeneticProblems.h"
```

———-

20.11.1 class ExtendToHomChecker

— class ExtendToHomChecker —

```
class ExtendToHomChecker
{
public:
  ExtendToHomChecker(class ExtendToHomProblem& boss, class SMMap& );
  Trichotomy doesExtendToHom( );
  Chars getExplanation( ) const
  Trichotomy preliminaryCheckings();
  bool checkRelatorsInNilpotent() const;
private:
  class ExtendToHomProblem& theBoss;
  class SMMap& theMap;
  const Map M;
  class SMFPGroup& theDomain;
  class SMFPGroup& theRange;
  FPGroup G1;
  FPGroup G2;
  SetOf<Word> relators;
  class GIC& gic1;
  class GIC& gic2;
  class GCM& gcm1;
  class GCM& gcm2;
  int theClass;
  bool triedAbelianization;
  Chars explanation;
  SetOf<Word> getAllRelators( class SMFPGroup& ) const;
};
```

———-

20.11.2 class ExtendToHomProblem

— class ExtendToHomProblem —

```
class ExtendToHomProblem : public Supervisor
{
public:
```

```
  ExtendToHomProblem( class SMMap& );
  Trichotomy answer( ) const
  Trichotomy nilpWPAnswer( ) ;
  Trichotomy nilpWPInQuotAnswer( ) ;
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  void sendResultMessage( Chars );
  SMMap& theMap;
  const SMFPGroup& theDomain;
  const SMFPGroup& theRange;
  int theClass;
  ExtendToHomChecker theChecker;
  Trichotomy theAnswer;
  Chars explanation;
  MirrorSubordinate normalClosure;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate nilpotentQuotients;
  Subordinate<ExtendToHomProblem,NilpotentWPInQuotients>
    nilpotentWPInQuotients;
  MirrorSubordinate computeBasis;
  Subordinate<ExtendToHomProblem,NilpotentWP> nilpotentWP;
  Subordinate<ExtendToHomProblem, GeneticWPCM> genetic;
};
```

———-

20.12 SMApps/include/fastProblems.h

— **fastProblems.h** —

```
#include "ComputationManager.h"
#include "Word.h"
#include "SMVectorOfWords.h"
#include "SMEnumerator.h"
```

———-

20.12.1 class FastComputation

— **class FastComputation** —

```
class FastComputation : public ComputationManager
{
public:
  FastComputation( );
  void readMessage(class istream &)
  void viewStructure(ostream& ostr) const;
  void start( )
  void terminate( )
};
```

———-

20.12.2 class CommutatorInFree

— class CommutatorInFree —

```
class CommutatorInFree : public FastComputation
{
public:
  CommutatorInFree(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.3 class FreeInCommutatorSG

— class FreeInCommutatorSG —

```
class FreeInCommutatorSG : public FastComputation
{
public:
  FreeInCommutatorSG(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.4 class ProductOfCommutators

— class ProductOfCommutators —

```
class ProductOfCommutators : public FastComputation
{
public:
  ProductOfCommutators(const class SMWord& w) : theWord( w ), theFile( )
  void takeControl( );
private:
  const class SMWord& theWord;
  File theFile;
};
```

———-

20.12.5 class ProductOfSquares

— class ProductOfSquares —

```
class ProductOfSquares : public FastComputation
{
public:
  ProductOfSquares(const class SMWord& w) : theWord( w ), theFile( )
  void takeControl( );
private:
  const class SMWord& theWord;
  File theFile;
};
```

———-

20.12.6 class FreeIsElementAProperPower

— class FreeIsElementAProperPower —

```
class FreeIsElementAProperPower : public FastComputation
{
public:
  FreeIsElementAProperPower(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.7 class FreeMaximalRootOfElement

— class FreeMaximalRootOfElement —

```
class FreeMaximalRootOfElement : public FastComputation
{
public:
  FreeMaximalRootOfElement(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.8 class FreeCentolizerOfElement

— class FreeCentolizerOfElement —

```
class FreeCentolizerOfElement : public FastComputation
{
public:
  FreeCentolizerOfElement(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.9 class FreeGetN

— class FreeGetN —

```
class FreeGetN_thElement : public FastComputation
{
public:
  FreeGetN_thElement( class SMFPGroup& G, int n = 0 ) :
    theGroup( G ), theNumber( n )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
  int theNumber;
};
```

———-

20.12.10 class FreeGetNextN

— class FreeGetNextN —

```
class FreeGetNextN_thElement : public FastComputation
{
public:
  FreeGetNextN_thElement( class SMWord& w, int n = 0 ) :
    theWord( w ), theNumber( n )
  void takeControl( );
private:
  class SMWord& theWord;
  int theNumber;
};
```

———-

20.12.11 class WordProblemInFree

— class WordProblemInFree —

```
class WordProblemInFree : public FastComputation
{
public:
  WordProblemInFree(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.12 class WordsAreEqual

— class WordsAreEqual —

```
class WordsAreEqual : public FastComputation
{
public:
  WordsAreEqual(const class SMWord& w1, const class SMWord& w2)
    : word1( w1 ), word2( w2 )
  void takeControl( );
private:
  const class SMWord& word1;
  const class SMWord& word2;
};
```

———-

20.12.13 class EndoOnFreeIsMono

— class EndoOnFreeIsMono —

```
class EndoOnFreeIsMono : public FastComputation
{
public:
  EndoOnFreeIsMono( class SMMap& m ) : map( m )
  void takeControl( );
private:
  class SMMap& map;
};
```

———-

20.12.14 class EndoOnFreeIsEpi

— class EndoOnFreeIsEpi —

```
class EndoOnFreeIsEpi : public FastComputation
{
public:
  EndoOnFreeIsEpi( class SMMap& m ) : map( m )
  void takeControl( );
private:
  class SMMap& map;
};
```

———-

20.12.15 class EndoOnFreeIsAut

— class EndoOnFreeIsAut —

```
class EndoOnFreeIsAut : public FastComputation
{
public:
  EndoOnFreeIsAut(const class SMMap& m) : map( m )
  void takeControl( );
private:
  const class SMMap& map;
};
```

———-

20.12.16 class EndoOnFreeIsInner

— class EndoOnFreeIsInner —

```
class EndoOnFreeIsInner : public FastComputation
{
public:
  EndoOnFreeIsInner(const class SMMap& m) : map( m )
  void takeControl( );
private:
  const class SMMap& map;
};
```

———-

20.12.17 class EndoOnFreeIsIAAut

— class EndoOnFreeIsIAAut —

```
class EndoOnFreeIsIAAut : public FastComputation
{
public:
  EndoOnFreeIsIAAut(const class SMMap& m) : map( m )
  void takeControl( );
private:
  const class SMMap& map;
};
```

———-

20.12.18 class InverseAuto

— class InverseAuto —

```
class InverseAuto : public FastComputation
{
public:
  InverseAuto(const class SMMap& m) : map( m )
  void takeControl( );
private:
  const class SMMap& map;
};
```

———-

20.12.19 class AutoWhiteheadDecomposition

— class AutoWhiteheadDecomposition —

```
class AutoWhiteheadDecomposition : public FastComputation
{
public:
  AutoWhiteheadDecomposition(const class SMMap& m) : map( m )
  void takeControl( );
private:
  const class SMMap& map;
};
```

———-

20.12.20 class WordInSGOfFree

— class WordInSGOfFree —

```
class WordInSGOfFree : public FastComputation
{
public:
  WordInSGOfFree(const class SMSubgroup& S, const class SMWord& w)
    : word( w ), subgroup( S )
  void takeControl( );
private:
  const class SMWord& word;
  const class SMSubgroup& subgroup;
};
```

———-

20.12.21 class PowerOfWordInSGOfFree

— class PowerOfWordInSGOfFree —

```
class PowerOfWordInSGOfFree : public FastComputation
{
public:
  PowerOfWordInSGOfFree(const class SMSubgroup& S, const class SMWord& w)
    : word( w ), subgroup( S )
  void takeControl( );
private:
  const class SMWord& word;
  const class SMSubgroup& subgroup;
};
```

———-

20.12.22 class ConjugacyProblemInFree

— class ConjugacyProblemInFree —

```
class ConjugacyProblemInFree : public FastComputation
{
public:
  ConjugacyProblemInFree(const class SMWord& w1, const class SMWord& w2)
    : word1( w1 ), word2( w2 )
  void takeControl( );
private:
  const class SMWord& word1;
  const class SMWord& word2;
};
```

———-

20.12.23 class ConjugateOfWordInSGOfFree

— class ConjugateOfWordInSGOfFree —

```
class ConjugateOfWordInSGOfFree : public FastComputation
{
public:
  ConjugateOfWordInSGOfFree(const class SMSubgroup& S, const class SMWord& w)
    : word( w ), subgroup( S )
  void takeControl( );
private:
  const class SMWord& word;
  const class SMSubgroup& subgroup;
};
```

———-

20.12.24 class WordInNielsenBasisSGOfFree

— class WordInNielsenBasisSGOfFree —

```
class WordInNielsenBasisSGOfFree : public FastComputation
{
public:
  WordInNielsenBasisSGOfFree(const class SMSubgroup& S, const class SMWord& w)
    : word( w ), subgroup( S )
  void takeControl( );
```

```
private:
  const class SMWord& word;
  const class SMSubgroup& subgroup;
};
```

———-

20.12.25 class SchreierRepOfWordInSGOfFree

— class SchreierRepOfWordInSGOfFree —

```
class SchreierRepOfWordInSGOfFree : public FastComputation
{
public:
  SchreierRepOfWordInSGOfFree(const class SMSubgroup& S, const class SMWord& w)
    : word( w ), subgroup( S )
  void takeControl( );
private:
  const class SMWord& word;
  const class SMSubgroup& subgroup;
};
```

———-

20.12.26 class SGOfFreeContainment

— class SGOfFreeContainment —

```
class SGOfFreeContainment : public FastComputation
{
public:
  SGOfFreeContainment(const class SMSubgroup& S1, const class SMSubgroup& S2)
    : subgroup1( S1 ), subgroup2( S2 )
  void takeControl( );
private:
  const class SMSubgroup& subgroup1;
  const class SMSubgroup& subgroup2;
};
```

———-

20.12.27 class SGOfFreeAreEqual

— class SGOfFreeAreEqual —

```
class SGOfFreeAreEqual : public FastComputation
{
public:
  SGOfFreeAreEqual(const class SMSubgroup& S1, const class SMSubgroup& S2)
    : subgroup1( S1 ), subgroup2( S2 )
  void takeControl( );
private:
  const class SMSubgroup& subgroup1;
  const class SMSubgroup& subgroup2;
};
```

———-

20.12.28 class SGOfFreeJoin

— class SGOfFreeJoin —

```
class SGOfFreeJoin : public FastComputation
{
public:
  SGOfFreeJoin(const class SMSubgroup& S1, const class SMSubgroup& S2)
    : subgroup1( S1 ), subgroup2( S2 )
  void takeControl( );
private:
  const class SMSubgroup& subgroup1;
  const class SMSubgroup& subgroup2;
};
```

———-

20.12.29 class SGOfFreeIntersection

— class SGOfFreeIntersection —

```
class SGOfFreeIntersection : public FastComputation
{
public:
  SGOfFreeIntersection(const class SMSubgroup& S1, const class SMSubgroup& S2)
    : subgroup1( S1 ), subgroup2( S2 )
  void takeControl( );
private:
  const class SMSubgroup& subgroup1;
  const class SMSubgroup& subgroup2;
};
```

———-

20.12.30 class SGOfFreeIsNormal

— class SGOfFreeIsNormal —

```
class SGOfFreeIsNormal : public FastComputation
{
public:
  SGOfFreeIsNormal( class SMSubgroup& S) : subgroup( S )
  void takeControl( );
private:
  class SMSubgroup& subgroup;
};
```

———-

20.12.31 class SGOfFreeIsAFreeFactor

— class SGOfFreeIsAFreeFactor —

```
class SGOfFreeIsAFreeFactor : public FastComputation
{
public:
  SGOfFreeIsAFreeFactor(const class SMSubgroup& S) : subgroup( S )
  void takeControl( );
private:
  const class SMSubgroup& subgroup;
};
```

———-

20.12.32 class SGOfFreeIsMalnormal

— class SGOfFreeIsMalnormal —

```
class SGOfFreeIsMalnormal : public FastComputation
{
public:
  SGOfFreeIsMalnormal(const class SMSubgroup& S) : subgroup( S )
  void takeControl( );
private:
  const class SMSubgroup& subgroup;
};
```

———-

20.12.33 class QuadEquationSurfaceForm

— class QuadEquationSurfaceForm —

```
class QuadEquationSurfaceForm : public FastComputation
{
public:
  QuadEquationSurfaceForm(const class SMEquation& S) : equation( S )
  void takeControl( );
private:
  const class SMEquation& equation;
};
```

———-

20.12.34 class SGOfFreeWhiteheadReduction

— class SGOfFreeWhiteheadReduction —

```
class SGOfFreeWhiteheadReduction : public FastComputation
{
public:
  SGOfFreeWhiteheadReduction(const class SMVectorOfWords& S) : vect( S )
  void takeControl( );
private:
  const class SMVectorOfWords& vect;
};
```

———-

20.12.35 class SGOfFreeNielsenBasis

— class SGOfFreeNielsenBasis —

```
class SGOfFreeNielsenBasis : public FastComputation
{
public:
  SGOfFreeNielsenBasis(const class SMSubgroup& S) : subgroup( S )
  void takeControl( );
private:
  const class SMSubgroup& subgroup;
};
```

———-

20.12.36 class SGOfFreeIndex

— class SGOfFreeIndex —

```
class SGOfFreeIndex : public FastComputation
{
public:
  SGOfFreeIndex(const class SMSubgroup& S) : subgroup( S )
  void takeControl( );
private:
  const class SMSubgroup& subgroup;
};
```

———-

20.12.37 class SGOfFreeRank

— class SGOfFreeRank —

```
class SGOfFreeRank : public FastComputation
{
public:
  SGOfFreeRank(const class SMSubgroup& S) : subgroup( S )
  void takeControl( );
private:
  const class SMSubgroup& subgroup;
};
```

———-

20.12.38 class SGOfFreeNormaliser

— class SGOfFreeNormaliser —

```
class SGOfFreeNormaliser : public FastComputation
{
public:
  SGOfFreeNormaliser(const class SMSubgroup& S) : subgroup( S )
  void takeControl( );
private:
  const class SMSubgroup& subgroup;
};
```

———-

20.12.39 class SGOfFreeHallCompletion

— class SGOfFreeHallCompletion —

```
class SGOfFreeHallCompletion : public FastComputation
{
public:
  SGOfFreeHallCompletion(const class SMSubgroup& S) : subgroup( S )
  void takeControl( );
private:
  const class SMSubgroup& subgroup;
};
```

———-

20.12.40 class NormalApproximationProblem

— class NormalApproximationProblem —

```
class NormalApproximationProblem : public FastComputation
{
public:
  NormalApproximationProblem(const class SMSubgroup& S, int l = 1 )
    : subgroup( S ), level( l )
  void takeControl( );
private:
  const class SMSubgroup& subgroup;
  int level;
};
```

———-

20.12.41 class FreeIsSGTrivial

— class FreeIsSGTrivial —

```
class FreeIsSGTrivial : public FastComputation
{
public:
  FreeIsSGTrivial(const class SMSubgroup& S) : subgroup( S )
  void takeControl( );
private:
  const class SMSubgroup& subgroup;
};
```

———-

20.12.42 class FreeIsAutomatic

— class FreeIsAutomatic —

```
class FreeIsAutomatic : public FastComputation
{
public:
  FreeIsAutomatic(const class SMFPGroup& G) : group( G )
  void takeControl( );
private:
  const class SMFPGroup& group;
};
```

———-

20.12.43 class FreeIsHyperbolic

— class FreeIsHyperbolic —

```
class FreeIsHyperbolic : public FastComputation
{
public:
  FreeIsHyperbolic(class SMFPGroup& G) : group( G )
  void takeControl( );
private:
  const class SMFPGroup& group;
};
```

———-

20.12.44 class FreelyReduceWord

— class FreelyReduceWord —

```
class FreelyReduceWord : public FastComputation
{
public:
  FreelyReduceWord(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.45 class CyclicallyReduceWord

— class CyclicallyReduceWord —

```
class CyclicallyReduceWord : public FastComputation
{
public:
  CyclicallyReduceWord(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.46 class FormalInverseOfWord

— class FormalInverseOfWord —

```
class FormalInverseOfWord : public FastComputation
{
public:
  FormalInverseOfWord(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.47 class WordLength

— class WordLength —

```
class WordLength : public FastComputation
{
public:
  WordLength( const class SMWord& w ) : theWord ( w )
  void takeControl( );
private:
  const SMWord& theWord;
};
```

———-

20.12.48 class InitialSegmentOfWord

— class InitialSegmentOfWord —

```
class InitialSegmentOfWord : public FastComputation
{
public:
  InitialSegmentOfWord(const class SMWord& w, int length = 1) :
    theWord( w ), theLength( length )
  void takeControl( );
private:
  const class SMWord& theWord;
  int theLength;
};
```

———-

20.12.49 class TerminalSegmentOfWord

— class TerminalSegmentOfWord —

```
class TerminalSegmentOfWord : public FastComputation
{
public:
  TerminalSegmentOfWord(const class SMWord& w, int length = 1) :
    theWord( w ), theLength( length )
  void takeControl( );
private:
  const class SMWord& theWord;
  int theLength;
};
```

———-

20.12.50 class SegmentOfWord

— class SegmentOfWord —

```
class SegmentOfWord : public FastComputation
{
public:
  SegmentOfWord(const class SMWord& w, int start = 1, int length = 1) :
    theWord( w ), theStart( start ), theLength( length )
  void takeControl( );
private:
  const class SMWord& theWord;
```

```
  int theStart;
  int theLength;
};
```

———-

20.12.51 class FormalProductOfWords

— class FormalProductOfWords —

```
class FormalProductOfWords : public FastComputation
{
public:
  FormalProductOfWords(const class SMWord& w1, const class SMWord& w2)
    : word1( w1 ), word2( w2 )
  void takeControl( );
private:
  const class SMWord& word1;
  const class SMWord& word2;
};
```

———-

20.12.52 class ConjugateOfWord

— class ConjugateOfWord —

```
class ConjugateOfWord : public FastComputation
{
public:
  ConjugateOfWord(const class SMWord& w1, const class SMWord& w2)
    : theWord1( w1 ), theWord2( w2 )
  void takeControl( );
private:
  const class SMWord& theWord1;
  const class SMWord& theWord2;
};
```

———-

20.12.53 class CommutatorOfWords

— class CommutatorOfWords —

```
class CommutatorOfWords : public FastComputation
{
public:
  CommutatorOfWords(const class SMWord& w1, const class SMWord& w2)
    : theWord1( w1 ), theWord2( w2 )
  void takeControl( );
private:
  const class SMWord& theWord1;
  const class SMWord& theWord2;
};
```

———-

20.12.54 class PowerOfMap

— class PowerOfMap —

```
class PowerOfMap : public FastComputation
{
public:
  PowerOfMap(const class SMMap& m, int p) : map( m ), power( p )
  PowerOfMap(const class SMMap& m) : map( m )
  void takeControl( );
private:
  const class SMMap& map;
  int power;
};
```

———-

20.12.55 class ComposeMaps

— class ComposeMaps —

```
class ComposeMaps : public FastComputation
{
public:
  ComposeMaps(const class SMMap& m1, const class SMMap& m2)
    : map1( m1 ), map2( m2 )
  void takeControl( );
private:
  const class SMMap& map1;
  const class SMMap& map2;
};
```

———-

20.12.56 class FreeAreHomsEqual

— class FreeAreHomsEqual —

```
class FreeAreHomsEqual : public FastComputation
{
public:
  FreeAreHomsEqual(const class SMMap& m1, const class SMMap& m2)
    : map1( m1 ), map2( m2 )
  void takeControl( );
private:
  const class SMMap& map1;
  const class SMMap& map2;
};
```

———-

20.12.57 class ImageUnderMap

— class ImageUnderMap —

```
class ImageUnderMap : public FastComputation
{
public:
  ImageUnderMap(const class SMMap& m, const class SMWord& w)
    : map( m ), word( w )
  void takeControl( );
private:
  const class SMMap& map;
  const class SMWord& word;
};
```

———-

20.12.58 class SGImageUnderMap

— class SGImageUnderMap —

```
class SGImageUnderMap : public FastComputation
{
public:
  SGImageUnderMap(const class SMMap& m, const class SMSubgroup& S)
    : map( m ), subgroup( S )
  void takeControl( );
private:
  const class SMMap& map;
```

```
  const class SMSubgroup& subgroup;
};
```

———-

20.12.59 class ExtendFreeByAut

— class ExtendFreeByAut —

```
class ExtendFreeByAut : public FastComputation
{
public:
  ExtendFreeByAut(const class SMMap& m);
  void takeControl( );
private:
  const class SMFPGroup& theGroup;
  const class SMMap& theMap;
};
```

———-

20.12.60 class FPIsMSC

— class FPIsMSC —

```
class FPIsMSC : public FastComputation
{
public:
  FPIsMSC(class SMFPGroup& G) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.61 class FastHomology

— class FastHomology —

```
class FastHomology : public FastComputation
{
public:
  FastHomology(class SMFPGroup& G) : theGroup( G )
  void takeControl( );
```

```
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.62 class SubgroupJoin

— class SubgroupJoin —

```
class SubgroupJoin : public FastComputation
{
public:
  SubgroupJoin(const class SMSubgroup& S1, const class SMSubgroup& S2)
    : subgroup1( S1 ), subgroup2( S2 )
  void takeControl( );
private:
  const class SMSubgroup& subgroup1;
  const class SMSubgroup& subgroup2;
};
```

———-

20.12.63 class SubgroupConjugateBy

— class SubgroupConjugateBy —

```
class SubgroupConjugateBy : public FastComputation
{
public:
  SubgroupConjugateBy(const class SMSubgroup& S, const class SMWord& W)
    : theSubgroup( S ), theWord( W )
  void takeControl( );
private:
  const class SMSubgroup& theSubgroup;
  const class SMWord& theWord;
};
```

———-

20.12.64 class FastAbelianForm

— class FastAbelianForm —

```
class FastAbelianForm : public FastComputation
{
public:
  FastAbelianForm(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.65 class FastInverseInAbelianForm

— class FastInverseInAbelianForm —

```
class FastInverseInAbelianForm : public FastComputation
{
public:
  FastInverseInAbelianForm(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.66 class ProductInAbelianForm

— class ProductInAbelianForm —

```
class ProductInAbelianForm : public FastComputation
{
public:
  ProductInAbelianForm(const class SMWord& w1, const class SMWord& w2)
    : word1( w1 ), word2( w2 )
  void takeControl( );
private:
  const class SMWord& word1;
  const class SMWord& word2;
};
```

———-

20.12.67 class AbelianSGJoin

— class AbelianSGJoin —

```
class AbelianSGJoin : public FastComputation
{
public:
  AbelianSGJoin(const class SMSubgroup& S1, const class SMSubgroup& S2)
    : subgroup1( S1 ), subgroup2( S2 )
  void takeControl( );
private:
  const class SMSubgroup& subgroup1;
  const class SMSubgroup& subgroup2;
};
```

———-

20.12.68 class AbelianIsAutomatic

— class AbelianIsAutomatic —

```
class AbelianIsAutomatic : public FastComputation
{
public:
  AbelianIsAutomatic(class SMFPGroup& G)
  AbelianIsAutomatic(class SMSubgroup& G)
  void takeControl( );
};
```

———-

20.12.69 class AbelianIsConfluent

— class AbelianIsConfluent —

```
class AbelianIsConfluent : public FastComputation
{
public:
  AbelianIsConfluent(class SMFPGroup& G)
  AbelianIsConfluent(class SMSubgroup& G)
  void takeControl( );
};
```

———-

20.12.70 class MSCOrder

— class MSCOrder —

```
class MSCOrder : public FastComputation
{
public:
  MSCOrder(class SMFPGroup& G) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.71 class MSCIsTrivial

— class MSCIsTrivial —

```
class MSCIsTrivial : public FastComputation
{
public:
  MSCIsTrivial(class SMFPGroup& G) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.72 class MSCIsFinite

— class MSCIsFinite —

```
class MSCIsFinite : public FastComputation
{
public:
  MSCIsFinite(class SMFPGroup& G) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.73 class MSCIsAbelian

— class MSCIsAbelian —

```
class MSCIsAbelian : public FastComputation
{
public:
  MSCIsAbelian(class SMFPGroup& G) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.74 class ORIsTrivial

— class ORIsTrivial —

```
class ORIsTrivial : public FastComputation
{
public:
  ORIsTrivial(class SMFPGroup& G) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.75 class ORIsFinite

— class ORIsFinite —

```
class ORIsFinite : public FastComputation
{
public:
  ORIsFinite(class SMFPGroup& G) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.76 class ORIsAbelian

— class ORIsAbelian —

```
class ORIsAbelian : public FastComputation
{
public:
  ORIsAbelian(class SMFPGroup& G) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.77 class OROrder

— class OROrder —

```
class OROrder : public FastComputation
{
public:
  OROrder(class SMFPGroup& G) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.78 class ORWithTorsionEltFiniteOrder

— class ORWithTorsionEltFiniteOrder —

```
class ORWithTorsionEltFiniteOrder : public FastComputation
{
public:
  ORWithTorsionEltFiniteOrder(class SMWord& w) : theWord( w )
  void takeControl( );
private:
  class SMWord& theWord;
};
```

———-

20.12.79 class ORWithTorsionAreEltsEqual

— class ORWithTorsionAreEltsEqual —

```
class ORWithTorsionAreEltsEqual : public FastComputation
{
public:
  ORWithTorsionAreEltsEqual(class SMWord& w1, class SMWord& w2)
    : theWord1( w1 ), theWord2( w2 )
  void takeControl( );
private:
  class SMWord& theWord1;
  class SMWord& theWord2;
};
```

———-

20.12.80 class ORWithTorsionExtendedWordProblem

— class ORWithTorsionExtendedWordProblem —

```
class ORWithTorsionExtendedWordProblem : public FastComputation
{
public:
  ORWithTorsionExtendedWordProblem(class SMSubgroup& S, class SMWord& w)
    : theSubgroup( S ),  theWord( w )
  void takeControl( );
private:
  class SMSubgroup& theSubgroup;
  class SMWord& theWord;
};
```

———-

20.12.81 class ORWithTorsionCentralizerOfElt

— class ORWithTorsionCentralizerOfElt —

```
class ORWithTorsionCentralizerOfElt : public FastComputation
{
public:
  ORWithTorsionCentralizerOfElt(class SMWord& w) : theWord( w )
  void takeControl( );
private:
  class SMWord& theWord;
};
```

———-

20.12.82 class ORWithTorsionConjugacyProblem

— class ORWithTorsionConjugacyProblem —

```
class ORWithTorsionConjugacyProblem : public FastComputation
{
public:
  ORWithTorsionConjugacyProblem(class SMWord& w1, class SMWord& w2)
    : word1( w1 ), word2( w2 )
  void takeControl( );
private:
  class SMWord& word1;
  class SMWord& word2;
};
```

———-

20.12.83 class MakeCyclicDecomposition

— class MakeCyclicDecomposition —

```
class MakeCyclicDecomposition : public FastComputation
{
public:
  MakeCyclicDecomposition(class SMFPGroup& G ) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.84 class MakeAbelianQuotient

— class MakeAbelianQuotient —

```
class MakeAbelianQuotient : public FastComputation
{
public:
  MakeAbelianQuotient(class SMFPGroup& G ) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.85 class MakeQuotientFromSubgroup

— class MakeQuotientFromSubgroup —

```
class MakeQuotientFromSubgroup : public FastComputation
{
public:
  MakeQuotientFromSubgroup(class SMSubgroup& S ) : theSubgroup( S )
  void takeControl( );
private:
  class SMSubgroup& theSubgroup;
};
```

———-

20.12.86 class MakeNilpotentQuotient

— class MakeNilpotentQuotient —

```
class MakeNilpotentQuotient : public FastComputation
{
public:
  MakeNilpotentQuotient( class SMFPGroup& G, int n = 1 ) :
    theGroup( G ), theClass( n )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
  int theClass;
};
```

———-

20.12.87 class MakeQuotient

— class MakeQuotient —

```
class MakeQuotient : public FastComputation
{
public:
  MakeQuotient( class SMFPGroup& G, const SetOf<Word>& S = SetOf<Word>() ) :
    theGroup( G ), relators( S )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
```

```
  SetOf<Word> relators;
};
```

———-

20.12.88 class MakeAPOfFree

— class MakeAPOfFree —

```
class MakeAPOfFree : public FastComputation
{
public:
  MakeAPOfFree( const SMFPGroup& G1, const SMFPGroup& G2,
          const SMSubgroup& S1, const SMSubgroup& S2 )
  : subgroup1( S1 ),
    subgroup2( S2 )
  void takeControl( );
private:
  const SMSubgroup& subgroup1;
  const SMSubgroup& subgroup2;
};
```

———-

20.12.89 class APOfFreeReducedForm

— class APOfFreeReducedForm —

```
class APOfFreeReducedForm : public FastComputation
{
public:
  APOfFreeReducedForm(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.90 class APOfFreeNormalForm

— class APOfFreeNormalForm —

```
class APOfFreeNormalForm : public FastComputation
{
public:
  APOfFreeNormalForm(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.91 class APOfFreeCyclicNormalForm

— class APOfFreeCyclicNormalForm —

```
class APOfFreeCyclicNormalForm : public FastComputation
{
public:
  APOfFreeCyclicNormalForm(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.92 class APOfFreeIsTrivial

— class APOfFreeIsTrivial —

```
class APOfFreeIsTrivial : public FastComputation
{
public:
  APOfFreeIsTrivial(class SMFPGroup& G) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.93 class APOfFreeIsHyperbolic

— class APOfFreeIsHyperbolic —

```
class APOfFreeIsHyperbolic : public FastComputation
{
public:
  APOfFreeIsHyperbolic(class SMFPGroup& G) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.94 class APOfFreeIsFinite

— class APOfFreeIsFinite —

```
class APOfFreeIsFinite : public FastComputation
{
public:
  APOfFreeIsFinite(class SMFPGroup& G) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.95 class APOfFreeIsAbelian

— class APOfFreeIsAbelian —

```
class APOfFreeIsAbelian : public FastComputation
{
public:
  APOfFreeIsAbelian(class SMFPGroup& G) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.96 class APOfFreeOrder

— class APOfFreeOrder —

```
class APOfFreeOrder : public FastComputation
{
public:
  APOfFreeOrder(class SMFPGroup& G) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.12.97 class APOfFreeWordProblem

— class APOfFreeWordProblem —

```
class APOfFreeWordProblem : public FastComputation
{
public:
  APOfFreeWordProblem(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.98 class APOfFreeNumberOfSubstitutions

— class APOfFreeNumberOfSubstitutions —

```
class APOfFreeNumberOfSubstitutions : public FastComputation
{
public:
  APOfFreeNumberOfSubstitutions(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.99 class APOfFreeAreEqual

— class APOfFreeAreEqual —

```
class APOfFreeAreEqual : public FastComputation
{
public:
  APOfFreeAreEqual(const class SMWord& w1, const class SMWord& w2)
    : word1( w1 ), word2( w2 )
  void takeControl( );
private:
  const class SMWord& word1;
  const class SMWord& word2;
};
```

———-

20.12.100 class APOfFree

— class APOfFree —

```
class APOfFree_DoEltsCommute : public FastComputation
{
public:
  APOfFree_DoEltsCommute(const class SMWord& w1, const class SMWord& w2)
    : word1( w1 ), word2( w2 )
  void takeControl( );
private:
  const class SMWord& word1;
  const class SMWord& word2;
};
```

———-

20.12.101 class APOfFreeIsSGTrivial

— class APOfFreeIsSGTrivial —

```
class APOfFreeIsSGTrivial : public FastComputation
{
public:
  APOfFreeIsSGTrivial(const class SMSubgroup& S) : subgroup( S )
  void takeControl( );
private:
  const class SMSubgroup& subgroup;
};
```

———-

20.12.102 class CheckinAPOfFree

— class CheckinAPOfFree —

```
class CheckinAPOfFree : public FastComputation
{
public:
  CheckinAPOfFree( )
  void takeControl( );
};
```

———-

20.12.103 class APOfFreeIsSGAbelian

— class APOfFreeIsSGAbelian —

```
class APOfFreeIsSGAbelian : public FastComputation
{
public:
  APOfFreeIsSGAbelian(const class SMSubgroup& S) : theSubgroup( S )
  void takeControl( );
private:
  const class SMSubgroup& theSubgroup;
};
```

———-

20.12.104 class APOfFreeCyclic

— class APOfFreeCyclic —

```
class APOfFreeCyclic_CentralizerOfElt : public FastComputation
{
public:
  APOfFreeCyclic_CentralizerOfElt(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.105 class APOfFreeCyclic

— class APOfFreeCyclic —

```
class APOfFreeCyclic_ConjugacyProblem : public FastComputation
{
public:
  APOfFreeCyclic_ConjugacyProblem( const class SMWord& w1,
           const class SMWord& w2 )
    : word1( w1 ), word2( w2 )
  void takeControl( );
private:
  const class SMWord& word1;
  const class SMWord& word2;
};
```

———-

20.12.106 class APOfFreeCyclic

— class APOfFreeCyclic —

```
class APOfFreeCyclic_MaximalRoot : public FastComputation
{
public:
  APOfFreeCyclic_MaximalRoot(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.107 class APOfFreeCyclic

— class APOfFreeCyclic —

```
class APOfFreeCyclic_IsEltAProperPower : public FastComputation
{
public:
  APOfFreeCyclic_IsEltAProperPower(const class SMWord& w) : theWord( w )
  void takeControl( );
private:
  const class SMWord& theWord;
};
```

———-

20.12.108 class APOfFreeCyclic

— class APOfFreeCyclic —

```
class APOfFreeCyclic_IsEltAProperPowerOfSecond : public FastComputation
{
public:
  APOfFreeCyclic_IsEltAProperPowerOfSecond( const class SMWord& w1,
              const class SMWord& w2 )
    : word1( w1 ), word2( w2 )
  void takeControl( );
private:
  const class SMWord& word1;
  const class SMWord& word2;
};
```

———-

20.12.109 class FNGAutoIsIAAut

— class FNGAutoIsIAAut —

```
class FNGAutoIsIAAut : public FastComputation
{
public:
  FNGAutoIsIAAut(const class SMMap& m) : map( m )
  void takeControl( );
private:
  const class SMMap& map;
};
```

———-

20.12.110 class SGOfNGjoinSubgroupProblem

— class SGOfNGjoinSubgroupProblem —

```
class SGOfNGjoinSubgroupProblem : public FastComputation
{
public:
  SGOfNGjoinSubgroupProblem(const class SMSubgroup& s1, const class SMSubgroup& s2)
    : subgroup1(s1), subgroup2(s2)
  void takeControl( );
private:
  const class SMSubgroup& subgroup1;
  const class SMSubgroup& subgroup2;
```

```
};
```

———-

20.12.111 class NGLCStermGensProblem

— class NGLCStermGensProblem —

```
class NGLCStermGensProblem : public FastComputation
{
public:
  NGLCStermGensProblem( class SMFPGroup& g, int num = 1)
    : group(g),number(num)
  void takeControl( );
private:
  class SMFPGroup& group;
  int number;
};
```

———-

20.12.112 class MakeFreeProduct

— class MakeFreeProduct —

```
class MakeFreeProduct : public FastComputation
{
public:
  MakeFreeProduct( class SMFPGroup& g1, class SMFPGroup& g2);
  void takeControl( );
protected:
  void makeHomomorphisms(SMFPGroup* smo);
  class SMFPGroup& group1;
  class SMFPGroup& group2;
  FPGroup fpGroup1;
  FPGroup fpGroup2;
  bool isAbelian;
  class FreeProduct* fp;
};
```

———-

20.12.113 class MakeDirectProduct

— class MakeDirectProduct —

```
class MakeDirectProduct : public MakeFreeProduct
{
public:
  MakeDirectProduct( class SMFPGroup& g1, class SMFPGroup& g2)
    : MakeFreeProduct(g1,g2)
  void takeControl( );
};
```

———-

20.12.114 class MakeFactorGroup

— class MakeFactorGroup —

```
class MakeFactorGroup : public FastComputation
{
public:
  MakeFactorGroup( class SMFPGroup& F, class SMSubgroup& H)
    : theGroup( F ), theSubgroup( H )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
  class SMSubgroup& theSubgroup;
};
```

———-

20.12.115 class MakeListOfWords

— class MakeListOfWords —

```
class MakeListOfWords : public FastComputation
{
public:
  MakeListOfWords( EnumeratorProblem<Word>& e):
    enumerator( e )
  void takeControl( );
protected:
  EnumeratorProblem<Word>& enumerator;
};
```

———-

20.12.116 class MakeRipsConstruction

— class MakeRipsConstruction —

```
class MakeRipsConstruction : public FastComputation
{
public:
  MakeRipsConstruction(class SMFPGroup& G ) : theGroup( G )
  void takeControl( );
private:
  class SMFPGroup& theGroup;
};
```

———-

20.13 SMApps/include/FNWP.h

— FNWP.h —

```
#include "Supervisor.h"
#include "SMWord.h"
#include "ARCer.h"
```

———-

20.13.1 class FNWPArcer

— class FNWPArcer —

```
class FNWPArcer : public ARCer {
public:
  FNWPArcer( ComputationManager& );
  void setArguments( const FreeGroup&, int nClass, const SetOf<Word>& );
  Trichotomy getRetValue()
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  FreeGroup theGroup;
  int c;
  SetOf<Word> theWords;
  Trichotomy retValue;
};
```

———-

20.13.2 class FNWPCM

— class FNWPCM —

```
class FNWPCM : public ComputationManager {
public:
  FNWPCM( class Supervisor& boss );
  void init( const FreeGroup& group, int nClass, const SetOf<Word> words );
  Trichotomy areTrivial( )
  void takeControl( );
  void start( );
  void terminate( );
private:
  FreeGroup theGroup;
  int c;
  SetOf<Word> theWords;
  const Supervisor& theBoss;
  Trichotomy tAreTrivial;
  bool bStarted;
  bool bInited;
  FNWPArcer arcer;
};
```

———-

20.14 SMApps/include/FreeIsPartOfBasisProblem.h

— FreeIsPartOfBasisProblem.h —

```
#include "Supervisor.h"
#include "GeneralWhitehead.h"
#include "GAIsPartOfBasis.h"
#include "SMSetOfWords.h"
#include "SMWord.h"
```

———-

20.14.1 class GAIsPartOfBasisArcer

— class GAIsPartOfBasisArcer —

```
class GAIsPartOfBasisArcer : public ARCer
{
public:
  GAIsPartOfBasisArcer( ComputationManager& );
  ~GAIsPartOfBasisArcer( )
```

```
  void setArguments( FreeGroup f,Word w );
  Chars getComputationFileName() const { return compFile.getFileName();}
  Chars getResultFileName() const { return resultFile.getFileName();}
  const VectorOf<Word>& getAutomorphism() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  Word theWord;
  FreeGroup theGroup;
  File compFile;
  File resultFile;
  VectorOf<Word> theAuto;
};
```

———-

20.14.2 class GAIsPartOfBasisCM

— class GAIsPartOfBasisCM —

```
class GAIsPartOfBasisCM : public ComputationManager
{
public:
  GAIsPartOfBasisCM( class FreeIsPartOfBasisProblem&  );
  Trichotomy getAnswer() const
  Chars getComputationFileName() const
  Chars getResultFileName() const
  const VectorOf<Word>& getAutomorphism() const
  void takeControl( );
  void start( );
  void terminate( );
private:
  GAIsPartOfBasisArcer arcer;
  Trichotomy answer;
  const class SMWord& theWord;
  bool linkHasBeenSent;
};
```

———-

20.14.3 class FreeIsPartOfBasis

— class FreeIsPartOfBasis —

```
class FreeIsPartOfBasis : public ComputationManager
{
```

```
public:
  FreeIsPartOfBasis(class FreeIsPartOfBasisProblem& problemObject );
  ~FreeIsPartOfBasis( );
  Trichotomy answer( ) const
  Map getAutomorphism( ) const;
  Chars getFileName( )
  void takeControl( );
  void start( )
  void terminate( )
private:
  bool linkHasBeenSent;
  const class SMWord& theWord;
  Trichotomy theAnswer;
  FreeGroup theGroup;
  Map theAutomorphism;
  GeneralWhitehead* GW;
  Chars theFileName;
};
```

———-

20.14.4 class FreeIsPartOfBasisProblem

— class FreeIsPartOfBasisProblem —

```
class FreeIsPartOfBasisProblem : public Supervisor
{
public:
  FreeIsPartOfBasisProblem(const class SMWord& w);
  const class SMWord& getWord( ) const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const SMWord& theWord;
  bool link1HasBeenSent;
  bool link2HasBeenSent;
  Subordinate<FreeIsPartOfBasisProblem, FreeIsPartOfBasis> freeIsPartOfBasis;
  Subordinate<FreeIsPartOfBasisProblem, GAIsPartOfBasisCM> gaIsPartOfBasisCM;
};
```

———-

20.14.5 class FreeGeneralIsPartOfBasis

— class FreeGeneralIsPartOfBasis —

```
class FreeGeneralIsPartOfBasis : public ComputationManager
{
public:
  FreeGeneralIsPartOfBasis( class FreeGeneralIsPartOfBasisProblem&
          problemObject );
  ~FreeGeneralIsPartOfBasis( );
  Trichotomy answer( ) const
  Map getAutomorphism( ) const;
  Chars getFileName( )
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class SMSetOfWords& theSet;
  Trichotomy theAnswer;
  FreeGroup theGroup;
  Map theAutomorphism;
  GeneralWhitehead* GW;
  Chars theFileName;
};
```

———-

20.14.6 class FreeGeneralIsPartOfBasisProblem

— class FreeGeneralIsPartOfBasisProblem —

```
class FreeGeneralIsPartOfBasisProblem : public Supervisor
{
public:
  FreeGeneralIsPartOfBasisProblem( const class SMSetOfWords& );
  const class SMSetOfWords& getSetOfWords( ) const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const SMSetOfWords& theSet;
  bool linkHasBeenSent;
  Subordinate<FreeGeneralIsPartOfBasisProblem, FreeGeneralIsPartOfBasis>
    freeIsPartOfBasis;
};
```

———-

20.15 SMApps/include/FreeProblems.h

— **FreeProblems.h** —

```
#include "ComputationManager.h"
#include "Supervisor.h"
#include "RandomAutoInFree.h"
#include "SGofFreeGroup.h"
#include "File.h"
```

———-

20.15.1 class AutoInFreeIsFinitARCer

— **class AutoInFreeIsFinitARCer** —

```
class AutoInFreeIsFinitARCer : public ARCer
{
public:
  AutoInFreeIsFinitARCer( ComputationManager& boss )
    : ARCer( boss ),theMap( 0 ),retValue(dontknow)
  ~AutoInFreeIsFinitARCer( )
  void setArguments( const Map& map,const FreeGroup& group );
  Trichotomy getRetValue() const
  int getOrder() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  FreeGroup theGroup;
  Map* theMap;
  Trichotomy retValue;
  int theOrder;
};
```

———-

20.15.2 class AutoInFreeIsFinite

— **class AutoInFreeIsFinite** —

```
class AutoInFreeIsFinite : public Supervisor
{
public:
  AutoInFreeIsFinite(const class SMMap& hom);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
```

```
private:
  const class SMMap& theHom;
  AutoInFreeIsFinitARCer arcer;
};
```

———-

20.15.3 class SGOfFreeContainsConjugateARCer

— class SGOfFreeContainsConjugateARCer —

```
class SGOfFreeContainsConjugateARCer : public ARCer
{
public:
  SGOfFreeContainsConjugateARCer( ComputationManager& boss )
    : ARCer( boss ),theSubgroup( 0 )
  ~SGOfFreeContainsConjugateARCer( )
 void setArguments( const SGofFreeGroup& subgroup, const VectorOf<Word>& gens);
  bool getRetValue(Word& c) const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SGofFreeGroup* theSubgroup;
  bool retValue;
  Word conjugator;
  SetOf<Word> generators;
};
```

———-

20.15.4 class SGOfFreeContainsConjugate

— class SGOfFreeContainsConjugate —

```
class SGOfFreeContainsConjugate : public Supervisor
{
public:
  SGOfFreeContainsConjugate(const class SMSubgroup& S1,
                            const class SMSubgroup& S2);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  const class SMSubgroup& subgroup1;
  const class SMSubgroup& subgroup2;
```

```
  SGOfFreeContainsConjugateARCer arcer;
};
```

———-

20.15.5 class SGOfFreeConjugateToARCer

— class SGOfFreeConjugateToARCer —

```
class SGOfFreeConjugateToARCer : public ARCer
{
public:
  SGOfFreeConjugateToARCer( ComputationManager& boss )
    : ARCer( boss ),theSubgroup( 0 )
  ~SGOfFreeConjugateToARCer( )
 void setArguments(const SGofFreeGroup& subgroup, const VectorOf<Word>& gens );
  bool getRetValue( ) const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SGofFreeGroup* theSubgroup;
  bool retValue;
  SetOf<Word> generators;
};
```

———-

20.15.6 class SGOfFreeConjugateTo

— class SGOfFreeConjugateTo —

```
class SGOfFreeConjugateTo : public Supervisor
{
public:
  SGOfFreeConjugateTo(const class SMSubgroup& S1, const class SMSubgroup& S2);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  const class SMSubgroup& subgroup1;
  const class SMSubgroup& subgroup2;
  SGOfFreeConjugateToARCer arcer;
};
```

———-

20.15.7 class AutEnumeratorARCer

— class AutEnumeratorARCer —

```
class AutEnumeratorARCer : public ARCer
{
public:
  AutEnumeratorARCer( ComputationManager& boss )
    : ARCer( boss ), randAuto( 0 ), isFinite(0)
  ~AutEnumeratorARCer( )
  void setArguments( const FreeGroup& group,int avgNumbers, bool isfinite);
  Chars getFileName();
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  RandomAutoInFree* randAuto;
  bool isFinite;
  File file;
};
```

———-

20.15.8 class AutEnumerator

— class AutEnumerator —

```
class AutEnumerator : public ComputationManager
{
public:
  AutEnumerator(class SMFPGroup& F,int avgNumGens = 1, bool is_finite = false);
  ~AutEnumerator( );
  const IconID iconID( ) const;
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  class SMFPGroup& theGroup;
  AutEnumeratorARCer arcer;
  int avgNumGens;
  bool isFinite;
  bool isStarted;
};
```

———-

20.16 SMApps/include/GAConjProblemForORGroup.h

— GAConjProblemForORGroup.h —

```
#include "Supervisor.h"
#include "SMWord.h"
#include "FreeGroup.h"
#include "File.h"
#include "GACPforORGSolver.h"
```

———-

20.16.1 class GAConjugacyForORGroupARCer

— class GAConjugacyForORGroupARCer —

```
class GAConjugacyForORGroupARCer : public ARCer
{
public:
  GAConjugacyForORGroupARCer( class GAConjugacyForORGroup& );
  ~GAConjugacyForORGroupARCer( );
  void setArguments( const OneRelatorGroup& G, const Word& u, const Word& v );
  Trichotomy answer( ) const
  Chars getFileName( ) const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  OneRelatorGroup theGroup;
  Word firstWord, secondWord;
  GAConjProblemForORGroupSolver* solver;
  Trichotomy theAnswer;
};
```

———-

20.16.2 class GAConjugacyForORGroup

— class GAConjugacyForORGroup —

```
class GAConjugacyForORGroup : public ComputationManager
{
public:
  GAConjugacyForORGroup( const ConjugacyProblem& CP);
  Trichotomy answer( ) const
  Chars getFileName( ) const
```

```
  OneRelatorGroup getGroup( ) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class ConjugacyProblem& problem;
  GAConjugacyForORGroupARCer arcer;
  Trichotomy theAnswer;
};
```

———-

20.17 SMApps/include/GAEquations.h

— **GAEquations.h** —

```
#include "Supervisor.h"
#include "GAEquationSolver.h"
#include "List.h"
#include "ARCer.h"
#include "FPGroup.h"
#include "Map.h"
#include "File.h"
#include "SMEquation.h"
```

———-

20.17.1 class GAEquationArcer

— **class GAEquationArcer** —

```
class GAEquationArcer : public ARCer
{
public:
  GAEquationArcer( ComputationManager& boss )
    : ARCer( boss ), retValue(dontknow)
  void setArguments( FreeGroup, int, Word );
  Trichotomy haveSolution() const
  Map getSolution( ) const;
  Chars getFileName() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  Trichotomy retValue;
  Map solution;
```

```
  File file;
  FreeGroup theGroup;
  int numOfVars;
  Word theWord;
};
```

———-

20.17.2 class GAEquationCM

— class GAEquationCM —

```
class GAEquationCM : public ComputationManager
{
public:
  GAEquationCM( class GAEquationProblem& PO );
  Chars getFileName() const
  Trichotomy haveSolution() const
  Map getSolution( ) const;
  void viewStructure(ostream& ostr) const
  void takeControl( );
  void start( );
  void terminate( )
private:
  SMEquation2& theEquation;
  GAEquationArcer arcer;
  Trichotomy theAnswer;
  Map solution;
  Chars filename;
  bool bStarted;
};
```

———-

20.17.3 class GAEquationProblem

— class GAEquationProblem —

```
class GAEquationProblem : public Supervisor
{
public:
  GAEquationProblem( class SMEquation2& );
  SMEquation2& getEquation() const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
private:
  SMEquation2& theEquation;
```

```
  bool linkHasBeenSent;
  Subordinate<GAEquationProblem, GAEquationCM> GA;
};
```

———-

20.17.4 class TwoCommArcer

— class TwoCommArcer —

```
class TwoCommArcer : public ARCer
{
public:
  TwoCommArcer( ComputationManager& boss )
    : ARCer( boss ), retValue(dontknow)
  void setArguments( FreeGroup, Word );
  Trichotomy haveSolution() const
  Chars getFileName() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  Trichotomy retValue;
  File file;
  FreeGroup theGroup;
  Word theWord;
};
```

———-

20.17.5 class TwoCommCM

— class TwoCommCM —

```
class TwoCommCM : public ComputationManager
{
public:
  TwoCommCM( class TwoCommProblem& PO );
  Chars getFileName() const
  Trichotomy haveSolution() const
  void viewStructure(ostream& ostr) const { }
  void takeControl( );
  void start( );
  void terminate( )
private:
  SMWord& theWord;
  TwoCommArcer arcer;
```

```
  Trichotomy theAnswer;
  bool bStarted;
};
```

———-

20.17.6 class TwoCommProblem

— class TwoCommProblem —

```
class TwoCommProblem : public Supervisor
{
public:
  TwoCommProblem( class SMWord& );
  SMWord& getWord() const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
private:
  SMWord& theWord;
  bool linkHasBeenSent;
  Subordinate<TwoCommProblem, TwoCommCM> GA;
};
```

———-

20.18 SMApps/include/GAWordProblemForORGroup.h

— GAWordProblemForORGroup.h —

```
#include "Supervisor.h"
#include "SMWord.h"
#include "FreeGroup.h"
#include "File.h"
#include "GACPforORGSolver.h"
```

———-

20.18.1 class GAWordForORGroupARCer

— class GAWordForORGroupARCer —

```
class GAWordForORGroupARCer : public ARCer
{
public:
  GAWordForORGroupARCer( class GAWordForORGroup& );
```

```
  ~GAWordForORGroupARCer( );
  void setArguments( const OneRelatorGroup& G, const Word& u );
  Trichotomy answer( ) const
  Chars getFileName( ) const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  OneRelatorGroup theGroup;
  Word theWord;
  GAConjProblemForORGroupSolver* solver;
  Trichotomy theAnswer;
};
```

———-

20.18.2 class GAWordForORGroup

— class GAWordForORGroup —

```
class GAWordForORGroup : public ComputationManager
{
public:
  GAWordForORGroup(class WordProblem& CP);
  Trichotomy answer( ) const
  Chars getFileName( ) const
  OneRelatorGroup getGroup( ) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const WordProblem& problem;
  GAWordForORGroupARCer arcer;
  Trichotomy theAnswer;
};
```

———-

20.19 SMApps/include/GeneticProblems.h

— GeneticProblems.h —

```
#include "Supervisor.h"
#include "SMWord.h"
#include "ARCer.h"
```

———-

20.19.1 struct GeneticWPBase

— **struct GeneticWPBase** —

```
struct GeneticWPBase {
  enum DetailType { NO_DETAILS, WORD, SET_OF_WORDS, COMMUTATORS };
};
```

———-

20.19.2 class GeneticWPArcer

— **class GeneticWPArcer** —

```
class GeneticWPArcer : public ARCer, public GeneticWPBase {
public:
  GeneticWPArcer( ComputationManager& );
  ~GeneticWPArcer( )
  void setArguments( const FPGroup&, const SetOf<Word>&,
         DetailType dt = NO_DETAILS );
  Trichotomy getRetValue()
  Chars getFileName();
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  FPGroup theGroup;
  SetOf<Word> theWords;
  Trichotomy retValue;
  DetailType dtype;
  File *file;
};
```

———-

20.19.3 class GeneticWPCM

— **class GeneticWPCM** —

```
class GeneticWPCM : public ComputationManager, public GeneticWPBase {
public:
  GeneticWPCM( class Supervisor& boss );
  void init( const FPGroup& group, const SetOf<Word> words,
       DetailType dt = NO_DETAILS );
  Trichotomy areTrivial( )
  Chars getFileName()
```

```
  void takeControl( );
  void start( );
  void terminate( );
private:
  FPGroup theGroup;
  SetOf<Word> theWords;
  const Supervisor& theBoss;
  Trichotomy tAreTrivial;
  bool bStarted;
  bool bInited;
  DetailType dtype;
  GeneticWPArcer arcer;
};
```

———-

20.20 SMApps/include/HNNProblems.h

— **HNNProblems.h** —

```
#include "IsFreeProblem.h"
#include "fastProblems.h"
```

———-

20.20.1 class MakeHNNExtOfFreeGroup

— **class MakeHNNExtOfFreeGroup** —

```
class MakeHNNExtOfFreeGroup : public FastComputation
{
public:
  MakeHNNExtOfFreeGroup( const SMFPGroup& G,
       const class SMSubgroup& S1, const class SMSubgroup& S2 )
    : theStableLetter( "t" ),
      subgroup1( S1 ), subgroup2( S2 )
  void takeControl( );
private:
  const Chars theStableLetter;
  const class SMSubgroup& subgroup1;
  const class SMSubgroup& subgroup2;
};
```

———-

20.20.2 class HNNofFreeGroup

— class HNNofFreeGroup —

```
class HNNofFreeGroup_IsTrivial : public FastComputation
{
public:
  HNNofFreeGroup_IsTrivial( const SMFPGroup& G )
    : group( G )
  void takeControl( );
private:
  const SMFPGroup& group;
};
```

———-

20.20.3 class HNNofFreeGroup

— class HNNofFreeGroup —

```
class HNNofFreeGroup_IsFinite : public FastComputation
{
public:
  HNNofFreeGroup_IsFinite( const SMFPGroup& G )
    : group( G )
  void takeControl( );
private:
  const SMFPGroup& group;
};
```

———-

20.20.4 class HNNofFreeGroup

— class HNNofFreeGroup —

```
class HNNofFreeGroup_IsAbelian : public FastComputation
{
public:
  HNNofFreeGroup_IsAbelian( const SMFPGroup& G )
    : group( G )
  void takeControl( );
private:
  const SMFPGroup& group;
};
```

———-

20.20.5 class HNNofFreeGroup

— class HNNofFreeGroup —

```
class HNNofFreeGroup_ConjugacyProblem : public FastComputation
{
public:
  HNNofFreeGroup_ConjugacyProblem( const SMWord& x, const SMWord& y )
    : word1( x ), word2( y )
  void takeControl( );
private:
  const SMWord& word1;
  const SMWord& word2;
};
```

———-

20.20.6 class HNNofFreeGroup

— class HNNofFreeGroup —

```
class HNNofFreeGroup_WordProblem : public FastComputation
{
public:
  HNNofFreeGroup_WordProblem( const SMWord& x ) : word( x )
  void takeControl( );
private:
  const SMWord& word;
};
```

———-

20.20.7 class HNNofFreeGroup

— class HNNofFreeGroup —

```
class HNNofFreeGroup_ReducedForm : public FastComputation
{
public:
  HNNofFreeGroup_ReducedForm( const SMWord& x ) : word( x )
  void takeControl( );
private:
  const SMWord& word;
};
```

———-

20.20.8 class HNNofFreeGroup

— class HNNofFreeGroup —

```
class HNNofFreeGroup_NormalForm : public FastComputation
{
public:
  HNNofFreeGroup_NormalForm( const SMWord& x ) : word( x )
  void takeControl( );
private:
  const SMWord& word;
};
```

———-

20.20.9 class HNNofFreeGroup

— class HNNofFreeGroup —

```
class HNNofFreeGroup_CyclicallyReducedForm : public FastComputation
{
public:
  HNNofFreeGroup_CyclicallyReducedForm( const SMWord& x ) : word( x )
  void takeControl( );
private:
  const SMWord& word;
};
```

———-

20.20.10 class HNNofFreeGroup

— class HNNofFreeGroup —

```
class HNNofFreeGroup_IsProperPowerOfSecond : public FastComputation
{
public:
  HNNofFreeGroup_IsProperPowerOfSecond( const SMWord& x, const SMWord& y )
    : word1( x ), word2( y )
  void takeControl( );
private:
  const SMWord& word1;
  const SMWord& word2;
};
```

———-

20.20.11 class HNNofFreeGroup

— class HNNofFreeGroup —

```
class HNNofFreeGroup_IsProperPower : public FastComputation
{
public:
  HNNofFreeGroup_IsProperPower( const SMWord& x ) : word( x )
  void takeControl( );
private:
  const SMWord& word;
};
```

———-

20.20.12 class HNNofFreeGroup

— class HNNofFreeGroup —

```
class HNNofFreeGroup_MaximalRoot : public FastComputation
{
public:
  HNNofFreeGroup_MaximalRoot( const SMWord& x ) : word( x )
  void takeControl( );
private:
  const SMWord& word;
};
```

———-

20.20.13 class HNNofFreeGroup

— class HNNofFreeGroup —

```
class HNNofFreeGroup_AreEqual : public FastComputation
{
public:
  HNNofFreeGroup_AreEqual( const SMWord& x, const SMWord& y )
    : word1( x ), word2( y )
  void takeControl( );
private:
  const SMWord& word1;
  const SMWord& word2;
};
```

———-

20.20.14 class HNNofFreeGroup

— class HNNofFreeGroup —

```
class HNNofFreeGroup_IsSGTrivial : public FastComputation
{
public:
  HNNofFreeGroup_IsSGTrivial( const SMSubgroup& S ) : subgroup( S )
  void takeControl( );
private:
  const SMSubgroup& subgroup;
};
```

———-

20.20.15 class HNNofFreeGroup

— class HNNofFreeGroup —

```
class HNNofFreeGroup_IsSGAbelian : public FastComputation
{
public:
  HNNofFreeGroup_IsSGAbelian( const SMSubgroup& S ) : subgroup( S )
  void takeControl( );
private:
  const SMSubgroup& subgroup;
};
```

———-

20.20.16 class CheckinHNNofFreeGroup

— class CheckinHNNofFreeGroup —

```
class CheckinHNNofFreeGroup : public FastComputation
{
public:
  CheckinHNNofFreeGroup( )
  void takeControl( );
};
```

———-

20.20.17 class HNNofFreeIsFree

— class HNNofFreeIsFree —

```
class HNNofFreeIsFree : public ComputationManager
{
public:
  HNNofFreeIsFree( class SMFPGroup& F );
  ~HNNofFreeIsFree();
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMFPGroup& theGroup;
  ORProblems* theORProblems;
  ORIsFreeProblemARCer arcer;
};
```

———-

20.20.18 class APofFreeIsFreeArcer

— class APofFreeIsFreeArcer —

```
class APofFreeIsFreeArcer : public ARCer
{
public:
  APofFreeIsFreeArcer( ComputationManager& boss )
    : ARCer( boss ), theAnswer( dontknow ), theGroup( 0 )
  ~APofFreeIsFreeArcer( )
  void setArguments( const AmalgProductOfFreeGroups& G );
  Trichotomy answer() const
  void runComputation( );
  void writeResults( ostream& ostr );
  void readResults( istream& istr );
private:
  AmalgProductOfFreeGroups *theGroup;
  Trichotomy theAnswer;
};
```

———-

20.20.19 class APofFreeIsFree

— class APofFreeIsFree —

```
class APofFreeIsFree : public ComputationManager
{
public:
  APofFreeIsFree( class SMFPGroup& F );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMFPGroup& theGroup;
  APofFreeIsFreeArcer arcer;
};
```

———-

20.20.20 class APofFreeIsPerfect

— class APofFreeIsPerfect —

```
class APofFreeIsPerfect : public Supervisor
{
public:
  APofFreeIsPerfect( class SMFPGroup& F );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMFPGroup& theGroup;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.20.21 class APofFreeHomologyArcer

— class APofFreeHomologyArcer —

```
class APofFreeHomologyArcer : public ARCer
{
public:
  APofFreeHomologyArcer( ComputationManager& boss )
    : ARCer( boss ), theGroup( 0 ), theRankOfH2(-1)
  ~APofFreeHomologyArcer( )
  void setArguments( const AmalgProductOfFreeGroups& G );
  int rankOfH2() const
  void runComputation( );
```

```
  void writeResults( ostream& ostr );
  void readResults( istream& istr );
private:
  AmalgProductOfFreeGroups *theGroup;
  int theRankOfH2;
};
```

———-

20.20.22 class APofFreeHomologyProblem

— class APofFreeHomologyProblem —

```
class APofFreeHomologyProblem : public Supervisor
{
public:
  APofFreeHomologyProblem( class SMFPGroup& F );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMFPGroup& theGroup;
  APofFreeHomologyArcer arcer;
  MirrorSubordinate abelianInvariants;
  bool cyclicSubgroup;
  bool abDone;
  bool arcerStarted;
};
```

———-

20.21 SMApps/include/HomologyProblem.h

— HomologyProblem.h —

```
#include "Supervisor.h"
#include "Vector.h"
```

———-

20.21.1 class HomologyARCer

— class HomologyARCer —

```
class HomologyARCer : public ARCer
{
public:
  HomologyARCer( ComputationManager& );
  ~HomologyARCer( );
  void setArguments( const class KBMachine& M, int startDimension,
         int endDimension );
  VectorOf<Integer> getTorsionInvariants( ) const
  int getTorsionFreeRank() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  class KBMachine *kbmachine;
  int startdim, enddim;
  VectorOf<Integer> torsionInvariants;
  int torsionFreeRank;
};
```

———-

20.21.2 class HomologyProblem

— class HomologyProblem —

```
class HomologyProblem : public ComputationManager
{
public:
  HomologyProblem( const class HomologySupervisor& );
  bool isSolved() const
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMFPGroup& theGroup;
  int theDimension;
  HomologyARCer homologyArcer;
  bool arcerStarted;
  bool solved;
};
```

———-

20.21.3 class HomologySupervisor

— class HomologySupervisor —

```
class HomologySupervisor : public Supervisor
{
public:
  HomologySupervisor(class SMFPGroup& G, int d = 2 );
  SMFPGroup& group() const
  int dimension( ) const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMFPGroup& theGroup;
  int theDimension;
  Subordinate<HomologySupervisor, HomologyProblem> homology;
  MirrorSubordinate kbSupervisor;
};
```

———-

20.22 SMApps/include/HToddCoxeter.h

— **HToddCoxeter.h** —

```
#include "Supervisor.h"
#include "HavasTC.h"
#include "File.h"
```

———-

20.22.1 class HToddCoxeterARCer

— **class HToddCoxeterARCer** —

```
class HToddCoxeterARCer : public ARCer
{
public:
  HToddCoxeterARCer( ComputationManager& boss,const FPGroup& group ) :
    ARCer( boss ),
    tc(group),
    theGroup(group),
    theIndex( 0 ),
    success( false )
  HToddCoxeterARCer( ComputationManager& boss,const FPGroup& group,
        const VectorOf<Word>& subgroup) :
    ARCer( boss ),
    tc(group,subgroup),
```

```
    theGroup(group),
    theIndex( 0 ),
    success( false )
  int getIndex() const
  bool isSuccessful() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  HavasTC tc;
  FPGroup theGroup;
  int theIndex;
  bool success;
};
```

———-

20.22.2 class HToddCoxeter

— class HToddCoxeter —

```
class HToddCoxeter : public ComputationManager
{
public:
  HToddCoxeter(class GCM& boss);
  void takeControl( );
  void start( )
  void terminate( )
private:
  class GCM& theBoss;
  class SMFPGroup& theGroup;
  HToddCoxeterARCer* arcer;
};
```

———-

20.22.3 class HSGIndexToddCoxeter

— class HSGIndexToddCoxeter —

```
class HSGIndexToddCoxeter : public ComputationManager
{
public:
  HSGIndexToddCoxeter(class SCM& scm);
  void takeControl( );
  void start( )
  void terminate( )
```

```
private:
  class SCM& theSCM;
  class SMSubgroup& theSubgroup;
  HToddCoxeterARCer arcer;
};
```

———-

20.23 SMApps/include/IsAbelianProblem.h

— **IsAbelianProblem.h** —

```
#include "Supervisor.h"
#include "SMFPGroup.h"
#include "NilpotentQuotients.h"
#include "GeneticProblems.h"
```

———-

20.23.1 class IsAbelianChecker

— **class IsAbelianChecker** —

```
class IsAbelianChecker
{
public:
  IsAbelianChecker( class SMFPGroup& );
  Trichotomy isAbelian( );
  Chars getExplanation( )
  Chars commutator( VectorOf<int> components );
  bool haveDetails ( ) const;
  Chars getDetailsFileName( ) const;
private:
  FPGroup G;
  class GIC& gic;
  class GCM& gcm;
  bool triedAbelianization;
  bool triedOneRelator;
  Chars explanation;
  DetailedReport abelianDetails;
};
```

———-

20.23.2 class IsAbelianProblem

— class IsAbelianProblem —

```
class IsAbelianProblem : public Supervisor
{
public:
  IsAbelianProblem( class SMFPGroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  SMFPGroup& theGroup;
  IsAbelianChecker theChecker;
  MirrorSubordinate normalClosure;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate computeBasis;
  Subordinate<IsAbelianProblem,NilpotentWPInQuotients> nilpotentWPInQuotients;
  MirrorSubordinate nilpotentQuotients;
  Subordinate<IsAbelianProblem,NilpotentWP> nilpotentWP;
  Subordinate<IsAbelianProblem, GeneticWPCM> genetic;
};

inline
bool IsAbelianChecker::haveDetails ( ) const

inline
Chars IsAbelianChecker::getDetailsFileName( ) const
```

———-

20.24 SMApps/include/IsEltCentral.h

— IsEltCentral.h —

```
#include "Supervisor.h"
#include "SetOfWordsChecker.h"
#include "NilpotentQuotients.h"
#include "GeneticProblems.h"
```

———-

20.24.1 class IsEltCentral

— class IsEltCentral —

```
class IsEltCentral : public Supervisor
{
public:
  IsEltCentral( const class SMWord& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const SMWord& theWord;
  SetOfWordsChecker theChecker;
  MirrorSubordinate normalClosure;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate nilpotentQuotients;
  Subordinate<IsEltCentral,NilpotentWPInQuotients> nilpotentWPInQuotients;
  MirrorSubordinate computeBasis;
  Subordinate<IsEltCentral,NilpotentWP> nilpotentWP;
  Subordinate<IsEltCentral, GeneticWPCM> genetic;
};
```

———-

20.25 SMApps/include/IsFiniteProblem.h

— IsFiniteProblem.h —

```
#include "Supervisor.h"
#include "SMFPGroup.h"
```

———-

20.25.1 class IsFiniteProblem

— class IsFiniteProblem —

```
class IsFiniteProblem : public Supervisor
{
public:
  IsFiniteProblem( SMFPGroup& );
```

```
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  SMFPGroup& theGroup;
  MirrorSubordinate abelianRank;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate theToddCoxeter;
  MirrorSubordinate ghToddCoxeter;
};
```

———-

20.26 SMApps/include/IsFreeProblem.h

— **IsFreeProblem.h** —

```
#include "SMFPGroup.h"
#include "ORProblems.h"
#include "ARCer.h"
```

———-

20.26.1 class ORIsFreeProblemARCer

— **class ORIsFreeProblemARCer** —

```
class ORIsFreeProblemARCer : public ARCer
{
public:
  ORIsFreeProblemARCer( ComputationManager& boss ) : ARCer( boss )
  void setArguments(ORProblems* );
  bool getRetValue()
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  ORProblems* theORProblems;
  bool retValue;
};
```

———-

20.26.2 class ORIsFreeProblem

— class ORIsFreeProblem —

```
class ORIsFreeProblem : public ComputationManager
{
public:
  ORIsFreeProblem( class SMFPGroup& );
  ~ORIsFreeProblem( );
  void viewStructure( ostream& ) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const SMFPGroup& theGroup;
  ORProblems* theORProblems;
  ORIsFreeProblemARCer arcer;
};
```

———-

20.26.3 class IsFreeChecker

— class IsFreeChecker —

```
class IsFreeChecker
{
public:
  IsFreeChecker( class SMFPGroup& );
  Trichotomy isFree( );
  Chars getExplanation( )
private:
  class SMFPGroup& theGroup;
  FPGroup G;
  class GIC& gic;
  class GCM& gcm;
  bool triedAbelianization;
  Chars explanation;
};
```

———-

20.26.4 class IsFreeProblem

— class IsFreeProblem —

```
class IsFreeProblem : public Supervisor
{
public:
  IsFreeProblem( class SMFPGroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  const SMFPGroup& theGroup;
  IsFreeChecker theChecker;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.27 SMApps/include/IsNilpotentProblem.h

— IsNilpotentProblem.h —

```
#include "Supervisor.h"
#include "SMFPGroup.h"
#include "CommutatorsChecker.h"
#include "NilpotentQuotients.h"
```

———-

20.27.1 class IsNilpotentProblem

— class IsNilpotentProblem —

```
class IsNilpotentProblem : public Supervisor
{
public:
  IsNilpotentProblem( class SMFPGroup& G, int nilpClass = 2 );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMFPGroup& theGroup;
  int theClass;
  CommutatorsChecker theChecker;
  CommutatorsCheckerARCer theArcer;
  MirrorSubordinate normalClosure;
  MirrorSubordinate kbSupervisor;
```

```
  MirrorSubordinate agSupervisor;
  MirrorSubordinate nilpotentQuotients;
  Subordinate<IsNilpotentProblem,NilpotentWPInQuotients>
     nilpotentWPInQuotients;
};
```

———-

20.28 SMApps/include/IsTrivialProblem.h

— IsTrivialProblem.h —

```
#include "Supervisor.h"
#include "FEData.h"
#include "Word.h"
#include "FPGroup.h"
#include "Chars.h"
#include "GroupFastChecks.h"
#include "NilpotentQuotients.h"
#include "GeneticProblems.h"
```

———-

20.28.1 class IsTrivialChecker

— class IsTrivialChecker —

```
class IsTrivialChecker
{
public:
  IsTrivialChecker( class SMFPGroup& );
  Trichotomy isTrivial( );
  Chars getExplanation( )
  Chars getDetailsFileName( ) const;
  bool haveDetails( ) const;
private:
  class SMFPGroup& theGroup;
  FPGroup G;
  GroupFastChecks checker;
  class GIC& gic;
  class GCM& gcm;
  bool triedAbelianization;
  bool triedPreliminaryCheckings;
  Chars explanation;
  DetailedReport trivialDetails;
  Trichotomy preliminaryCheckings();
```

```
};
```

———-

20.28.2 class IsTrivialProblem

— class IsTrivialProblem —

```
class IsTrivialProblem : public Supervisor
{
public:
  IsTrivialProblem( class SMFPGroup& );
  Trichotomy answer( ) const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  const SMFPGroup& theGroup;
  IsTrivialChecker theChecker;
  Trichotomy theAnswer;
  Chars explanation;
  MirrorSubordinate normalClosure;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate nilpotentQuotients;
  Subordinate<IsTrivialProblem,NilpotentWPInQuotients> nilpotentWPInQuotients;
  Subordinate<IsTrivialProblem, GeneticWPCM> genetic;
};

inline
Chars IsTrivialChecker::getDetailsFileName( ) const

inline
bool IsTrivialChecker::haveDetails( ) const
```

———-

20.29 SMApps/include/IsWordAPE.h

— IsWordAPE.h —

```
#include "TurnerProperSubgroupEnumerator.h"
#include "FreeGroup.h"
#include "Supervisor.h"
```

```
#include "RankOfSubgroup.h"
```

———-

20.29.1 class IsWordAPEARCer

— class IsWordAPEARCer —

```
class IsWordAPEARCer : public ARCer
{
public:
  IsWordAPEARCer( ComputationManager& boss,
      const class FreeGroup& group,
      const Word& word );
  ~IsWordAPEARCer();
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
  Chars getFileName( ) const
  Trichotomy getAnswer() const
  bool HaveSG() const
  bool getSG(SGofFreeGroup& sg)
private:
  const class FreeGroup theGroup;
  const Word theWord;
  const static int period = 1000;
  Trichotomy answer;
  SGofFreeGroup* theSG;
  bool haveSG;
  ProperSubgroupEnumerator* PSE;
  void lookingup();
  bool checkForPrimitive(SGofFreeGroup sg);
  void printSG( ostream& file, const SGofFreeGroup& subgroup );
```

———-

20.29.2 class pairSG

— class pairSG —

```
  class pairSG_bool
    {
    public:
      pairSG_bool(const SGofFreeGroup& sg) : theSG(sg) , checked(false)
      pairSG_bool(const SGofFreeGroup& sg, bool b) : theSG(sg), checked(b)
      inline bool operator == (const pairSG_bool& sg) const
      inline int hash() const
```

```
      void setChecked( )
      bool getChecked( ) const
      SGofFreeGroup getSG() const
    private:
      SGofFreeGroup theSG;
      bool checked;
    };
};
```

———-

20.29.3 class IsWordAPE

— class IsWordAPE —

```
class IsWordAPE : public ComputationManager
{
public:
  IsWordAPE(class IsWordAPEProblem& problemObject );
  ~IsWordAPE( );
  Trichotomy getAnswer() const
  bool haveSG() const
  bool getSG(SGofFreeGroup& sg)
  Chars getFileName( )
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class SMWord& theWord;
  FreeGroup theGroup;
  IsWordAPEARCer theARCer;
};
```

———-

20.29.4 class IsWordAPEProblem

— class IsWordAPEProblem —

```
class IsWordAPEProblem : public Supervisor
{
public:
  IsWordAPEProblem(const class SMWord& w);
  const class SMWord& getWord( ) const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
```

```
  void terminate( )
private:
  const SMWord& theWord;
  bool linkHasBeenSent;
  Subordinate<IsWordAPEProblem, IsWordAPE> isWordAPE;
};
```

———-

20.30 SMApps/include/KBModule.h

— **KBModule.h** —

```
#include <signal.h>
#include <unistd.h>
#include "Timer.h"
#include "Supervisor.h"
#include "RKBPackage.h"
#include "KBMachine.h"
#include "List.h"
#include "ARCer.h"
#include "FPGroup.h"
```

———-

20.30.1 class KBSupervisorARCer

— **class KBSupervisorARCer** —

```
class KBSupervisorARCer : public ARCer
{
public:
  KBSupervisorARCer( ComputationManager& boss ) : ARCer( boss )
  void setArguments( const FPGroup& );
  KBMachine getKBMachine()
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  FPGroup G;
  RKBPackage* RKBP;
  KBMachine KBM;
};
```

———-

20.30.2 class KBSupervisor

— class KBSupervisor —

```
class KBSupervisor : public ComputationManager
{
public:
  KBSupervisor(class GCM& gcm);
  void viewStructure(ostream& ostr) const
  void takeControl( );
  void start( );
  void terminate( );
private:
  SMFPGroup& theGroup;
  KBSupervisorARCer arcer;
};
```

———-

20.30.3 class KBProblem

— class KBProblem —

```
class KBProblem : public Supervisor
{
public:
  KBProblem(class SMFPGroup& G);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( )
private:
  SMFPGroup& theGroup;
  MirrorSubordinate kbSupervisor;
};
```

———-

20.31 SMApps/include/KernelPresentation.h

— KernelPresentation.h —

```
#include "Supervisor.h"
#include "KBModule.h"
#include "FEData.h"
#include "Word.h"
```

```
#include "FPGroup.h"
#include "SMVectorOfWords.h"
#include "Chars.h"
#include "PresentationProblems.h"
```

———-

20.31.1 class FPNewPresentationARCer

— class FPNewPresentationARCer —

```
class FPNewPresentationARCer : public ARCer
{
public:
  FPNewPresentationARCer( ComputationManager& boss ) :
    ARCer( boss ),
    dntg( false )
  void setArguments( const KBMachine& kb ,
         const FPGroup& g ,
         const VectorOf<Word>& y );
  FPGroup getRetValue()
  bool doesntGenerate()
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  NewPresentation N;
  FPGroup G;
  VectorOf<Word> newGens;
  bool dntg;
  FPGroup retValue;
};
```

———-

20.31.2 class FPNewPresentationProblem

— class FPNewPresentationProblem —

```
class FPNewPresentationProblem : public Supervisor
{
public:
  FPNewPresentationProblem( class SMVectorOfWords& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
```

```
private:
  SMVectorOfWords& theVector;
  bool init;
  FPNewPresentationARCer arcer;
  MirrorSubordinate kbSupervisor;
};
```

———-

20.31.3 class FPImagePresentationARCer

— class FPImagePresentationARCer —

```
class FPImagePresentationARCer : public ARCer
{
public:
  FPImagePresentationARCer( ComputationManager& boss ) : ARCer( boss )
  void setArguments( const FPGroup& preimage, const FPGroup& image,
         const VectorOf<Word>& y );
  FPGroup getRetValue()
  VectorOf<Word> getVectorOfImages()
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  ImageOfHom Im;
  VectorOf<Word> newImages;
  FPGroup retValue;
};
```

———-

20.31.4 class FPImagePresentationCM

— class FPImagePresentationCM —

```
class FPImagePresentationCM : public ComputationManager
{
public:
  FPImagePresentationCM( class FPKernelPresentationProblem& );
  VectorOf<Word> getVectorOfImages()
  FPGroup getImage()
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMHomomorphism2& theMap;
```

```
  FPGroup G;
  VectorOf<Word> im;
  FPImagePresentationARCer arcer;
};
```

———-

20.31.5 class FPKernelPresentationARCer

— class FPKernelPresentationARCer —

```
class FPKernelPresentationARCer : public ARCer
{
public:
  FPKernelPresentationARCer( ComputationManager& boss ) : ARCer( boss )
  void setArguments( const KBMachine& kb , const FPGroup& preimage ,
         const FPGroup& image , const VectorOf<Word>& y );
  FPGroup getRetValue()
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  KernelOfHom K;
  FPGroup retValue;
};
```

———-

20.31.6 class FPKernelPresentationCM

— class FPKernelPresentationCM —

```
class FPKernelPresentationCM : public ComputationManager
{
public:
  FPKernelPresentationCM( class FPKernelPresentationProblem& );
  void setFlag( bool f )
  FPGroup getResult()
  void takeControl( );
  void start( )
  void terminate( )
private:
  FPKernelPresentationProblem& KPP;
  bool init;
  bool flag;
  FPGroup result;
  FPKernelPresentationARCer arcer;
```

```
};
```

———-

20.31.7 class KBSupervisorCM

— class KBSupervisorCM —

```
class KBSupervisorCM : public ComputationManager
{
public:
  KBSupervisorCM( class FPKernelPresentationProblem& );
  void setFlag( bool f )
  void viewStructure(ostream& ostr) const
  void takeControl( );
  void start( )
  void terminate( )
private:
  FPKernelPresentationProblem& KPP;
  bool init;
  bool flag;
  KBSupervisorARCer arcer;
};
```

———-

20.31.8 class FPKernelPresentationProblem

— class FPKernelPresentationProblem —

```
class FPKernelPresentationProblem : public Supervisor
{
public:
  FPKernelPresentationProblem( class SMHomomorphism2& );
  SMHomomorphism2& getInitialHomo()
  SMHomomorphism2& getHomo()
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMHomomorphism2& theMap;
  SMHomomorphism2* theMap2;
  bool foundImage;
  bool foundKB;
  Subordinate<FPKernelPresentationProblem,FPKernelPresentationCM> kernel;
  Subordinate<FPKernelPresentationProblem,FPImagePresentationCM> image;
```

```
  Subordinate<FPKernelPresentationProblem,KBSupervisorCM> kbs;
};
```

———-

20.32 SMApps/include/MakeRandomPresentation.h

— MakeRandomPresentation.h —

```
#include "Supervisor.h"
#include "CosetEnumerator.h"
#include "File.h"
```

———-

20.32.1 class RandomPresentationARCer

— class RandomPresentationARCer —

```
class RandomPresentationARCer : public ARCer
{
public:
  RandomPresentationARCer( ComputationManager& boss )
    : ARCer( boss ), maxGens(10),  maxRels(10), averageRels(20)
  void setArguments(int maxG,int maxR,int averR){
    maxGens = maxG;
    maxRels = maxR;
    averageRels = averR;
  }
  FPGroup getRetValue() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  FPGroup theGroup;
  int maxGens;
  int maxRels;
  int averageRels;
};
```

———-

20.32.2 class MakeRandomPresentation

— class MakeRandomPresentation —

```
class MakeRandomPresentation : public Supervisor
{
public:
  MakeRandomPresentation();
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( )
private:
  RandomPresentationARCer arcer;
  bool arcerStarted;
  int nOfPresentations;
};
```

———-

20.33 SMApps/include/NGSubgroupProblems.h

— NGSubgroupProblems.h —

```
#include "Supervisor.h"
#include "File.h"
#include "ARCer.h"
#include "SMFPGroup.h"
#include "SMWord.h"
#include "SGOfNilpotentGroup.h"
```

———-

20.33.1 class SGOfNGinitPreimageARCer

— class SGOfNGinitPreimageARCer —

```
class SGOfNGinitPreimageARCer : public ARCer
{
public:
  SGOfNGinitPreimageARCer( ComputationManager& boss )
    : ARCer( boss ),theSNG( 0 )
  ~SGOfNGinitPreimageARCer( )
  void setArguments(const NilpotentGroup& NG ,
        const VectorOf<Word>& gens);
  const SGOfNilpotentGroup& getSubgroup() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
```

```
  SGOfNilpotentGroup* theSNG;
};
```

———-

20.33.2 class SGOfNGinitPreimageProblem

— class SGOfNGinitPreimageProblem —

```
class SGOfNGinitPreimageProblem : public ComputationManager
{
public:
  SGOfNGinitPreimageProblem(class SCM& scm);
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SCM& theSCM;
  class GIC& theGIC;
  SGOfNGinitPreimageARCer arcer;
};
```

———-

20.33.3 class SGOfNGinitializeARCer

— class SGOfNGinitializeARCer —

```
class SGOfNGinitializeARCer : public ARCer
{
public:
  SGOfNGinitializeARCer( ComputationManager& boss )
    : ARCer( boss ),theSNG( 0 )
  ~SGOfNGinitializeARCer( )
  void setArguments(const NilpotentGroup& FNG ,
        const VectorOf<Word>& gens);
  const SGOfNilpotentGroup& getSubgroup() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SGOfNilpotentGroup* theSNG;
};
```

———-

20.33.4 class SGOfNGinitializeProblem

— class SGOfNGinitializeProblem —

```
class SGOfNGinitializeProblem : public ComputationManager
{
public:
  SGOfNGinitializeProblem(class SCM& scm);
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SCM& theSCM;
  class GIC& theGIC;
  SGOfNGinitializeARCer arcer;
  bool haveParentInited;
};
```

———-

20.33.5 class SGOfNGcomputeBasisProblem

— class SGOfNGcomputeBasisProblem —

```
class SGOfNGcomputeBasisProblem : public Supervisor
{
public:
  SGOfNGcomputeBasisProblem(class SMSubgroup& sg);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  MirrorSubordinate initializeSGOfNG;
  MirrorSubordinate computeBasis;
  class SMSubgroup& theSubgroup;
};
```

———-

20.33.6 class SGOfNGDecomposeWordARCer

— class SGOfNGDecomposeWordARCer —

```
class SGOfNGDecomposeWordARCer : public ARCer
{
```

```
public:
  SGOfNGDecomposeWordARCer( ComputationManager& boss )
    : ARCer( boss ),theSNG(NULL)
  ~SGOfNGDecomposeWordARCer()
  void setArguments( const SGOfNilpotentGroup&, const PolyWord& );
  const PolyWord& decomposition()
  bool contains()
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SGOfNilpotentGroup* theSNG;
  bool isContain;
  PolyWord theDecomposition;
  PolyWord theWord;
};
```

———-

20.33.7 class SGOfNGDecomposeWordProblem

— class SGOfNGDecomposeWordProblem —

```
class SGOfNGDecomposeWordProblem : public Supervisor
{
public:
  SGOfNGDecomposeWordProblem( class SMSubgroup&, const class SMWord& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class SMWord& theWord;
  class SMSubgroup& theSubgroup;
  class SIC& theSIC;
  SGOfNGDecomposeWordARCer arcer;
  MirrorSubordinate  initializeSGOfNG;
  MirrorSubordinate  computeBasis;
  MirrorSubordinate  ngDecomposeWord;
  bool started;
};
```

———-

20.33.8 class SGOfNGWordContainARCer

— class SGOfNGWordContainARCer —

```
class SGOfNGWordContainARCer : public ARCer
{
public:
  SGOfNGWordContainARCer( ComputationManager& boss )
    : ARCer( boss ),theSubgroup(0)
  ~SGOfNGWordContainARCer()
  void setArguments( const SGOfNilpotentGroup&, const SetOf<Word>& );
  bool answer()
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SGOfNilpotentGroup* theSubgroup;
  bool theAnswer;
  SetOf<Word> theTestWords;
};
```

———-

20.33.9 class SGOfNGWordContain

— class SGOfNGWordContain —

```
class SGOfNGWordContain : public ComputationManager
{
public:
  SGOfNGWordContain( class SGOfNGWordContainProblem& );
  Trichotomy answer( ) const;
  void viewStructure(ostream& ostr) const { }
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class SMSubgroup& theSubgroup;
  SetOf<Word> theTestWords;
  SGOfNGWordContainARCer arcer;
  bool arcerStarted;
  Trichotomy theAnswer;
};
```

———-

20.33.10 class SGOfNGWordContainProblem

— class SGOfNGWordContainProblem —

```
class SGOfNGWordContainProblem : public Supervisor
{
public:
  SGOfNGWordContainProblem( class SMSubgroup&, const class SMWord& );
  const SMSubgroup& getSubgroup( ) const
  const SMWord& getTestWord( ) const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMSubgroup& theSubgroup;
  const class SMWord& theTestWord;
  MirrorSubordinate initPreimageSGOfNG;
  Subordinate<SGOfNGWordContainProblem, SGOfNGWordContain> sgContain;
};
```

———-

20.33.11 class SGOfNGcontainSubgroupARCer

— class SGOfNGcontainSubgroupARCer —

```
class SGOfNGcontainSubgroupARCer : public ARCer
{
public:
  SGOfNGcontainSubgroupARCer( ComputationManager& boss )
    : ARCer( boss ),theSubgroup1( 0 )
  ~SGOfNGcontainSubgroupARCer( )
  void setArguments(const SGOfNilpotentGroup& SNG1 ,
        const VectorOf<Word>& SNG2);
  bool getAnswer() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SGOfNilpotentGroup* theSubgroup1;
  VectorOf<Word> theSubgroup2;
  bool answer;
};
```

———-

20.33.12 class SGOfNGcontainSubgroupProblem

— class SGOfNGcontainSubgroupProblem —

```
class SGOfNGcontainSubgroupProblem : public Supervisor
{
public:
  SGOfNGcontainSubgroupProblem(class SMSubgroup& sg1,
                               const class SMSubgroup& sg2);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  MirrorSubordinate initPreimageSGOfNG;
  class SMSubgroup& theSubgroup1;
  const class SMSubgroup& theSubgroup2;
  SGOfNGcontainSubgroupARCer arcer;
  bool started;
};
```

———-

20.33.13 class SGOfNGequalSubgroupProblem

— class SGOfNGequalSubgroupProblem —

```
class SGOfNGequalSubgroupProblem : public Supervisor
{
public:
  SGOfNGequalSubgroupProblem(class SMSubgroup& sg1, class SMSubgroup& sg2);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  MirrorSubordinate initPreimageSGOfNG1;
  MirrorSubordinate initPreimageSGOfNG2;
  class SMSubgroup& theSubgroup1;
  class SMSubgroup& theSubgroup2;
  SGOfNGcontainSubgroupARCer arcer1;
  SGOfNGcontainSubgroupARCer arcer2;
  bool started1;
  bool started2;
};
```

———-

20.33.14 class SGOfNGindexARCer

— class SGOfNGindexARCer —

```
class SGOfNGindexARCer : public ARCer
{
public:
  SGOfNGindexARCer( ComputationManager& boss )
    : ARCer( boss ),theSNG( 0 )
  ~SGOfNGindexARCer( )
  void setArguments(const SGOfNilpotentGroup& );
  int getIndex() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SGOfNilpotentGroup* theSNG;
  int index;
};
```

———-

20.33.15 class SGOfNGindexProblem

— class SGOfNGindexProblem —

```
class SGOfNGindexProblem : public Supervisor
{
public:
  SGOfNGindexProblem(class SMSubgroup& sg);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  SGOfNGindexARCer arcer;
  MirrorSubordinate initPreimageSGOfNG;
  class SMSubgroup& theSubgroup;
  bool started;
};
```

———-

20.33.16 class SGOfNGhirschNumberProblem

— class SGOfNGhirschNumberProblem —

```
class SGOfNGhirschNumberProblem : public Supervisor
{
public:
  SGOfNGhirschNumberProblem(class SMSubgroup& sg);
```

```
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  MirrorSubordinate initializeSGOfNG;
  class SMSubgroup& theSubgroup;
  bool started;
};
```

———-

20.33.17 class SGOfNGisNormalARCer

— class SGOfNGisNormalARCer —

```
class SGOfNGisNormalARCer : public ARCer
{
public:
  SGOfNGisNormalARCer( ComputationManager& boss )
    : ARCer( boss ),theSNG( 0 )
  ~SGOfNGisNormalARCer( )
  void setArguments(const SGOfNilpotentGroup& );
  bool answer() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SGOfNilpotentGroup* theSNG;
  bool theAnswer;
};
```

———-

20.33.18 class SGOfNGisNormalProblem

— class SGOfNGisNormalProblem —

```
class SGOfNGisNormalProblem : public Supervisor
{
public:
  SGOfNGisNormalProblem(class SMSubgroup& sg);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
```

```
  SGOfNGisNormalARCer arcer;
  MirrorSubordinate initPreimageSGOfNG;
  class SMSubgroup& theSubgroup;
  bool started;
};
```

———-

20.33.19 class SGOfNGcomputeNClassARCer

— class SGOfNGcomputeNClassARCer —

```
class SGOfNGcomputeNClassARCer : public ARCer
{
public:
  SGOfNGcomputeNClassARCer( ComputationManager& boss )
    : ARCer( boss ),theSNG( 0 )
  ~SGOfNGcomputeNClassARCer( )
  void setArguments(const NilpotentGroup&,const VectorOf<Word>& );
  int getClass() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SGOfNilpotentGroup* theSNG;
  int theClass;
};
```

———-

20.33.20 class SGOfNGcomputeNClassProblem

— class SGOfNGcomputeNClassProblem —

```
class SGOfNGcomputeNClassProblem : public Supervisor
{
public:
  SGOfNGcomputeNClassProblem( class SMSubgroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMSubgroup& theSubgroup;
  class GIC& theGIC;
  SGOfNGcomputeNClassARCer arcer;
  MirrorSubordinate computeBasis;
```

```
  bool started;
};
```

———-

20.33.21 class SGOfNGPresentationARCer

— class SGOfNGPresentationARCer —

```
class SGOfNGPresentationARCer : public ARCer
{
public:
  SGOfNGPresentationARCer( ComputationManager& boss )
    : ARCer( boss ),theSubgroup( 0 ), thePresentation( 0 )
  ~SGOfNGPresentationARCer()
  void setArguments(const SGOfNilpotentGroup& presentation);
  const PresentationForSNG& getPresentation() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  PresentationForSNG* thePresentation;
  SGOfNilpotentGroup* theSubgroup;
};
```

———-

20.33.22 class SGOfNGbuildPresentationProblem

— class SGOfNGbuildPresentationProblem —

```
class SGOfNGbuildPresentationProblem : public Supervisor
{
public:
  SGOfNGbuildPresentationProblem(class SMSubgroup& subgroup);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  SGOfNGPresentationARCer arcer;
  class SMSubgroup& theSubgroup;
  MirrorSubordinate initPreimageSGOfNG;
  bool started;
};
```

———-

20.33.23 class SGOfNGnormalClosureARCer

— class SGOfNGnormalClosureARCer —

```
class SGOfNGnormalClosureARCer : public ARCer
{
public:
  SGOfNGnormalClosureARCer( ComputationManager& boss )
    : ARCer( boss ),theSubgroup( 0 )
  ~SGOfNGnormalClosureARCer( )
  void setArguments(const SGOfNilpotentGroup& presentation);
  const VectorOf<Word>& normalClosure() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SGOfNilpotentGroup* theSubgroup;
  VectorOf<Word> nClosure;
};
```

———-

20.33.24 class SGOfNGnormalClosureProblem

— class SGOfNGnormalClosureProblem —

```
class SGOfNGnormalClosureProblem : public Supervisor
{
public:
  SGOfNGnormalClosureProblem(class SMSubgroup& subgroup);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  SGOfNGnormalClosureARCer arcer;
  MirrorSubordinate initializeSGOfNG;
  bool started;
  class SMSubgroup& theSubgroup;
};
```

———-

20.33.25 class SGOfNGnormalClosureGensARCer

— class SGOfNGnormalClosureGensARCer —

```
class SGOfNGnormalClosureGensARCer : public ARCer
{
public:
  SGOfNGnormalClosureGensARCer( ComputationManager& boss )
    : ARCer( boss ),theSubgroup( 0 )
  ~SGOfNGnormalClosureGensARCer( )
  void setArguments(const SGOfNilpotentGroup& presentation);
  const VectorOf<Word>& normalClosure() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SGOfNilpotentGroup* theSubgroup;
  VectorOf<Word> nClosure;
};
```

———-

20.33.26 class SGOfNGnormalClosureGensProblem

— class SGOfNGnormalClosureGensProblem —

```
class SGOfNGnormalClosureGensProblem : public Supervisor
{
public:
  SGOfNGnormalClosureGensProblem(class SMSubgroup& subgroup);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  SGOfNGnormalClosureGensARCer arcer;
  class SMSubgroup& theSubgroup;
};
```

———-

20.34 SMApps/include/NilpotentProblems.h

— NilpotentProblems.h —

```
#include "Integer.h"
#include "Supervisor.h"
#include "ComputationManager.h"
#include "NilpotentQuotients.h"
#include "NilpotentGroup.h"
```

```
#include "List.h"
#include "ARCer.h"
```

———-

20.34.1 class NGOrderOfEltProblemARCer

— class NGOrderOfEltProblemARCer —

```
class NGOrderOfEltProblemARCer : public ARCer
{
public:
  NGOrderOfEltProblemARCer( ComputationManager& boss )
    : ARCer( boss ), theNG( 0 )
  void setArguments( const NilpotentGroup&, const Word& );
  Integer getOrder() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  const NilpotentGroup* theNG;
  Integer retValue;
  Word theWord;
};
```

———-

20.34.2 class NGorderOfEltProblem

— class NGorderOfEltProblem —

```
class NGorderOfEltProblem : public Supervisor
{
public:
  NGorderOfEltProblem( class SMWord& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMWord& theWord;
  class GIC& theGIC;
  MirrorSubordinate computeBasis;
  NGOrderOfEltProblemARCer arcer;
  bool started;
};
```

———-

20.34.3 class NGHirschNumberProblem

— class NGHirschNumberProblem —

```
class NGHirschNumberProblem : public Supervisor
{
public:
  NGHirschNumberProblem( class SMFPGroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMFPGroup& theGroup;
  class GIC& theGIC;
  MirrorSubordinate computeBasis;
};
```

———-

20.34.4 class NGcomputeLCSQuotientsProblem

— class NGcomputeLCSQuotientsProblem —

```
class NGcomputeLCSQuotientsProblem : public Supervisor
{
public:
  NGcomputeLCSQuotientsProblem( class SMFPGroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMFPGroup& theGroup;
  class GIC& theGIC;
  MirrorSubordinate computeBasis;
  MirrorSubordinate computeLCSQuotientsComp;
};
```

———-

20.34.5 class NGcomputeNClassProblem

— class NGcomputeNClassProblem —

```
class NGcomputeNClassProblem : public Supervisor
{
public:
  NGcomputeNClassProblem( class SMFPGroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMFPGroup& theGroup;
  class GIC& theGIC;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate computeBasis;
};
```

———-

20.34.6 class NGisFreeNilpotentProblem

— class NGisFreeNilpotentProblem —

```
class NGisFreeNilpotentProblem : public Supervisor
{
public:
  NGisFreeNilpotentProblem( class SMFPGroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMFPGroup& theGroup;
  class GIC& theGIC;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate computeBasis;
  Trichotomy answer;
  int freeRank, freeClass;
};
```

———-

20.34.7 class NGdecomposeWordProblem

— class NGdecomposeWordProblem —

```
class NGdecomposeWordProblem : public Supervisor
{
public:
```

```
  NGdecomposeWordProblem( class SMWord& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMWord& theWord;
  class WIC& theWIC;
  MirrorSubordinate computeBasis;
  MirrorSubordinate ngDecomposeWord;
};
```

———-

20.34.8 class NGisWordInCommutatorSGProblem

— class NGisWordInCommutatorSGProblem —

```
class NGisWordInCommutatorSGProblem : public Supervisor
{
public:
  NGisWordInCommutatorSGProblem( class SMWord& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMWord& theWord;
  class GIC& theGIC;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.34.9 class NGweightOfWordARCer

— class NGweightOfWordARCer —

```
class NGweightOfWordARCer : public ARCer
{
public:
  NGweightOfWordARCer( ComputationManager& boss )
    : ARCer( boss ),theNG(NULL)
  ~NGweightOfWordARCer()
  void setArguments( const NilpotentGroup&, const Word& );
  int getWeight()
  void runComputation( );
```

```
  void writeResults( ostream& );
  void readResults( istream& );
private:
  NilpotentGroup* theNG;
  int weight;
  Word theWord;
};
```

———-

20.34.10 class NGweightOfWordProblem

— class NGweightOfWordProblem —

```
class NGweightOfWordProblem : public Supervisor
{
public:
  NGweightOfWordProblem( class SMWord& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMWord& theWord;
  class WIC& theWIC;
  MirrorSubordinate computeBasis;
  bool started;
  int weight;
  NGweightOfWordARCer arcer;
};
```

———-

20.34.11 class NGorderOfTorsionSubgroupProblem

— class NGorderOfTorsionSubgroupProblem —

```
class NGorderOfTorsionSubgroupProblem : public Supervisor
{
public:
  NGorderOfTorsionSubgroupProblem( class SMFPGroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMFPGroup& theGroup;
```

```
  class GIC& theGIC;
  MirrorSubordinate computeBasis;
  Integer order;
};
```

———-

20.34.12 class NGbuildPresentationProblem

— class NGbuildPresentationProblem —

```
class NGbuildPresentationProblem : public Supervisor
{
public:
  NGbuildPresentationProblem(class SMFPGroup& group);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  NGPresentationARCer arcer;
  class SMFPGroup& theGroup;
  MirrorSubordinate computeBasis;
  bool started;
};
```

———-

20.34.13 class NGAutoIsIAAut

— class NGAutoIsIAAut —

```
class NGAutoIsIAAut : public Supervisor
{
public:
  NGAutoIsIAAut( class SMMap& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMMap& theMap;
  class GIC& theGIC;
  MirrorSubordinate abelianInvariants;
};
```

———-

20.34.14 class NGisCentralARCer

— class NGisCentralARCer —

```
class NGisCentralARCer : public ARCer
{
public:
  NGisCentralARCer( ComputationManager& boss )
    : ARCer( boss ),theNG( 0 ),theAnswer(dontknow)
  ~NGisCentralARCer( ) { delete theNG; }
  void setArguments(const NilpotentGroup& NG, const Word& theWord );
  Trichotomy getAnswer() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  NilpotentGroup* theNG;
  Word theWord;
  Trichotomy theAnswer;
};
```

———-

20.34.15 class NGisCentralProblem

— class NGisCentralProblem —

```
class NGisCentralProblem : public Supervisor
{
public:
   NGisCentralProblem( class SMWord& w);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  NGisCentralARCer arcer;
  MirrorSubordinate   computeBasis;
  class SMWord& theWord;
  Trichotomy answer;
  bool started;
};
```

———-

20.34.16 class NGMaximalRootARCer

— class NGMaximalRootARCer —

```
class NGMaximalRootARCer : public ARCer
{
public:
  NGMaximalRootARCer( ComputationManager& boss )
    : ARCer( boss ), NG( 0 ), thePower(0)
  ~NGMaximalRootARCer( )
  void setArguments( const NilpotentGroup& group, const Word& w);
  int getPower() const;
  Word getRoot() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  NilpotentGroup* NG;
  int thePower;
  Word theRoot;
  Word theWord;
};
```

———-

20.34.17 class NGMaximalRootProblem

— class NGMaximalRootProblem —

```
class NGMaximalRootProblem : public Supervisor
{
public:
  NGMaximalRootProblem( class SMWord& w);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  NGMaximalRootARCer arcer;
  MirrorSubordinate computeBasis;
  class SMWord& theWord;
  bool started;
};
```

———-

20.34.18 class NGIsProperPower

— class NGIsProperPower —

```
class NGIsProperPower : public Supervisor
{
public:
  NGIsProperPower( class SMWord& w);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  NGMaximalRootARCer arcer;
  MirrorSubordinate computeBasis;
  class SMWord& theWord;
  bool started;
};
```

———-

20.34.19 class NGInverseAutoARCer

— class NGInverseAutoARCer —

```
class NGInverseAutoARCer : public ARCer
{
public:
  NGInverseAutoARCer( ComputationManager& boss )
    : ARCer( boss ), NG( 0 ), result(0)
  ~NGInverseAutoARCer( )
  void setArguments( const NilpotentGroup& group, const VectorOf<Word>& image);
  const VectorOf<Word> getInverse() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  NilpotentGroup* NG;
  VectorOf<Word> theImages;
  VectorOf<Word> result;
};
```

———-

20.34.20 class NGInverseAuto

— class NGInverseAuto —

```
class NGInverseAuto  : public Supervisor
{
public:
  NGInverseAuto(const class SMHomomorphism& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class SMHomomorphism& theAuto;
  NGInverseAutoARCer arcer;
};
```

———-

20.34.21 class NGcentralizerARCer

— class NGcentralizerARCer —

```
class NGcentralizerARCer : public ARCer
{
public:
  NGcentralizerARCer( ComputationManager& boss )
    : ARCer( boss ), NG( 0 ), result(0)
  ~NGcentralizerARCer( )
  void setArguments( const NilpotentGroup& group, const Word& w);
  const VectorOf<Word>& getCentralizer() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  NilpotentGroup* NG;
  Word theWord;
  VectorOf<Word> result;
};
```

———-

20.34.22 class NGcentralizer

— class NGcentralizer —

```
class NGcentralizer  : public Supervisor
{
public:
  NGcentralizer( class SMWord& );
  void viewStructure(ostream& ostr) const;
```

```
  void takeControl( );
  void start( )
  void terminate( )
private:
  NGcentralizerARCer arcer;
  MirrorSubordinate computeBasis;
  class SMWord& theWord;
  bool started;
};
```

———-

20.34.23 class NGIsomorphismARCer

— class NGIsomorphismARCer —

```
class NGIsomorphismARCer : public ARCer
{
public:
  NGIsomorphismARCer( ComputationManager& boss )
    : ARCer( boss ),
      g1( ),
      g2( ),
      ans(dontknow)
  void setArguments( const FPGroup& gr1 , const FPGroup& gr2 ,
         int n1 , int n2 );
  Trichotomy getAnswer() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  FPGroup g1;
  FPGroup g2;
  int nc1;
  int nc2;
  Trichotomy ans;
};
```

———-

20.34.24 class NGIsomorphismProblem

— class NGIsomorphismProblem —

```
class NGIsomorphismProblem  : public ComputationManager
{
public:
```

```
  NGIsomorphismProblem( class SMFPGroup& , class SMFPGroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  NGIsomorphismARCer arcer;
  SMFPGroup& theGroup1;
  SMFPGroup& theGroup2;
  int flag;
};
```

———-

20.35 SMApps/include/NilpotentQuotients.h

— **NilpotentQuotients.h** —

```
#include "Supervisor.h"
#include "NilpotentGroup.h"
#include "ComputationManager.h"
#include "List.h"
#include "File.h"
#include "Presentation.h"
```

———-

20.35.1 class NilpotentWPARCer

— **class NilpotentWPARCer** —

```
class NilpotentWPARCer : public ARCer
{
public:
  NilpotentWPARCer( ComputationManager& boss )
    : ARCer( boss ),theGroup(0)
  ~NilpotentWPARCer( )
  void setArguments(const SetOf<Word>& setW, const NilpotentGroup& NG);
  void setArguments(const VectorOf<Word>& vecW, const NilpotentGroup& NG);
  void setArguments(const int& length, const NilpotentGroup& NG);
  void setArguments(const int& length, const NilpotentGroup& NG,
        const VectorOf<Word>& gen);
  bool getResult() const;
  Word getWord() const;
  VectorOf<int> getCommutator() const;
  void runComputation( );
```

```
  void writeResults( ostream& );
  void readResults( istream& );
private:
  void commutatorsCheck();
  void wordsCheck();
  void setGroup(const NilpotentGroup& NG);
  VectorOf<Word> theWords;
  int theCommutatorsLength;
  VectorOf<Word> generators;
  NilpotentGroup* theGroup;
  bool result;
  bool isCommutatorsChecks;
  Word retWord;
  VectorOf<int> commutator;
};
```

———-

20.35.2 class NilpotentWPInQuotients

— class NilpotentWPInQuotients —

```
class NilpotentWPInQuotients : public ComputationManager
{
public:
  NilpotentWPInQuotients(Supervisor& sup);
  void initialize(const VectorOf<Word>& vecW,const class SMFPGroup* group,
      int startClass = 2);
  void initialize(const SetOf<Word>& w,const class SMFPGroup* group,
      int startClass = 2);
  void initialize(const int& length,const class SMFPGroup* group,
      int startClass = 2);
  void initialize(const int& length,const class SMFPGroup* group,
      const VectorOf<Word>& gen,int startClass = 2);
  Trichotomy isTrivial() const
  Trichotomy isTrivial( int& nClass ) const;
  Word getWord() const;
  VectorOf<int> getCommutator() const;
  void takeControl( );
  void start( );
  void terminate( )
private:
  SetOf<Word> theWords;
  VectorOf<Word> generators;
  int theCommutatorsLength;
  bool isCommutatorsCheck;
  const SMFPGroup* theGroup;
  int currentClass;
```

```
  int upperBound;
  NilpotentWPARCer arcer;
  Trichotomy is_trivial;
  bool isStarted;
  Word retWord;
  VectorOf<int> commutator;
};
```

———-

20.35.3 class NilpotentWP

— class NilpotentWP —

```
class NilpotentWP : public ComputationManager
{
public:
  NilpotentWP( Supervisor& sup);
  void initialize(const VectorOf<Word>& vecW,const class  SMFPGroup* group);
  void initialize(const SetOf<Word>& w,const class  SMFPGroup* group);
  void initialize(const int& length,const class  SMFPGroup* group);
  void initialize(const int& length,const class  SMFPGroup* group,
      const VectorOf<Word>& gen);
  Trichotomy isTrivial() const;
  Word getWord() const;
  VectorOf<int> getCommutator() const;
  void takeControl( );
  void start( );
  void terminate( )
private:
  SetOf<Word> theWords;
  const SMFPGroup* theGroup;
  int theCommutatorsLength;
  bool isCommutatorsCheck;
  VectorOf<Word> generators;
  int theClass;
  NilpotentWPARCer arcer;
  bool isStarted;
  Trichotomy is_trivial;
  Word retWord;
  VectorOf<int> commutator;
};
```

———-

20.35.4 class NGcomputeBasisARCer

— class NGcomputeBasisARCer —

```
class NGcomputeBasisARCer : public ARCer
{
public:
  NGcomputeBasisARCer( ComputationManager& boss, bool put_in_file = false)
    : ARCer( boss ), init( 0 ),putInFile(put_in_file)
  ~NGcomputeBasisARCer( )
  void setArguments(const FPGroup& group,const int& nilpClass);
  const NilpotentGroup& getGroup() const;
  Chars getFileName() const {return structFile.getFileName();}
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  NilpotentGroup* init;
  File structFile;
  bool putInFile;
};
```

———-

20.35.5 class NGcomputeBasis

— class NGcomputeBasis —

```
class NGcomputeBasis : public ComputationManager
{
public:
  NGcomputeBasis(class GCM& gcm, bool internal = true);
  ~NGcomputeBasis( );
  void viewStructure(ostream& ostr) const
  void takeControl( );
  void start( );
  void terminate( );
private:
  GCM& theGCM;
  NGcomputeBasisARCer arcer;
  int arcsNumber;
  bool haveCD;
};
```

———-

20.35.6 class NilpotentQuotients

— class NilpotentQuotients —

```
class NilpotentQuotients : public ComputationManager
{
public:
  NilpotentQuotients(class GCM& gcm, bool internal = true);
  ~NilpotentQuotients( );
  void viewStructure(ostream& ostr) const
  void takeControl( );
  void start( );
  void terminate( );
private:
  bool increaseCurrentClass();
  GCM& theGCM;
  SMFPGroup& theGroup;
  NGcomputeBasisARCer arcer;
  int currentClass;
  int upperBound;
};
```

———-

20.35.7 class NGcomputeBasisProblem

— class NGcomputeBasisProblem —

```
class NGcomputeBasisProblem : public Supervisor
{
public:
  NGcomputeBasisProblem(class SMFPGroup& G);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( )
private:
  SMFPGroup& theGroup;
  MirrorSubordinate computeBasis;
};
```

———-

20.35.8 class NGPresentationARCer

— class NGPresentationARCer —

```
class NGPresentationARCer : public ARCer
{
public:
  NGPresentationARCer( ComputationManager& boss )
```

```
    : ARCer( boss ),theGroup( 0 ), thePresentation( 0 )
  ~NGPresentationARCer( )
  void setArguments(const NilpotentGroup& presentation);
  const PresentationForNG& getPresentation() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  PresentationForNG* thePresentation;
  NilpotentGroup* theGroup;
};
```

———-

20.35.9 class NGdecomposeWordARCer

— class NGdecomposeWordARCer —

```
class NGdecomposeWordARCer : public ARCer
{
public:
  NGdecomposeWordARCer( ComputationManager& boss )
    : ARCer( boss ),theNG(NULL)
  ~NGdecomposeWordARCer()
  void setArguments( const NilpotentGroup&, const Word& );
  PolyWord decomposition()
  void runComputation( );
   void writeResults( ostream& );
  void readResults( istream& );
private:
  NilpotentGroup* theNG;
  PolyWord retWord;
  Word theWord;
};
```

———-

20.35.10 class NGdecomposeWord

— class NGdecomposeWord —

```
class NGdecomposeWord : public ComputationManager
{
public:
  NGdecomposeWord(class WCM& wcm);
  ~NGdecomposeWord( );
  void viewStructure(ostream& ostr) const
```

```
  void takeControl( );
  void start( );
  void terminate( );
private:
  class WCM& theWCM;
  class SMWord& theWord;
  NGdecomposeWordARCer arcer;
  bool  started;;
};
```

———-

20.36 SMApps/include/NormalClosure.h

— NormalClosure.h —

```
#include "ComputationManager.h"
#include "SubgroupGraph.h"
#include "FPGroup.h"
```

———-

20.36.1 class NormalClosureARCer

— class NormalClosureARCer —

```
class NormalClosureARCer : public ARCer2
{
public:
  NormalClosureARCer( ComputationManager& boss );
  ~NormalClosureARCer( );
  void setArguments( const FPGroup&, const VectorOf<Word>& sgens );
  class DecomposeInSubgroupOfFPGroup& getDecomposer() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& )
  bool readResults2( istream& );
private:
  class DecomposeInSubgroupOfFPGroup* theDecomposer;
  class DecomposeInSubgroupOfFPGroup* tempDecomposer;
};
```

———-

20.36.2 class NormalClosure

— class NormalClosure —

```
class NormalClosure : public ComputationManager
{
public:
  NormalClosure(class GCM& gcm);
  NormalClosure(class SCM& scm);
  Trichotomy isTrivial(const Word& w) const;
  DecomposeInSubgroupOfFPGroup& getDecomposer( );
  SubgroupGraph getSubgroupGraph( ) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  void putHaveWordDecomposer( const DecomposeInSubgroupOfFPGroup& D );
  void putHaveCompleteCayleyGraph( const SubgroupGraph& S );
  NormalClosureARCer arcer;
  class ComputationManager& theCM;
  class SMFPGroup& theGroup;
  class SMObject& theObject;
};
```

———-

20.37 SMApps/include/OneRelatorProblems.h

— OneRelatorProblems.h —

```
#include "fastProblems.h"
```

———-

20.37.1 class ORFindHNNPresentation

— class ORFindHNNPresentation —

```
class ORFindHNNPresentation : public FastComputation
{
public:
  ORFindHNNPresentation(const SMFPGroup& G)
    : theGroup( G ), numOfGens(0), stableGen(1), accompGen(1)
  ORFindHNNPresentation(const SMFPGroup& G, const Generator& stable)
    : theGroup( G ), numOfGens(1), stableGen(stable), accompGen(1)
  ORFindHNNPresentation(const SMFPGroup& G, const Generator& stable,
```

```
      const Generator& accomp)
    : theGroup( G ), numOfGens(2), stableGen(stable), accompGen(accomp)
  void takeControl( );
private:
  const SMFPGroup& theGroup;
  int numOfGens;
  Generator stableGen;
  Generator accompGen;
};
```

———-

20.37.2 class ORGroup

— class ORGroup —

```
class ORGroup_IsMagnusSubgroup : public FastComputation
{
public:
  ORGroup_IsMagnusSubgroup( const SMSubgroup& S )
    : subgroup( S )
  void takeControl( );
private:
  const SMSubgroup& subgroup;
};
```

———-

20.37.3 class ORExtendedWordProblemARCer

— class ORExtendedWordProblemARCer —

```
class ORExtendedWordProblemARCer : public ARCer
{
public:
  ORExtendedWordProblemARCer( ComputationManager& );
  void setArguments( const Word& relator, const Word& testWord,
         const VectorOf<Word>& target );
  bool getRetValue() const
  Word wordInBasis( ) const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  void printWord(const Word& w, ostream& ostr);
  Word theRelator;
  Word theTestWord;
```

```
  VectorOf<Word> theTarget;
  Trichotomy answer;
  Word theWordInBasis;
  bool retValue;
};
```

———-

20.37.4 class ORExtendedWordProblemCM

— class ORExtendedWordProblemCM —

```
class ORExtendedWordProblemCM : public ComputationManager
{
public:
  ORExtendedWordProblemCM( class ORExtendedWordProblem& PO );
  Trichotomy isExpressed( ) const
  Word wordInBasis( ) const
  void takeControl( );
  void start( )
  void terminate( );
private:
  const Word theRelator;
  const Word theTestWord;
  const VectorOf<Word> theTarget;
  Trichotomy itIsExpressed;
  Word theWordInBasis;
  bool bStarted;
  ORExtendedWordProblemARCer arcer;
};
```

———-

20.37.5 class ORExtendedWordProblem

— class ORExtendedWordProblem —

```
class ORExtendedWordProblem : public Supervisor
{
public:
  ORExtendedWordProblem( class SMSubgroup& target, class SMWord& w );
  Trichotomy answer( ) const
  Word wordInBasis( ) const
  const SMWord& getTestWord( ) const
  const SMSubgroup& getTarget( ) const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
```

```
  void start( )
  void terminate( )
private:
  SMWord& theWord;
  SMSubgroup& theTarget;
  Trichotomy theAnswer;
  Word theWordInBasis;
  Subordinate<ORExtendedWordProblem, ORExtendedWordProblemCM> orwp;
};
```

———-

20.37.6 class ORIsMagnusSubgroup

— class ORIsMagnusSubgroup —

```
class ORIsMagnusSubgroup : public FastComputation
{
public:
  ORIsMagnusSubgroup( const SMSubgroup& subgroup )
    : theSubgroup( subgroup )
  void takeControl( );
private:
  const SMSubgroup& theSubgroup;
};
```

———-

20.38 SMApps/include/OrderOfElt.h

— OrderOfElt.h —

```
#include "Supervisor.h"
#include "SetOfWordsChecker.h"
#include "NilpotentProblems.h"
```

———-

20.38.1 class OrderOfEltInQuotients

— class OrderOfEltInQuotients —

```
class OrderOfEltInQuotients : public ComputationManager
{
public:
```

```
  OrderOfEltInQuotients(class OrderOfElt& sup);
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SMWord& theWord;
  const SMFPGroup& theGroup;
  int currentClass;
  int upperBound;
  NGOrderOfEltProblemARCer arcer;
  bool isStarted;
};
```

———-

20.38.2 class OrderOfEltARCer

— class OrderOfEltARCer —

```
class OrderOfEltARCer : public ARCer
{
public:
  OrderOfEltARCer( ComputationManager& boss,
       const class SMFPGroup& group,
       const Word& word );
  int getRetValue()
  int getCurrentPower() const
  void setCurrentPower(const int& p)
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  const class SMFPGroup& theGroup;
  Word theWord;
  Word current;
  int retValue;
  int power;
};
```

———-

20.38.3 class OrderOfElt

— class OrderOfElt —

```
class OrderOfElt : public Supervisor
{
```

```
public:
  OrderOfElt( class SMWord& );
  SMWord& getWord()
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  int getOrder( );
  SMWord& theWord;
  OrderOfEltARCer theArcer;
  bool triedAbelianization;
  Chars explanation;
  MirrorSubordinate normalClosure;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate nilpotentQuotients;
  Subordinate<OrderOfElt,OrderOfEltInQuotients> orderOfEltInQuotients;
};
```

———-

20.39 SMApps/include/OrderProblem.h

— **OrderProblem.h** —

```
#include "Supervisor.h"
#include "CosetEnumerator.h"
#include "ToddCoxeter.h"
#include "HToddCoxeter.h"
#include "SMFPGroup.h"
```

———-

20.39.1 class OrderProblem

— **class OrderProblem** —

```
class OrderProblem : public Supervisor
{
public:
  OrderProblem(SMFPGroup& G);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
```

```
  void terminate( )
private:
  SMFPGroup& theGroup;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate theToddCoxeter;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate computeBasis;
  MirrorSubordinate ghToddCoxeter;
};
```

———-

20.40 SMApps/include/PreAbelianRepProblem.h

— **PreAbelianRepProblem.h** —

```
#include "SMFPGroup.h"
#include "File.h"
```

———-

20.40.1 class PreAbelianRepProblem

— **class PreAbelianRepProblem** —

```
class PreAbelianRepProblem : public FastComputation
{
public:
  PreAbelianRepProblem(class SMFPGroup& theGroup)
  void takeControl( );
private:
  class FPGroup G;
  File f;
  int *a, c, m, n;
  VectorOf<Word> g, x, y, z;
  int findmin(int);
  int plus(int, int, int, char);
  void show(void);
  void swap(int, int, char);
  Word pow(const Word &, int);
};
```

———-

20.41 SMApps/include/QuadEquationSolver.h

— **QuadEquationSolver.h** —

```
#include "Supervisor.h"
#include "SMEquation.h"
#include "File.h"
#include "LogWatcher.h"
#include "fastProblems.h"
```

———-

20.41.1 class EquationSolverARCer

— **class EquationSolverARCer** —

```
class EquationSolverARCer : public ARCer
{
public:
  EquationSolverARCer( ComputationManager& boss ) : ARCer( boss )
  void setArguments( class SMEquation& eq,
         const Chars& aBasicFileName, const Chars& aStabFileName,
         bool workOnBasicSolutions, bool workOnStabilizer);
  int numberOfBasicSolutions( ) const;
  int numberOfRegStabGenerators( ) const;
  void runComputation( );
  void writeResults( ostream& ostr );
  void readResults( istream& istr );
private:
  FreeGroup F;
  Word word;
  int numberOfVariables;
  bool workOnBasicSolutions;
  bool workOnStabilizer;
  Chars basicFileName;
  Chars stabFileName;
  int numOfBasicSolutions;
  int numOfRegStabGenerators;
};
```

———-

20.41.2 class EquationSolver

— **class EquationSolver** —

```
class EquationSolver : public ComputationManager
{
public:
  EquationSolver( class EquationProblem& equationProblem );
  EquationSolver( class QuickEquationProblem& equationProblem );
  void takeControl( );
  void start( )
  void terminate( );
private:
  void updateBasicSolutions( );
  void updateRegStabGenerators( );
  SMEquation& equation;
  Chars basicFileName;
  Chars stabFileName;
  LogFileWatcher watchBasicSolutions;
  LogFileWatcher watchRegStabGenerators;
  bool workOnBasicSolutions;
  bool workOnStabilizer;
  ListOf<Endomorphism> foundBasicSolutions;
  ListOf<Automorphism> foundRegStabGenerators;
  EquationSolverARCer arcer;
};
```

———-

20.41.3 class EquationRandomSolutionARCer

— class EquationRandomSolutionARCer —

```
class EquationRandomSolutionARCer : public ARCer
{
public:
  EquationRandomSolutionARCer( ComputationManager& boss ) :
    ARCer( boss ), equation(0)
  void setArguments( class SMEquation& eq, const Chars& aRandomFileName,
         const Chars& aBasicFileName, const Chars& aStabFileName );
  void runComputation( );
  void writeResults( ostream& ostr )
  void readResults( istream& istr )
private:
  class SMEquation *equation;
  Chars randomFileName;
  Chars basicFileName;
  Chars stabFileName;
};
```

———-

20.41.4 class EquationRandomSolutions

— class EquationRandomSolutions —

```
class EquationRandomSolutions : public ComputationManager
{
public:
  EquationRandomSolutions( class EquationProblem& ep );
  void takeControl( );
  void start( )
  void terminate( )
private:
  EquationRandomSolutionARCer arcer;
};
```

———-

20.41.5 class EquationProblem

— class EquationProblem —

```
class EquationProblem : public Supervisor
{
public:
  EquationProblem( class SMEquation& w );
  SMEquation& getEquation() const
  Chars getBasicSolutionsFileName( ) const
  Chars getRegStabGeneratorsFileName( ) const
  Chars getRandomSolutionsFileName( ) const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMEquation& equation;
  bool linksSent;
  Chars basicSolutionsFileName;
  Chars regStabGeneratorsFileName;
  Chars randomSolutionsFileName;
  Subordinate<EquationProblem, EquationSolver> basicSubordinate;
  Subordinate<EquationProblem, EquationRandomSolutions> randomSubordinate;
};
```

———-

20.41.6 class QuickEquationProblem

— class QuickEquationProblem —

```
class QuickEquationProblem : public Supervisor
{
public:
  QuickEquationProblem( class SMEquation& w );
  SMEquation& getEquation() const
  Chars getBasicSolutionsFileName( ) const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMEquation& equation;
  bool linkSent;
  Chars basicSolutionsFileName;
  Subordinate<QuickEquationProblem, EquationSolver> basicSubordinate;
};

  QEquationInFreeBasicSolutions(const class SMEquation& e) : equation( e )
  void takeControl( );
private:
  const class SMEquation& equation;
};
```

———-

20.41.7 class QEquationInFreeRegStabGenerators

— class QEquationInFreeRegStabGenerators —

```
class QEquationInFreeRegStabGenerators : public FastComputation
{
public:
  QEquationInFreeRegStabGenerators(const class SMEquation& e) : equation( e )
  void takeControl( );
private:
  const class SMEquation& equation;
};
```

———-

20.42 SMApps/include/RankOfElt.h

— RankOfElt.h —

```
#include "FreeGroup.h"
#include "Supervisor.h"
#include "RankOfSubgroup.h"
```

———-

20.42.1 class RankOfElement

— class RankOfElement —

```
class RankOfElement : public ComputationManager
{
public:
  RankOfElement(class RankOfElementProblem& problemObject );
  ~RankOfElement( );
  Trichotomy getAnswer() const
  Chars getFileName( )
  void takeControl( );
  void start( )
  void terminate( )
private:
  VectorOf<Word> constructVector( const Word& word );
  const class SMWord& theWord;
  FreeGroup theGroup;
  RankOfSubgroupARCer theARCer;
};
```

———-

20.42.2 class RankOfElementProblem

— class RankOfElementProblem —

```
class RankOfElementProblem : public Supervisor
{
public:
  RankOfElementProblem(const class SMWord& w);
  const class SMWord& getWord( ) const;
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const SMWord& theWord;
  bool linkHasBeenSent;
  Subordinate<RankOfElementProblem, RankOfElement> rankOfElement;
};
```

———-

20.43 SMApps/include/RankOfSubgroup.h

— RankOfSubgroup.h —

```
#include "TurnerProperSubgroupEnumerator.h"

#include "Supervisor.h"
#include "SMWord.h"
#include "File.h"
```

———-

20.43.1 class RankOfSubgroupARCer

— class RankOfSubgroupARCer —

```
class RankOfSubgroupARCer : public ARCer
{
public:
  RankOfSubgroupARCer( ComputationManager& boss,
      const class SMFPGroup& group,
      const SGofFreeGroup& sg );
  ~RankOfSubgroupARCer();
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
  Chars getFileName( ) const
  Trichotomy getAnswer() const
private:
  bool isAFreeFactorIn( SGofFreeGroup& g ) const;
  const class SMFPGroup& theGroup;
  const SGofFreeGroup theSubgroup;
  ProperSubgroupEnumerator* PSE;
  Trichotomy answer;
  void lookingup();
  void printSG( ostream& file, const SGofFreeGroup& subgroup );
};
```

———-

20.43.2 class RankOfSubgroup

— class RankOfSubgroup —

```
class RankOfSubgroup : public ComputationManager
{
public:
  RankOfSubgroup(class RankOfSubgroupProblem& problemObject );
  ~RankOfSubgroup( );
  Trichotomy getAnswer() const
  Chars getFileName( )
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class SMSubgroup& theSubgroup;
  FreeGroup theGroup;
  RankOfSubgroupARCer theARCer;
};
```

———-

20.43.3 class RankOfSubgroupProblem

— **class RankOfSubgroupProblem** —

```
class RankOfSubgroupProblem : public Supervisor
{
public:
  RankOfSubgroupProblem(const class SMSubgroup& sg);
  const class SMSubgroup& getSubgroup( ) const;
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const SMSubgroup& theSubgroup;
  bool linkHasBeenSent;
  Subordinate<RankOfSubgroupProblem, RankOfSubgroup> rankOfSubgroup;
};
```

———-

20.44 SMApps/include/Rewrites.h

— **Rewrites.h** —

```
#include "FPGroup.h"
#include "ComputationManager.h"
#include "SMWord.h"
```

```
#include "Word.h"
#include "OutMessages.h"
#include "ARCer.h"
```

———-

20.44.1 class RewritesARCer

— class RewritesARCer —

```
class RewritesARCer : public ARCer
{
public:
  RewritesARCer( ComputationManager& boss )
    : ARCer( boss ),
      retValue(),
      theWord(),
      G(),
      order(0),
      file()
  void setArguments(const FPGroup&,const Word&,int ord);
  ~RewritesARCer()
  Chars getRetValue() const
  Chars getFileName() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  Chars retValue;
  Word theWord;
  int order;
  FPGroup G;
  File file;
};
```

———-

20.44.2 class CommutatorRewriteProblem

— class CommutatorRewriteProblem —

```
class CommutatorRewriteProblem : public ComputationManager
{
public:
  CommutatorRewriteProblem(SMWord& w);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
```

```
  void start( )
  void terminate( )
private:
  SMWord& theWord;
  RewritesARCer arcer;
};
```

———-

20.44.3 class SquareRewriteProblem

— class SquareRewriteProblem —

```
class SquareRewriteProblem : public ComputationManager
{
public:
  SquareRewriteProblem(SMWord& w);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMWord& theWord;
  RewritesARCer arcer;
};
```

———-

20.45 SMApps/include/SetOfWordsChecker.h

— SetOfWordsChecker.h —

```
#include "SMFPGroup.h"
#include "Word.h"
```

———-

20.45.1 class SetOfWordsChecker

— class SetOfWordsChecker —

```
class SetOfWordsChecker
{
public:
  SetOfWordsChecker( const SetOf<Word>&, const class SMFPGroup& );
```

```
  SetOfWordsChecker( const VectorOf<Word>&, const class SMFPGroup& );
  SetOfWordsChecker( const class SMFPGroup& );
  Trichotomy isTrivial( );
  Chars getExplanation( ) const
  GIC::AlgorithmID getAlgorithm( ) const
  void replaceTheSet( const VectorOf<Word>& V);
  void replaceTheSet( const SetOf<Word>& );
  void enablePutDetailsToFile( );
  void disablePutDetailsToFile( );
  bool haveDetails( ) const;
  Chars getDecompositionFileName( ) const;
  Chars getDehnTransformationFileName( ) const;
private:
  void init( );
  VectorOf<Word> theWords;
  FPGroup G;
  const class GIC& gic;
  class GCM& gcm;
  bool triedAbelianization;
  GIC::AlgorithmID solutionAlgorithm;
  Chars explanation;
  VectorOf<bool> theTrivialWords;
  bool keepDetails;
  DetailedReport dehnsDetails;
  DetailedReport wordsDecomposition;
};

inline void SetOfWordsChecker::enablePutDetailsToFile( )
inline void SetOfWordsChecker::disablePutDetailsToFile( )
inline bool SetOfWordsChecker::haveDetails ( ) const
inline Chars SetOfWordsChecker::getDecompositionFileName( ) const
inline Chars SetOfWordsChecker::getDehnTransformationFileName( ) const
```

———-

20.46 SMApps/include/SGNilpotentQuotients.h

— SGNilpotentQuotients.h —

```
#include "Supervisor.h"
#include "NilpotentGroup.h"
#include "ComputationManager.h"
```

———-

20.46.1 class SCPNilpotentARCer

— class SCPNilpotentARCer —

```
class SCPNilpotentARCer : public ARCer
{
public:
  SCPNilpotentARCer( ComputationManager& boss );
  ~SCPNilpotentARCer( );
  void setArguments( const class SGOfNilpotentGroup& subgroup,
         const SetOf<Word>& testWords );
  Trichotomy contains() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SetOf<Word> theTestWords;
  class SGOfNilpotentGroup* theSubgroup;
  Trichotomy theAnswer;
};
```

———-

20.46.2 class SCPinNilpotentQuotients

— class SCPinNilpotentQuotients —

```
class SCPinNilpotentQuotients : public ComputationManager
{
public:
  SCPinNilpotentQuotients( Supervisor& boss );
  void initialize( const SetOf<Word>& testWords,
       const class SMSubgroup* subgroup,
       int startClass = 2);
  Trichotomy contains( int& nClass ) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SetOf<Word> theTestWords;
  const SMSubgroup* theSubgroup;
  Trichotomy theAnswer;
  int currentClass;
  int upperBound;
  SCPNilpotentARCer arcer;
  bool arcerStarted;
};
```

———-

20.46.3 class SGofNGinitPreimageARCer

— class SGofNGinitPreimageARCer —

```
class SGofNGinitPreimageARCer : public ARCer
{
public:
  SGofNGinitPreimageARCer( ComputationManager& boss );
  ~SGofNGinitPreimageARCer( );
  void setArguments( const NilpotentGroup& NG, const VectorOf<Word>& gens );
  const SGOfNilpotentGroup& getSubgroup() const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  SGOfNilpotentGroup* theSubgroup;
};
```

———-

20.46.4 class SGNilpotentQuotients

— class SGNilpotentQuotients —

```
class SGNilpotentQuotients : public ComputationManager
{
public:
  SGNilpotentQuotients(class SCM& scm, bool internal = true);
  ~SGNilpotentQuotients( );
  void viewStructure(ostream& ostr) const
  void takeControl( );
  void start( );
  void terminate( );
private:
  bool increaseCurrentClass();
  void setNewSubgroup( );
  SCM& theSCM;
  SMSubgroup& theSubgroup;
  SGofNGinitPreimageARCer arcer;
  int currentClass;
  int upperBound;
};

int getNilpotencyUpperBound( const SMFPGroup& G );
```

———-

20.47 SMApps/include/SMMagnusBreakdown.h

— SMMagnusBreakdown.h —

```
#include "SMFPGroup.h"
```

————-

20.47.1 class SMMagnusBreakdown

— class SMMagnusBreakdown —

```
class SMMagnusBreakdown : public SMObject
{
public:
  SMMagnusBreakdown( const SMFPGroup& G );
  static const char* type( )
  const IconID iconID( ) const
  const char* typeID( ) const
  void viewStructure(ostream& ostr) const
  void printProperties(ostream& ostr) const
  void printDefinition(ostream& ostr) const
  bool displayInFE( ) const
protected:
  enum TagMessage { ACCEPT };
  void readMessage(istream&);
private:
  const SMFPGroup& theGroup;
};
```

————-

20.48 SMApps/include/SubgroupContainmentProblem.h

— SubgroupContainmentProblem.h —

```
#include "ComputationManager.h"
#include "Supervisor.h"
#include "SGNilpotentQuotients.h"
#include "ToddCoxeter.h"
#include "AP-fixups.h"
```

————-

20.48.1 class SubgroupContainment

— class SubgroupContainment —

```
class SubgroupContainment : public ComputationManager
{
public:
  SubgroupContainment( class SubgroupContainmentProblem& );
  SubgroupContainment( class IsSGNormal& );
  Trichotomy contains() const;
  bool haveDetails( ) const;
  Chars getDecompositionFileName( ) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SetOf<Word> reduceTestingSet( const class DecomposeInSubgroupOfFPGroup& D,
        SetOf<Word>& testingSet );
  const SMSubgroup& theSubgroup;
  VectorOf<Word> theTestWords;
  SetOf<Word> wordsToCheck;
  Trichotomy theAnswer;
  DetailedReport wordsDecomposition;
};
```

———-

20.48.2 class SubgroupContainmentProblem

— class SubgroupContainmentProblem —

```
class SubgroupContainmentProblem : public Supervisor
{
public:
  SubgroupContainmentProblem( class SMSubgroup& subgroup, class SMWord& w );
  const SMSubgroup& getSubgroup() const
  const SMWord& getTestWord( ) const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  const SMSubgroup& theSubgroup;
  const SMWord& theTestWord;
  Trichotomy isWordInSubgroup;
  Chars explanation;
  Subordinate<SubgroupContainmentProblem,SubgroupContainment> subgroupContain;
  MirrorSubordinate normalClosure;
```

```
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate sgNilpotentQuotients;
  Subordinate<SubgroupContainmentProblem, SCPinNilpotentQuotients>
    scpInNilpotentQuotients;
  MirrorSubordinate toddCoxeter;
  Subordinate<SubgroupContainmentProblem, WordRepresentative> wordRepCM;
};
```

———-

20.48.3 class IsSGNormal

— **class IsSGNormal** —

```
class IsSGNormal : public Supervisor
{
public:
  IsSGNormal( class SMSubgroup& subgroup );
  const SMSubgroup& getSubgroup() const
  SetOf<Word> getTestWords( ) const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( );
  void terminate( );
private:
  SetOf<Word> makeTestWords( const class SMSubgroup& H ) const;
  const SMSubgroup& theSubgroup;
  SetOf<Word> theTestWords;
  Trichotomy isNormal;
  Chars explanation;
  Subordinate<IsSGNormal,SubgroupContainment>  subgroupContain;
  MirrorSubordinate normalClosure;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate sgNilpotentQuotients;
  Subordinate<IsSGNormal, SCPinNilpotentQuotients>  scpInNilpotentQuotients;
  MirrorSubordinate toddCoxeter;
  Subordinate<IsSGNormal, WordRepresentative> wordRepCM;
};

inline bool SubgroupContainment::haveDetails ( ) const
inline Chars SubgroupContainment::getDecompositionFileName( ) const
```

———-

20.49 SMApps/include/SubgroupProblems.h

— **SubgroupProblems.h** —

```
#include "Supervisor.h"
#include "SetOfWordsChecker.h"
#include "CommutatorsChecker.h"
#include "NilpotentProblems.h"
#include "ToddCoxeter.h"
#include "Subgroup.h"
#include "GeneticProblems.h"
```

———-

20.49.1 class IsSGAbelian

— class IsSGAbelian —

```
class IsSGAbelian : public Supervisor
{
public:
  IsSGAbelian( class SMSubgroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  SetOfWordsChecker theChecker;
  MirrorSubordinate normalClosure;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate nilpotentQuotients;
  Subordinate<IsSGAbelian,NilpotentWPInQuotients> nilpotentWPInQuotients;
  MirrorSubordinate computeBasis;
  Subordinate<IsSGAbelian,NilpotentWP> nilpotentWP;
  Subordinate<IsSGAbelian, GeneticWPCM> genetic;
};
```

———-

20.49.2 class IsSGCentral

— class IsSGCentral —

```
class IsSGCentral : public Supervisor
{
public:
  IsSGCentral( class SMSubgroup& );
  void viewStructure(ostream& ostr) const;
```

```
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  SetOfWordsChecker theChecker;
  MirrorSubordinate normalClosure;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate nilpotentQuotients;
  Subordinate<IsSGCentral,NilpotentWPInQuotients> nilpotentWPInQuotients;
  MirrorSubordinate computeBasis;
  Subordinate<IsSGCentral,NilpotentWP> nilpotentWP;
  Subordinate<IsSGCentral, GeneticWPCM> genetic;
};
```

———-

20.49.3 class IsSGNilpotent

— class IsSGNilpotent —

```
class IsSGNilpotent : public Supervisor
{
public:
  IsSGNilpotent( const class SMSubgroup& S, int nilpClass = 2 );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  const SMSubgroup& theSubgroup;
  int theClass;
  CommutatorsChecker theChecker;
  CommutatorsCheckerARCer theArcer;
  MirrorSubordinate normalClosure;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate nilpotentQuotients;
  Subordinate<IsSGNilpotent,NilpotentWPInQuotients> nilpotentWPInQuotients;
};
```

———-

20.49.4 class IsSGTrivial

— class IsSGTrivial —

```
class IsSGTrivial : public Supervisor
{
public:
  IsSGTrivial(  class SMSubgroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  SetOfWordsChecker theChecker;
  MirrorSubordinate normalClosure;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate nilpotentQuotients;
  Subordinate<IsSGTrivial,NilpotentWPInQuotients> nilpotentWPInQuotients;
  MirrorSubordinate computeBasis;
  Subordinate<IsSGTrivial,NilpotentWP> nilpotentWP;
  Subordinate<IsSGTrivial, GeneticWPCM> genetic;
};
```

———-

20.49.5 class SGIndexProblem

— class SGIndexProblem —

```
class SGIndexProblem : public Supervisor
{
public:
  SGIndexProblem(SMSubgroup& subgroup);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMSubgroup& theSubgroup;
  MirrorSubordinate sgToddCoxeter;
  MirrorSubordinate sgGHToddCoxeter;
};
```

———-

20.49.6 class ApproxMethodARCer

— class ApproxMethodARCer —

```
class ApproxMethodARCer : public ARCer
{
 public:
  ApproxMethodARCer( ComputationManager& boss )
    : ARCer( boss ) , sg( FPGroup() ) , tmpOutput( )
  void setArguments( const FPGroup& g , const VectorOf<Word>& v );
  FPGroup getRetValue( );
  Chars tmpOutputFilename();
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  Subgroup sg;
  File tmpOutput;
  FPGroup result;
};
```

———-

20.49.7 class ApproxMethod

— class ApproxMethod —

```
class ApproxMethod : public ComputationManager
{
public:
  ApproxMethod( class SGPresentationProblem& sgp );
  FPGroup getResult();
  Chars tmpOutputFilename();
  void takeControl( );
  void start( );
  void terminate( );
private:
  SMSubgroup& theSubgroup;
  FPGroup result;
  ApproxMethodARCer arcer;
};
```

———-

20.49.8 class TCMethodARCer

— class TCMethodARCer —

```
class TCMethodARCer : public ARCer
{
 public:
  TCMethodARCer( ComputationManager& boss )
    : ARCer( boss ) , sg( FPGroup() ) ,
      pr( ) , tmpOutput( )
  void setArguments( const FPGroup& g , const VectorOf<Word>& v ,
         const PermutationRepresentation& prep );
  FPGroup getRetValue( );
  Chars tmpOutputFilename();
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  Subgroup sg;
  PermutationRepresentation pr;
  File tmpOutput;
  FPGroup result;
};
```

———-

20.49.9 class TCMethod

— class TCMethod —

```
class TCMethod : public ComputationManager
{
public:
  TCMethod( class SGPresentationProblem& sgp );
  FPGroup getResult();
  Chars tmpOutputFilename();
  void takeControl( );
  void start( );
  void terminate( );
private:
  SMSubgroup& theSubgroup;
  Bool init;
  FPGroup result;
  TCMethodARCer arcer;
};
```

———-

20.49.10 class SGPresentationProblem

— class SGPresentationProblem —

```
class SGPresentationProblem : public Supervisor
{
public:
  SGPresentationProblem(SMSubgroup& s);
  void viewStructure(ostream& ostr) const;
  SMSubgroup& getTestSubgroup();
  void takeControl( );
  void start( );
  void terminate( );
private:
  SMSubgroup& theSubgroup;
  Bool done;
  Bool term;
  Subordinate<SGPresentationProblem,ApproxMethod> am;
  Subordinate<SGPresentationProblem,TCMethod> tcm;
  MirrorSubordinate sgToddCoxeter;
};
```

———-

20.49.11 class RewriteWordARCer

— class RewriteWordARCer —

```
class RewriteWordARCer : public ARCer
{
public:
  RewriteWordARCer( ComputationManager& boss )
    : ARCer( boss ),
      retValue(),
      theWord(),
      F(),
      rank( 0 ),
      tmpOutput(),
      pr(NULL)
  void setArguments(const PermutationRepresentation&,
        const FreeGroup&,const Word&);
  Chars tmpOutputFilename();
  int getRank();
  Word getRepresentative();
  Word rewrite(const PermutationRepresentation&,const VectorOf<Word>&,
         const FreeGroup& F,const Word&,bool);
  Word getRetValue() const{ return retValue; }
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  Word retValue;
```

```
  Word theWord;
  FreeGroup F;
  int rank;
  File tmpOutput;
  PermutationRepresentation* pr;
};
```

———-

20.49.12 class SGRewriteWordProblem

— class SGRewriteWordProblem —

```
class SGRewriteWordProblem : public Supervisor
{
public:
  SGRewriteWordProblem(SMSubgroup& s,SMWord& w);
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMWord& theWord;
  SMSubgroup& theSubgroup;
  bool init;
  RewriteWordARCer arcer;
  MirrorSubordinate sgToddCoxeter;
};
```

———-

20.50 SMApps/include/ToddCoxeter.h

— ToddCoxeter.h —

```
#include "Supervisor.h"
#include "CosetEnumerator.h"
#include "File.h"
```

———-

20.50.1 class ToddCoxeterARCer

— class ToddCoxeterARCer —

```
class ToddCoxeterARCer : public ARCer
{
public:
  ToddCoxeterARCer( ComputationManager& boss,const FPGroup& group )
    : ARCer( boss ), enumerator(group),permutationRepresentation(NULL),
    theGroup(group)
  ToddCoxeterARCer( ComputationManager& boss,const FPGroup& group,
       const VectorOf<Word>& subgroup)
    : ARCer( boss ), permutationRepresentation(NULL),
    enumerator(group,subgroup), theGroup(group)
  int getRetValue() const
  const PermutationRepresentation& getTransversal()const;
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  CosetEnumerator enumerator;
  FPGroup theGroup;
  int retValue;
  PermutationRepresentation* permutationRepresentation;
};
```

———-

20.50.2 class ToddCoxeter

— class ToddCoxeter —

```
class ToddCoxeter : public ComputationManager
{
public:
  ToddCoxeter(class GCM& gcm);
  void takeControl( );
  void start( )
  void terminate( )
private:
  class GCM& theGCM;
  class SMFPGroup& theGroup;
  ToddCoxeterARCer* arcer;
};
```

———-

20.50.3 class SGIndexToddCoxeter

— class SGIndexToddCoxeter —

```
class SGIndexToddCoxeter : public ComputationManager
{
public:
  SGIndexToddCoxeter(class SCM& scm);
  void takeControl( );
  void start( )
  void terminate( )
private:
  class SCM& theSCM;
  class SMSubgroup& theSubgroup;
  ToddCoxeterARCer* arcer;
};
```

———-

20.50.4 class SchreierTransversal

— class SchreierTransversal —

```
class SchreierTransversal : public Supervisor
{
public:
  SchreierTransversal(class SMFPGroup& );
  SchreierTransversal(class SMSubgroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  bool haveSchreierTransvl()const;
  const VectorOf<Word>& getRepresentatives()const;
  bool isInfinite()const;
  void start( )
  void terminate( )
private:
  class AlgebraicObject& theObject;
  class SMFPGroup& theGroup;
  MirrorSubordinate theToddCoxeter;
};
```

———-

20.50.5 class PermutationRepProblem

— class PermutationRepProblem —

```
class PermutationRepProblem : public Supervisor
{
public:
  PermutationRepProblem(class SMFPGroup& );
```

```
  PermutationRepProblem(class SMSubgroup& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  bool haveSchreierTransvl()const;
  bool isInfinite()const;
  const class PermutationRepresentation& getPermutations()const;
  void start( )
  void terminate( )
private:
  class AlgebraicObject& theObject;
  class SMFPGroup& theGroup;
  MirrorSubordinate theToddCoxeter;
};
```

———-

20.50.6 class WordRepresentativeARCer

— class WordRepresentativeARCer —

```
class WordRepresentativeARCer : public ARCer
{
public:
  WordRepresentativeARCer( ComputationManager& boss )
    : ARCer( boss ), permutationRepresentation(NULL)
  void setArguments(const PermutationRepresentation&,const Word&);
  Word getRetValue() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  Word retValue;
  Word theWord;
  PermutationRepresentation* permutationRepresentation;
};
```

———-

20.50.7 class WordRepresentative

— class WordRepresentative —

```
class WordRepresentative : public ComputationManager
{
public:
  WordRepresentative( class WordRepresentativeProblem& boss );
  WordRepresentative( class SubgroupContainmentProblem& boss );
```

```
  WordRepresentative( class IsSGNormal& boss );
  bool haveSchreierTransvl( ) const;
  bool isInfinite( ) const;
  const class PermutationRepresentation& getPermutations( ) const;
  VectorOf<bool> foundWordsReps( ) const
  VectorOf<Word> getWordsReps( ) const
  void takeControl( );
  void start( )
  void terminate( )
private:
  const class AlgebraicObject& theParent;
  const class SMFPGroup& theGroup;
  VectorOf<Word> theTestWords;
  VectorOf<Word> theWordsReps;
  VectorOf<bool> theWordsRepsFound;
  int checkedWord;
  WordRepresentativeARCer arcer;
  bool arcerStarted;
};
```

———-

20.50.8 class WordRepresentativeProblem

— class WordRepresentativeProblem —

```
class WordRepresentativeProblem : public Supervisor
{
public:
  WordRepresentativeProblem(class SMWord& );
  WordRepresentativeProblem(class SMSubgroup&,class SMWord& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  bool haveSchreierTransvl()const;
  bool isInfinite()const;
  const class AlgebraicObject& getParent( ) const
  const class SMFPGroup& getGroup() const
  const class SMWord& getWord( ) const
  void start( )
  void terminate( )
private:
  class AlgebraicObject& theParent;
  class SMFPGroup& theGroup;
  class SMWord& theWord;
  MirrorSubordinate theToddCoxeter;
  Subordinate<WordRepresentativeProblem,WordRepresentative> theWordRepCM;
};
```

———-

20.51 SMApps/include/TTProblem.h

— TTProblem.h —

```
#include "Supervisor.h"
#include "SMFPGroup.h"
#include "TTP.h"
```

———-

20.51.1 class TTArcer

— class TTArcer —

```
class TTArcer : public ARCer
{
public:
  TTArcer( ComputationManager& boss )
    : ARCer( boss ), ttp( 0 ), bHappy( true )
  ~TTArcer( )
  void setArguments( const FPGroup& G );
  Chars getFileName( );
  bool isHappy( )
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  FPGroup theGroup;
  bool bHappy;
  TTP* ttp;
};
```

———-

20.51.2 class TTCM

— class TTCM —

```
class TTCM : public ComputationManager
{
public:
  TTCM(class TTProblem& PO);
  ~TTCM( );
  Chars getFileName( )
```

```
  bool isHappy()
  void viewStructure(ostream& ostr) const
  void takeControl( );
  void start( );
  void terminate( )
private:
  const SMFPGroup& theGroup;
  long inPos;
  File out;
  FILE* in;
  bool bInstructions;
  TTArcer arcer;
};
```

———-

20.51.3 class TTProblem

— **class TTProblem** —

```
class TTProblem : public Supervisor
{
public:
  TTProblem(class SMFPGroup& G);
  const SMFPGroup& getGroup( ) const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMFPGroup& theGroup;
  bool linkHasBeenSent;
  Subordinate<TTProblem, TTCM> tt;
};
```

———-

20.52 SMApps/include/WhiteheadMinimal.h

— **WhiteheadMinimal.h** —

```
#include "Supervisor.h"
#include "GAIsPartOfBasis.h"
#include "SMVectorOfWords.h"
```

———-

20.52.1 class GAFindWhiteheadMinimalArcer

— class GAFindWhiteheadMinimalArcer —

```
class GAFindWhiteheadMinimalArcer : public ARCer
{
public:
  GAFindWhiteheadMinimalArcer( ComputationManager& );
  ~GAFindWhiteheadMinimalArcer( )
  void setArguments( FreeGroup f, const VectorOf<Word>& v );
  Chars getComputationFileName() const
  Chars getResultFileName() const
  const VectorOf<Word>& getAutomorphism() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  VectorOf<Word> theTuple;
  FreeGroup theGroup;
  File compFile;
  File resultFile;
  VectorOf<Word> theAuto;
};
```

———-

20.52.2 class FindWhiteheadMinimalProblem

— class FindWhiteheadMinimalProblem —

```
class FindWhiteheadMinimalProblem : public Supervisor
{
public:
  FindWhiteheadMinimalProblem(const class SMVectorOfWords& );
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( );
private:
  const SMVectorOfWords& theTuple;
  bool linkHasBeenSent;
  GAFindWhiteheadMinimalArcer arcer;
};
```

———-

20.53 SMApps/include/WordProblem.h

— **WordProblem.h** —

```
#include "Supervisor.h"
#include "SMWord.h"
#include "SetOfWordsChecker.h"
#include "NilpotentQuotients.h"
#include "ARCer.h"
#include "GeneticProblems.h"
#include "FNWP.h"
#include "GAWordProblemForORGroup.h"
```

———-

20.53.1 class ORWordProblemARCer

— **class ORWordProblemARCer** —

```
class ORWordProblemARCer : public ARCer
{
public:
  ORWordProblemARCer( ComputationManager& );
  void setArguments( const Word& relator, const Word& testWord );
  bool getRetValue()
  ProductOfRelatorConjugates wordDecomposition() const
  void runComputation( );
  void writeResults( ostream& );
  void readResults( istream& );
private:
  void printWord(const Word& w, ostream& ostr);
  Word theRelator;
  Word theTestWord;
  Trichotomy answer;
  bool retValue;
  ProductOfRelatorConjugates wDeco;
};
```

———-

20.53.2 class ORWordProblemCM

— **class ORWordProblemCM** —

```
class ORWordProblemCM : public ComputationManager
{
```

```
public:
  ORWordProblemCM( class WordProblem& PO );
  Trichotomy isTrivial( )
  ProductOfRelatorConjugates wordDecomposition() const
  void takeControl( );
  void start( );
  void terminate( );
private:
  Word theRelator;
  const SMWord& testWord;
  Trichotomy itIsTrivial;
  bool bStarted;
  ProductOfRelatorConjugates wDeco;
  ORWordProblemARCer arcer;
};
```

———-

20.53.3 class WordProblem

— class WordProblem —

```
class WordProblem : public Supervisor
{
public:
  WordProblem( class SMWord& w);
  Trichotomy answer( ) const
  const SMWord& getTestWord( ) const
  void viewStructure(ostream& ostr) const;
  void takeControl( );
  void start( )
  void terminate( )
private:
  SMWord& theWord;
  SetOfWordsChecker theChecker;
  Trichotomy theAnswer;
  Chars explanation;
  bool triedAbelianization;
  MirrorSubordinate normalClosure;
  MirrorSubordinate abelianInvariants;
  MirrorSubordinate kbSupervisor;
  MirrorSubordinate agSupervisor;
  MirrorSubordinate computeBasis;
  MirrorSubordinate nilpotentQuotients;
  Subordinate<WordProblem,NilpotentWPInQuotients> nilpotentWPInQuotients;
  Subordinate<WordProblem, ORWordProblemCM> orwp;
  Subordinate<WordProblem, NilpotentWP> nilpotentWP;
  Subordinate<WordProblem, GeneticWPCM> genetic;
```

```
  bool linkGenORHasBeenSent;
  Subordinate<WordProblem, GAWordForORGroup> geneticForORG;
  Subordinate<WordProblem, FNWPCM> fnwp;
};
```

———-

21 The Subgroup classes

21.1 Subgroup/include/DecomposeInSubgroup.h

— **DecomposeInSubgroup.h** —

```
#include "SubgroupGraph.h"
#include "FreeGroup.h"
#include "../AProducts/include/AP-fixups.h"
```

———-

21.1.1 class DecomposeInSubgroupOfFreeGroup

— **class DecomposeInSubgroupOfFreeGroup** —

```
class DecomposeInSubgroupOfFreeGroup
{
public:
  DecomposeInSubgroupOfFreeGroup( const int ambientRank,
          const VectorOf<Word>& gens );
  virtual ~DecomposeInSubgroupOfFreeGroup( )
  virtual DecomposeInSubgroupOfFreeGroup* clone( ) const
  int rankOfFreeGroup( ) const
  VectorOf<Word> generators( ) const
  const SubgroupGraph& graph() const
  VectorOf<Word> decomposeNielsenBasis( ) const
  bool contains( const Word& w ) const
  Word decompose( const Word& w ) const;
  Trichotomy checkDecomposition( const Word& w ) const;
  DecomposeInSubgroupOfFreeGroup( const SubgroupGraph& G,
          const VectorOf<Word>& gens );
  friend
  ostream& operator<(ostream& ostr, const DecomposeInSubgroupOfFreeGroup& d)
  friend
  istream& operator>(istream& istr, DecomposeInSubgroupOfFreeGroup& d );
  virtual bool readPiece( istream& istr, const class Timer& timer );
private:
  void init( );
```

```
protected:
  VectorOf<Word> makeMapBetweenNielsens( SubgroupGraph& S,
           const VectorOf<Word>& words ) const;
  int theAmbientRank;
  VectorOf<Word> theGenerators;
protected:
  SubgroupGraph theGraph;
  VectorOf<Word> nielsenInSubgroupGenerators;
  virtual void write( ostream& ostr ) const;
  int n;
private:
  enum ReadingState { STOP, GENS, GRAPH, NIELSEN };
  ReadingState readState;
};
```

———-

21.1.2 class DecomposeInExpandingSubgroup

— class DecomposeInExpandingSubgroup —

```
class DecomposeInExpandingSubgroup : public DecomposeInSubgroupOfFreeGroup
{
public:
  DecomposeInExpandingSubgroup( int ambientRank,
        const VectorOf<Word>& constantSubgroup,
        const VectorOf<Word>& expandingSubgroup );
  DecomposeInSubgroupOfFreeGroup* clone( ) const
  VectorOf<Word> constantGenerators( ) const
  VectorOf<Word> expandingGenerators( ) const
  VectorOf<Generator> expandingConjugators( ) const
  void expandSubgroupByConjugation( const Generator& conjugator );
  void expressAsConjugateOfOriginalGenerator( const Generator& g,
                Generator& og, Word& conjugator)
    const;
  ProductOfRelatorConjugates fullDecomposition( const Word& w ) const;
  Trichotomy checkDecomposition( const Word& w ) const;
  bool readPiece( istream& istr, const class Timer& timer );
private:
  VectorOf<Word> shiftGenerators( const VectorOf<Word>& v, int shift );
  SubgroupGraph theConstantGraph;
  SubgroupGraph theExpandingGraph;
  VectorOf<Word> theConstantGenerators;
  VectorOf<Word> theExpandingGenerators;
  int theNumberOfExpandedGenerators;
  VectorOf<Generator> theExpandingConjugators;
  VectorOf<Word> Nc, NcInOld, NeOInOld;
protected:
```

```
  virtual void write( ostream& ostr ) const;
private:
  enum ReadingState2 { STOP, BASECLASS, CONSTGRAPH, EXPANDGRAPH, CONSTGENS,
           EXPANDGENS, READNC,READNCINOLD, READNEOINOLD };
  ReadingState2 readState2;
};
```

———-

21.1.3 class DecomposeInSubgroupOfFPGroup

— class DecomposeInSubgroupOfFPGroup —

```
class DecomposeInSubgroupOfFPGroup : public DecomposeInExpandingSubgroup
{
public:
  DecomposeInSubgroupOfFPGroup( const FPGroup& G, const VectorOf<Word>& gens );
  DecomposeInSubgroupOfFreeGroup* clone( ) const
  FPGroup group() const
  void expandGraph( );
  int numberOfIterations( ) const
  bool readPiece( istream& istr, const class Timer& timer );
private:
  FPGroup theGroup;
  Generator g;
  int theNumberOfIterations;
protected:
  virtual void write( ostream& ostr ) const;
private:
  enum ReadingState3 { STOP, NEWDATA };
  ReadingState3 readState3;
};
```

———-

21.2 Subgroup/include/DoubleCosetGraph.h

— DoubleCosetGraph.h —

```
#include "SubgroupGraph.h"

typedef SubgroupGraphRep::LabelType DCGLabelType;
typedef SubgroupGraphRep::VertexType DCGVertexType;
enum DCGPlace { inH, inU, inK };
```

———-

21.2.1 struct DCGState

— struct DCGState —

```
struct DCGState {
  DCGState( DCGPlace place = inH ,
  DCGVertexType vertex = SubgroupGraphRep::baseVertex )
    : currentPlace( place ), currentVertex( vertex )
  DCGState( DCGPlace place, int uCurrent ) :
    currentPlace( place ), UCurrent( uCurrent )
  DCGState& operator = ( const DCGState& state )
  bool operator == ( const DCGState& state ) const
  bool operator != ( const DCGState& state ) const
  friend ostream& operator << ( ostream& ostr, const DCGState& state )
  void assign( DCGPlace place, DCGVertexType vertex)
  void assign( DCGPlace place, int uCurrent )
  DCGPlace currentPlace;
  union {
    DCGVertexType currentVertex;
    int UCurrent;
  };

};
```

———-

21.2.2 class DoubleCosetGraph

— class DoubleCosetGraph —

```
class DoubleCosetGraph
{
public:
  DoubleCosetGraph( const SubgroupGraph& h, const Word& u,
        const SubgroupGraph& k, int hWidth, int kWidth);
  ~DoubleCosetGraph();
  bool contains( const Word& ) const;
  bool canGo( const DCGState& originState, DCGState& newState,
        DCGLabelType label ) const;
  bool canBack( const DCGState& originState, DCGState& newState,
          DCGLabelType label ) const;
  bool canAdvancedGo( const DCGState& originState,
          DCGState& newState, DCGLabelType label ) const;
  bool canAdvancedBack( const DCGState& originState,
            DCGState& newState, DCGLabelType label ) const;
  bool goThroughWord( const Word& W, const DCGState& originState,
          DCGState& finishState, int& WCurrent ) const;
  bool findWord( const Word& W, const DCGState& originState,
```

```
    DCGState& finishState ) const;
  static const DCGVertexType emptyTarget = SubgroupGraphRep::emptyTarget;
  static const DCGVertexType baseVertex = SubgroupGraphRep::baseVertex;
  const SubgroupGraph H;
  const SubgroupGraph K;
  const Word U;
private:
  DCGVertexType **subgroupArrows;
  DCGVertexType *wordArrows;
  DCGVertexType **backSubgroupArrows;
  DCGVertexType *backWordArrows;
  int HWidth;
  int KWidth;
  DCGVertexType enumOriginVertex;
  int enumWordLength;
  DoubleCosetGraph( const DoubleCosetGraph& );
  DoubleCosetGraph& operator = ( const DoubleCosetGraph& );
  void joinPieces( DCGVertexType HVertex, DCGVertexType KVertex );
public:
  void debugPrint( );
};

inline bool DoubleCosetGraph::findWord( const Word& W,
      const DCGState& originState, DCGState& finishState ) const
```

———-

21.2.3 class DCGVertexIterator

— class DCGVertexIterator —

```
class DCGVertexIterator
{
public:
  DCGVertexIterator( const DoubleCosetGraph& dcg ) : DCG( dcg )
  bool done( )
  void reset( );
  DCGState value( )
  bool next( );
private:
  const DoubleCosetGraph& DCG;
  DCGState state;
  bool bDone;
};
```

———-

21.3 Subgroup/include/GraphConjugacyProblem.h

— GraphConjugacyProblem.h —

```
#include "DoubleCosetGraph.h"
#include "FPGroup.h"
#include "Timer.h"
```

———-

21.3.1 struct DoubleWayElt

— struct DoubleWayElt —

```
struct DoubleWayElt {
  void assign( const DCGState& state1, const DCGState& state2,
         DCGLabelType label )
  void take( DCGState& state1, DCGState& state2, DCGLabelType& label)
  DCGState leftState, rightState;
  DCGLabelType currentLabel;
};
```

———-

21.3.2 class GraphConjugacyProblem

— class GraphConjugacyProblem —

```
class GraphConjugacyProblem
{
public:
  GraphConjugacyProblem( const FPGroup&, const Word& u, const Word& v );
  ~GraphConjugacyProblem( )
  void startComputation( )
  void continueComputation( const SubgroupGraph& theGraph );
  void continueComputation( );
  bool done( ) const
  bool theNewGraphIsNeeded( ) const
  Word getConjugator( )
private:
  int numberOfGenerators;
  int maxGeneratorLength;
  Word theConjugator;
  const FPGroup theGroup;
  Word UConjugator;
  Word VConjugator;
  Word U;
```

```
  Word V;
  bool bStart;
  bool bDone;
  DoubleCosetGraph *DCG;
  DCGVertexIterator *I;
  Timer *timer;
  static const int timerValue = 1000;
  bool isInterrupted;
  DoubleWayElt *way;
  int saveWayIndex;
  int saveLabel;
  DCGState saveLeftState;
  DCGState saveRightState;
  int *leftMarks;
  int *rightMarks;
  GraphConjugacyProblem( const GraphConjugacyProblem& );
  GraphConjugacyProblem& operator = ( const GraphConjugacyProblem& );
  void finishComputation( )
  bool findConjugator( const DCGState& state1, DCGLabelType label1,
           const DCGState& state2, DCGLabelType label2,
           int length, Word& conjugator);
  bool theConjugatorIsFound( const DCGState& state1,
                             const DCGState& state2 ) const;
  void mark( int *, const DCGState&, int *, const DCGState&, int );
  void clear( int *, const DCGState&, int *, const DCGState& );
  void getMarks( const int *, const DCGState&, const int *, const DCGState&,
    int&, int& ) const;
  bool weHaveTheSameCycles( int *, const DCGState&,
          int *, const DCGState& ) const;
  void finishInterruption( );
};
```

———-

21.4 Subgroup/include/PresentationsOfSubgroup.h

— PresentationsOfSubgroup.h —

```
#include "Subgroup.h"
#include "FPGroup.h"
#include "File.h"
#include "CosetEnumerator.h"
```

———-

21.4.1 class PresentationsOfSubgroup

— class PresentationsOfSubgroup —

```
class PresentationsOfSubgroup
{
 public:
  PresentationsOfSubgroup( const Subgroup& );
  FPGroup makePresentation( File &tmpOutput );
  FPGroup makePresentationTC( const PermutationRepresentation &pr,
            File &tmpOutput );
  FPGroup makePurePresentationTC( const PermutationRepresentation &pr );
  Word rewriteWord( const PermutationRepresentation& pr , const Word& w );
 private:
  Subgroup H;
  FPGroup G;
  VectorOf<Word> theGenerators;
};
```

———-

21.5 Subgroup/include/SGofFreeGroup.h

— SGofFreeGroup.h —

```
#include "ObjectOf.h"
#include "RefCounter.h"
#include "FreeGroup.h"
#include "SubgroupGraph.h"
```

———-

21.5.1 class SGofFreeGroupRep

— class SGofFreeGroupRep —

```
class SGofFreeGroupRep : public GenericRep {
public:
  SGofFreeGroupRep( const FreeGroup& parent, const VectorOf<Word>& gens );
  SGofFreeGroupRep( const FreeGroup& parent, SubgroupGraph SGG ) :
    computedNielsenBasis(false),
    theSubgroupGraph(SGG),
    builtSubgroupGraph(true),
    theGenerators(SGG.nielsenBasis()),
    theParentGroup(parent)
  PureRep* clone( ) const
  static const Type theSGofFreeGroupType;
  static Type type( )
  virtual Type actualType( ) const
```

```
  int hash() const;
  int order( )
  Bool isTrivial( )
  Bool isFinite( )
  Bool isInfinite( )
  Bool isAbelian( )
  bool isMalnormal( Word& conjugator );
  SGofFreeGroupRep* join(SGofFreeGroupRep& SGR);
  SGofFreeGroupRep* intersection(SGofFreeGroupRep& SGR);
  Bool isNormal( );
  VectorOf<Word> normalizer( );
  VectorOf<Word> nielsenBasis( );
  Word nielsenWord(int i);
  Word inNielsenWords(const Word& w);
  int rank();
  Elt eval( const Word& w ) const;
  Bool wordProblem( const Word& w ) const;
  Bool conjugacyProblem( const Word& u, const Word& v ) const;
  Bool contains(const Word& w);
  Bool contains(const SetOf<Word>& S);
  Bool contains(const VectorOf<Word>& V);
  Bool contains(const SGofFreeGroupRep& SGR);
  Bool equalTo(const SetOf<Word>& S);
  Bool conjugateInSubgroup(const Word& w, Word& conjugator);
  Bool conjugateInSubgroup(const SetOf<Word>& S, Word& conjugator);
  bool conjugateTo(const SetOf<Word>& S);
  long powerInSubgroup( const Word& w );
  int findIndex();
  VectorOf<Word> findWhiteheadBasis();
  Bool isAFreeFactor();
  Bool generatesTheFreeGroup();
  Word rightSchreierRepresentative(const Word& w);
  SGofFreeGroupRep* MHallCompletion( );
  void makeSubgroupGraph( );
  void printOn(ostream&) const;
  SGofFreeGroupRep* readFrom(istream&, Chars&) const
  void printGenerator( ostream& ostr, int n ) const;
  void printGenerators( ostream& ostr ) const;
  void printWord( ostream& ostr, const Word& w ) const;
  bool computedNielsenBasis;
  bool builtSubgroupGraph;
  VectorOf<Word> theGenerators;
  VectorOf<Word> NielsenBasis;
  SubgroupGraph theSubgroupGraph;
  FreeGroup theParentGroup;
private:
  SGofFreeGroupRep& operator = ( const SGofFreeGroupRep& );
};
```

———-

21.5.2 class SGofFreeGroup

— class SGofFreeGroup —

```
class SGofFreeGroup : public GenericObject {
public:
  SGofFreeGroup( const FreeGroup& parent, const VectorOf<Word>& gens )
  : GenericObject( new SGofFreeGroupRep(parent,gens) )
  int hash() const
  bool operator == ( const SGofFreeGroup& g) const;
  const FreeGroup& parentGroup( ) const
  const VectorOf<Word>& generators( ) const
  static Type type( )
  Type actualType( ) const
  int order( )
  Bool isTrivial( )
  Bool isFinite( )
  Bool isInfinite( )
  Bool isAbelian( )
  bool isMalnormal( Word& conjugator )
  SGofFreeGroup join(SGofFreeGroup& SG)
  SGofFreeGroup intersection(SGofFreeGroup& SG)
  Bool isNormal( )
  VectorOf<Word> normalizer( )
  VectorOf<Word> nielsenBasis( )
  Word nielsenWord(int i)
  Word inNielsenWords(const Word& w)
  int rank()
  SGofFreeGroup MHallCompletion( )
  Elt eval( const Word& w ) const
  Bool wordProblem( const Word& w ) const
  Bool conjugacyProblem( const Word& u, const Word& v ) const
  Bool contains(const Word& w)
  Bool contains(const SetOf<Word>& S)
  Bool contains(const VectorOf<Word>& V)
  Bool contains(const SGofFreeGroup& SG)
  Bool equalTo(const SetOf<Word>& S)
  Bool conjugateInSubgroup(const Word& w, Word& conjugator)
  Bool conjugateInSubgroup(const SetOf<Word>& S, Word& conjugator)
  bool conjugateTo(const SetOf<Word>& S)
  long powerInSubgroup( const Word& w )
  int findIndex()
  VectorOf<Word> findWhiteheadBasis()
  Bool isAFreeFactor()
  Bool generatesTheFreeGroup()
  Word rightSchreierRepresentative(const Word& w)
  void printWord( ostream& ostr, const Word& w ) const
```

```
protected:
  const SGofFreeGroupRep* look( ) const
  SGofFreeGroupRep* enhance( ) const
  SGofFreeGroupRep* change( )
  SGofFreeGroup( SGofFreeGroupRep* newrep ) : GenericObject(newrep)
};

Word expressWordInSubgroupGenerators( const SGofFreeGroup& H, const Word& w);
```

———-

21.6 Subgroup/include/SubgroupGraph.h

— **SubgroupGraph.h** —

```
#include "SubgroupGraphRep.h"
```

———-

21.6.1 class SubgroupGraph

— **class SubgroupGraph** —

```
class SubgroupGraph : public ObjectOf<SubgroupGraphRep> {
public:
  typedef SubgroupGraphRep::VertexType VertexType;
  typedef SubgroupGraphRep::LabelType LabelType;
  SubgroupGraph(int ambientRank, const SetOf<Word>& S) :
    ObjectOf<SubgroupGraphRep>( new SubgroupGraphRep(ambientRank, S) )
  SubgroupGraph(int ambientRank, const VectorOf<Word>& V) :
    ObjectOf<SubgroupGraphRep>( new SubgroupGraphRep(ambientRank, V) )
  int rank( ) const
  VectorOf<Word> normalizer( )
  VectorOf<Word> nielsenBasis( ) const
  Word nielsenWord(int i) const
  Word inNielsenWords(const Word& w) const
  SubgroupGraph join(const SubgroupGraph& SG) const
  SubgroupGraph intersection(const SubgroupGraph& SG) const
  Bool contains(const Word& w) const
  Bool contains(const SetOf<Word>& S) const
  Bool contains(const VectorOf<Word>& V) const
  Bool contains(SubgroupGraph& SG) const
  Bool equalTo(const SetOf<Word>& S)
  Bool equalTo(SubgroupGraph& SG)
  Bool conjugateInSubgroup(const Word& w, Word& conjugator) const
  Bool conjugateInSubgroup(const SetOf<Word>& S, Word& conjugator)
```

```
  bool conjugateTo(const SetOf<Word>& S)
  long powerInSubgroup( const Word& w ) const
  int findIndex()
  VectorOf<Word> findWhiteheadBasis()
  Bool isAFreeFactor()
  Bool generatesTheFreeGroup() const
  Word rightSchreierRepresentative(const Word& w)
  SubgroupGraph MHallCompletion( ) const
  void MHallComplete( )
  void joinConjugate(Generator g)
  float completeness( ) const
  Bool isComplete( ) const
  VertexType vertexCount( ) const
  VertexType targetOfGenerator(VertexType source, int generator) const
  VertexType targetOfLabel(VertexType source, LabelType label) const
  long getValence( ) const
  LabelType inverseLabel(LabelType label) const
  int labelToGenerator(LabelType label) const
  LabelType generatorToLabel(int g) const
  friend ostream& operator < ( ostream& ostr, const SubgroupGraph& g )
  friend istream& operator > ( istream& istr, SubgroupGraph& g )
  bool readPiece( istream& istr, const class Timer& timer )
protected:
  SubgroupGraph(SubgroupGraphRep* SGRp) : ObjectOf<SubgroupGraphRep>(SGRp)
};
```

———-

21.7 Subgroup/include/SubgroupGraphRep.h

— SubgroupGraphRep.h —

```
#include "Set.h"
#include "Word.h"
#include "Vector.h"
#include "Generator.h"
#include "RefCounter.h"
```

———-

21.7.1 class SubgroupGraphRep

— class SubgroupGraphRep —

```
class SubgroupGraphRep : public PureRep {
  friend class SubgroupGraphRepReader;
```

```
public:
  SubgroupGraphRep(int ambientRank, const SetOf<Word>& S);
  SubgroupGraphRep(int ambientRank, const VectorOf<Word>& V);
  SubgroupGraphRep(const SubgroupGraphRep&);
  ~SubgroupGraphRep( );
  SubgroupGraphRep* clone( ) const
  int rank( ) const
  VectorOf<Word> normalizer( );
  VectorOf<Word> nielsenBasis( );
  Word nielsenWord(int i);
  Word inNielsenWords(const Word& w);
  SubgroupGraphRep* join(const SubgroupGraphRep&) const;
  SubgroupGraphRep* intersection(const SubgroupGraphRep& SGR) const;
  Bool contains(const Word& w) const
  Bool contains(const SetOf<Word>& S) const;
  Bool contains(const VectorOf<Word>& V) const;
  Bool contains(SubgroupGraphRep& SGR) const;
  Bool equalTo(const SetOf<Word>& S);
  Bool equalTo(SubgroupGraphRep& SGR);
  Bool conjugateInSubgroup(const Word& w, Word& conjugator) const;
  Bool conjugateInSubgroup(const SetOf<Word>& S, Word& conjugator);
  bool conjugateTo(const SetOf<Word>& S);
  long powerInSubgroup( const Word& aWord ) const;
  int findIndex();
  VectorOf<Word> findWhiteheadBasis();
  Bool isAFreeFactor();
  Bool generatesTheFreeGroup() const
  Word rightSchreierRepresentative(const Word&);
  void MHallComplete();
  void joinConjugate(int generator);
  float completeness( ) const
  Bool isComplete( ) const
  long vertexCount( ) const
  typedef long VertexType;
  static const VertexType baseVertex = 0;
  static const VertexType emptyTarget = -1;
  typedef int LabelType;
  LabelType generatorToLabel(int g) const
  int labelToGenerator(LabelType label) const
  LabelType inverseLabel(LabelType label) const
  VertexType targetOfGenerator(VertexType source, int generator) const
  VertexType targetOfLabel(VertexType source, LabelType label) const
  LabelType getValence( ) const
  virtual void write( ostream& ostr ) const;
  virtual void read( istream& istr );
  virtual bool readPiece( istream& istr, const class Timer& timer );
protected:
  enum { STOP, TABLE, SUBTREE, BASIS_EDGES };
  int isReading;
  int tableSize, n;
```

———-

21.7.2 struct Edge

— **struct Edge** —

```
  struct Edge {
    VertexType v;
    LabelType  l;
    Edge& operator = ( const Edge& anEdge )
    Bool operator < ( const Edge& anEdge ) const
    Bool operator > ( const Edge& anEdge ) const
    friend ostream& operator < ( ostream& ostr, const Edge& e )
    friend istream& operator > ( istream& istr, Edge& e )
  };
  VertexType* table;
  VertexType numberOfVertices;
  VertexType spaceForVertices;
  LabelType  valence;
  LabelType* maximalSubtree;
  Edge* basisEdges;
  SubgroupGraphRep(int whatValence, long howManyVertices);
  void resize(long howManyVertices);
  void defineEdges(VertexType source, int generator, VertexType target)
  VertexType defineEdges(VertexType source, int generator)
  void defineEdge(VertexType source, LabelType label, VertexType target)
  long numberOfEdges( ) const;
  long valenceAt(VertexType v) const;
  void adjoinWord(const Word&, VertexType);
  void addWordArc(const Word&, VertexType, VertexType, int, int);
  void addWordLoop(const Word&, VertexType, int, int);
  void identifyVertices(VertexType, VertexType);
  void reduceGraph(VertexType*);
  SubgroupGraphRep* disjointUnion(const SubgroupGraphRep&) const;
  Bool loopSearch(const Word& w, VertexType v) const;
  Word getLabel(VertexType v) const;
  Word getInverseLabel(VertexType v) const;
  int findBasisEdgeIndex(VertexType source,LabelType label);
  void makeMaxlSubtree( );
  int distanceFromOrigin(VertexType) const;

};
```

———-

21.8 Subgroup/include/Subgroup.h

— **Subgroup.h** —

```
#include "FGGroupRep.h"
#include "FGGroup.h"
#include "FPGroup.h"
#include "File.h"
#include "CosetEnumerator.h"
```

———-

21.8.1 struct SubgroupRep

— **struct SubgroupRep** —

```
struct SubgroupRep : FGGroupRep {
  SubgroupRep( const FGGroup& parent )
  : FGGroupRep(0),
   theParentGroup(parent)
  SubgroupRep( const FGGroup& parent, const VectorOf<Word>& gens )
  : FGGroupRep( gens.length() ),
   theParentGroup( parent ),
   theGenerators( gens )
  SubgroupRep( const FGGroup& parent, const SetOf<Word>& gens );
  PureRep* clone( ) const
  static const Type theSubgroupType;
  static Type type( )
  Type actualType( ) const
private:
  SubgroupRep& operator = ( const SubgroupRep& );
public:
  int order( ) const
  Trichotomy isTrivial( ) const
  Trichotomy isFinite( ) const
  Trichotomy isInfinite( ) const
  Trichotomy isAbelian( ) const
  Trichotomy areEqual(const Elt&, const Elt&) const
  Elt eval( const Word& w ) const;
  Trichotomy wordProblem( const Word& w ) const;
  Trichotomy conjugacyProblem( const Word& u, const Word& v ) const;
  Word findRelator( );
  Bool redundantRelator(const Word& u);
  void printOn(ostream&) const
  virtual GroupRep* readFrom(istream&, Chars&) const
  FGGroup theParentGroup;
  VectorOf<Word> theGenerators;
  Word lastWordTried;
  SetOf<Word> relators;
};
```

———-

21.8.2 class Subgroup

— class Subgroup —

```
class Subgroup : public FGGroup {
public:
  Subgroup( const FGGroup& parent )
  : FGGroup( new SubgroupRep( parent ) )
  Subgroup( const FGGroup& parent, const VectorOf<Word>& gens )
  : FGGroup( new SubgroupRep(parent,gens) )
  Subgroup( const FGGroup& parent, const SetOf<Word>& gens )
  : FGGroup( new SubgroupRep(parent,gens) )
  static Type type( )
  const FGGroup& parentGroup( ) const
  const VectorOf<Word>& generators( ) const
  void setParentGroup( const Group& g )
  void setGenerators( const VectorOf<Word>& s )
public:
  Subgroup& addGenerator( const Word& w );
  Subgroup& deleteGenerator( const Word& w );
  Word findRelator( )
  Bool redundantRelator(const Word& u)
private:
  const SubgroupRep* look( ) const
  SubgroupRep* enhance( ) const
  SubgroupRep* change( )
  Subgroup( SubgroupRep* newrep ) : FGGroup(newrep)
};
```

———-

21.9 Subgroup/include/TurnerProperSubgroupEnumerator.h

— TurnerProperSubgroupEnumerator.h —

```
#include "FreeGroup.h"
#include "File.h"
#include "SGofFreeGroup.h"
```

———-

21.9.1 class ProperSubgroupEnumerator

— class ProperSubgroupEnumerator —

```
class ProperSubgroupEnumerator
{
public:
  ProperSubgroupEnumerator(const FreeGroup& F, const VectorOf<Word>& words);
  ProperSubgroupEnumerator(const FreeGroup& F, const Word& word);
 ~ProperSubgroupEnumerator();
  ProperSubgroupEnumerator(const ProperSubgroupEnumerator& PSE);
  int operator == (const ProperSubgroupEnumerator& PSE);
  bool getNextProperSubgroup(SGofFreeGroup& sg);
  Chars getFileName()
private:
  const VectorOf<Word> theWords;
  const FreeGroup theGroup;
  VectorOf<Word> genOfGroup;
  const int rank;
  bool stepTo();
  void stepBack();
  bool setRefs();
  void buildSG( );
  Word getWord(int x, int y);
  File file;
  SGofFreeGroup *subgroup;
  VectorOf<Word> genOfSG;
  int qWord;
  int nVert;
  int nSet;
  int *partition;
  int **_partition;
  int qSet;
  int qVert;
  int *setToVert;
  int *vertToWord;
  int *vertToPos;
```

———-

21.9.2 struct LC

— **struct LC** —

```
  struct LC {
    int qSet;
  }* lCells;
  enum { UNKNOWN=-1 };
};
```

———-

22 The Todd-Coxeter classes

22.1 Todd-Coxeter/include/CosetEnumerator.h

— **CosetEnumerator.h** —

```
#include "CosetTables.h"
#include "FPGroup.h"
```

———-

22.1.1 class PermutationRepresentation

— **class PermutationRepresentation** —

```
class PermutationRepresentation {
public:
  PermutationRepresentation():permTable(NULL), numberOfGens(0)
  PermutationRepresentation(const PermutationRepresentation& p);
  ~PermutationRepresentation()
  PermutationRepresentation& operator = ( const PermutationRepresentation& p);
  void printOn(const VectorOf<Chars>& n, ostream& ostr)const;
  Word representativeOf(const Word& )const;
  const VectorOf<Word>& getRepresentatives()const
  bool isEmpty()const
  friend ostream& operator < (ostream& ostr,
            const PermutationRepresentation& PR )
  friend istream& operator > ( istream& istr, PermutationRepresentation& PR)
private:
  friend class CosetEnumerator;
  PermutationRepresentation(int numOfGens):
    numberOfGens(numOfGens),
    permTable(new ListOf< VectorOf<int> >[numOfGens])
  void addCycle(int g,const VectorOf<int>& v);
  void setRepresentatives(const VectorOf<Word>& r)
  int searchNext(const Generator& g,int coset)const;
  ListOf< VectorOf<int> >* permTable;
  int numberOfGens;
  VectorOf<Word> representatives;
};
```

———-

22.1.2 class CosetEnumerator

— **class CosetEnumerator** —

```
class CosetEnumerator {
public:
  CosetEnumerator(const FPGroup& group);
  CosetEnumerator(const FPGroup& group,const VectorOf<Word>& subgroup);
  ~CosetEnumerator();
  int enumerate();
  const VectorOf<Word> schreierRepresentatives()const
  const PermutationRepresentation& permutationRepresentation()const
private:
  bool addCoset(int coset_num);
  void  iterateTables( );
  void removeCollisions( );
  bool infinite()const;
  CosetEnumerator(const CosetEnumerator& );
  void makeRepresentatives();
  void makePermutations();
  CosetRelationsSet theRelationSet;
  CosetTable** tables;
  FPGroup theGroup;
  VectorOf<Word> theSubgroup;
  int theOrder;
  bool haveFastSolution;
  int numberOfGroupRelators;
  int numberOfTables;
  VectorOf<Integer> cosetNumbers;
  VectorOf<Word> schreierRepres;
  PermutationRepresentation thePermutationRepresentation;
  BTree<int,int> cosets2repres;
};
```

———-

22.2 Todd-Coxeter/include/CosetRelations.h

— **CosetRelations.h** —

```
#include "Word.h"
#include "BTree.h"
#include "Vector.h"
```

———-

22.2.1 class CosetRelationsSet

— **class CosetRelationsSet** —

```
class CosetRelationsSet {
public:
  CosetRelationsSet(int num_of_gens):
    relations(num_of_gens*2),
    numOfCollisions(0),
    theCollisions(4){}
  const BTree<int,int>& relationsOf(Generator i)const
  bool addRelation(int first,const Generator& g,int second,bool inverse=false);
  int getRelationNumber(int ,const Generator&)const;
  void replaceCosets(int good,int bad);
  int numberOfCollisions()const;
  bool removeCollision(int);
  bool collision(int& good,int& bad)const;
  int indexRelOf(const Generator& g)const;
  Generator genOfRels(int i)const;
  friend ostream& operator << ( ostream& ostr, const CosetRelationsSet& cr )
private:
  CosetRelationsSet(const CosetRelationsSet&);
  void addCollision(int bad,int good);
  int numOfCollisions;
  VectorOf<BTree<int,int> > relations;
  BTree< int, int > theCollisions;
};
```

———-

22.3 Todd-Coxeter/include/CosetTables.h

— **CosetTables.h** —

```
#include "Word.h"
#include "CosetRelations.h"
#include "EasyList.h"
#include "Integer.h"
```

———-

22.3.1 class CosetRow

— **class CosetRow** —

```
class CosetRow {
public:
  CosetRow(int length):
    row(new int[length]),
    left(0),
```

```
    right(length)
  ~CosetRow()
  CosetRow(const CosetRow&);
  bool operator == (const CosetRow& cr);
  int getCell (int i)const
  void setCell(int i,int index)
  CosetRow operator = (const CosetRow& cr);
  int left;
  int right;
private:
  void printOn(ostream& ostr=cout)  }
  int* row;
};
```

———-

22.3.2 class CosetTable

— class CosetTable —

```
class CosetTable {
public:
  CosetTable(const Word& w,CosetRelationsSet& crs,
       VectorOf<Integer>& coset_nums,
       bool sgTable = false);
  ~CosetTable();
  bool complete() const {return !filedTable.isEmpty() &&
         currentTable.isEmpty();}
  bool addCoset(int coset_number,bool first_used = false);
  bool removeCosets(int bad);
  bool iterateTable();
  int numberOfRows() const
  friend ostream& operator << ( ostream& ostr, const CosetTable& ct )
private:
  CosetTable(const CosetTable&);
  void printOn( ostream& ostr=cout)const;
  bool iterateFromRight();
  bool iterateFromLeft();
  bool getClosed( );
  Integer getCosetInRow(int i,CosetRow& row)const;
  int numOfLetters;
  Word relation;
  EasyList<CosetRow*> currentTable;
  EasyList<CosetRow*> filedTable;
  EasyList<CosetRow*> emptyTable;
  CosetRelationsSet& theRelationSet;
  bool relatorsChanged;
  int activeOrder;
```

```
  bool isSubgroupTable;
  VectorOf<Integer>& cosetNumbers;
};
```

———-

22.4 Todd-Coxeter/include/EasyList.h

— **EasyList.h** —

```
#include "Cell.h"

template <class T> class EasyList;

template <class T> class EasyCell {
public:
  EasyCell(const T& t): theContent(t),theNext(NULL){}
  EasyCell(const T& t, EasyCell* c): theContent(t),theNext(c){}
  EasyCell* next() const
  const T& content()const
  void setContent(const T& t)
  void setNext(EasyCell* n)
private:
  EasyCell(const EasyCell&);
  friend class EasyList<T>;
  T theContent;
  T& ref() {return theContent;}
  EasyCell* theNext;
};

template <class T> class EasyList {
public:
  EasyList( ) : root(NULL), last(NULL), current(NULL),end_was_red(false)
  EasyList( const T& t )
  EasyList( const EasyList& lr );
  ~EasyList( )
  bool isFirst() const
  bool isLast() const
  bool gotEnd() const
  EasyCell<T>* getLast() const
  EasyCell<T>* getFirst() const
  EasyCell<T>* getCurrent()const;
  const T& value()const
  bool setCurrent(EasyCell<T>* setpoint);
  void setContent(T setcont)
  bool setToFirst()
  bool setToLast()
  bool isEmpty() const
```

```
  bool next();
  void prepend(const T&);
  void append(const T&);
  void insertAfter(const T&);
  void insertBefore(const T&);
  bool deleteFirst();
  bool deleteCurrent();
  bool deleteAfter();
  bool setCurrentToIndex(int i);
  bool search(const T&,EasyCell<T>** ) const;
  bool deleteLast();
  bool operator == ( const EasyList& lr ) const;
  friend ostream& operator << ( ostream& ostr, const EasyList& ls )
  friend ostream& operator < ( ostream& ostr, const EasyList& cr )
  friend istream& operator > ( istream& istr, EasyList& cr )
private:
  void printOn(ostream& ostr=cout)const;
  EasyCell<T>* root;
  EasyCell<T>* last;
  EasyCell<T>* current;
  bool end_was_red;
};
```

———-

22.5 Todd-Coxeter/include/HavasTC.h

— HavasTC.h —

```
#include "BlackBox.h"
#include "FPGroup.h"

enum TC_STATE {NSTARTED, RUNNING, SUCCESS, NOMEM};
```

———-

22.5.1 class HavasTC

— class HavasTC —

```
class HavasTC {
public:
  HavasTC(const FPGroup& group);
  HavasTC(const FPGroup& group,const VectorOf<Word>& subgroup);
  ~HavasTC();
  void start();
```

```
  bool finished();
  int getIndex()const
  TC_STATE getState() const
  void printWord( ostream& ostr, const Word& w ) const;
  void setWorkspace(int w) { theWorkspace = w; }
private:
  bool isInitSegment(Chars str, Chars seg ) const;
  int theIndex;
  BlackBox tc;
  FPGroup theGroup;
  VectorOf<Word> theSubgroup;
  TC_STATE theState;
  int theWorkspace;
};
```

———-

References

[1] nothing