Open Geometry Prover
Programmer Manual

Nuno Baeta Pedro Quaresma

Draft 2022-12-15 09:58:53

Contents

Chapter 1

Introduction

This document:

- Defines a set of rules that every OGP prover must adhere;

As part of the OGP Community Project, contributions are welcome, whether they are, e.g., provers or improvements to the definition of API.

Chapter 2

Common Settings

- Each prover must create a library and a stand-alone program
- No programming language is enforced. However, the languange must be available as free/libre and open source software (FLOSS).
- Other tools must also be FLOSS.
- The library must include a function to call the prover:
  - it has only one argument, a file file with the conjecture;
  - it must return codes and file with other information. Other files may be created. The codes are: ...

  If the library implements several provers, write one such function for each prover.
- About stand-alone must adhere to a standard set of command line options.
  - ....

Chapter 3

Provers

3.1 OGP GDDM

dbRAM.hpp

```
/*
 * dbRAM.hpp
 *
 * Open, create and close the (in memory) SQLite database.
 *
 * This file is part of the OGP GDDM prover, which, in turn, is part of
 * the Open Geometry prover Community Project (OGPCP)
 * <https://github.com/opengeometryprover>.
 *
 * Copyright (C) 2022 Nuno Baeta, Pedro Quaresma
 * Distributed under GNU GPL 3.0 or later
 */


#ifndef DBINMEMORY
#define DBINMEMORY

#include <sqlite3.h>

#include <iostream>
#include <map>

class DBinMemory {
    friend class Prover;
    friend class FOFtoDB;
protected:
    // SQLite variables
    int rc;
    char *zErrMsg = 0;
    sqlite3 *db;
    sqlite3_stmt *stmt, *stmt1, *stmt2;

    int res = 0;
    std::map<std::string, int> geoCmds = {{"coll", 1},
```

```
                                        {"para", 2},
                                        {"perp", 3},
                                        {"midp", 4},
                                        {"circle", 5},
                                        {"cong", 6},
                                        {"contri", 7},
                                        {"cyclic", 8},
                                        {"eqangle", 9},
                                        {"eqratio", 10},
                                        {"simtri", 11}};

public:
  void openInMemoryDB();        // Open database (in memory)
  void createDBforGDDM();// Create database (in memory)
  void closeDB();               // Close (in memory) database
  int backupDb(const char *,void (*f)(int,int)); // backup DB in file
};

#endif
```

foftodb.hpp

```
/*
 * foftodb.cpp
 *
 * First Order Form to database manipulation: Driver, FOF parser, FOF
 * format to GDDM database FOFtoDB.
 *
 * This file is part of the OGP GDDM prover, which, in turn, is part of
 * the Open Geometry Prover Community Project (OGPCP)
 * <https://github.com/opengeometryprover>.
 *
 * Copyright (C) 2022 Nuno Baeta, Pedro Quaresma
 * Distributed under GNU GPL 3.0 or later
 */


#ifndef FOFTODB
#define FOFTODB

#include <map>

#include "parser.hpp"


class DBinMemory; // Just declare the name

/*
 * Driver classes (parser FOF)
 */

// Give Flex the prototype of yylex we want ...
#define YY_DECL yy::parser::symbol_type yylex (Driver& drv)
// ... and declare it for the parser's sake.
YY_DECL;

// Conducting the whole scanning and parsing of Calc++.
class Driver {
```

```
public:
    Driver ();

    int numGeoCmd = 0;
    int antconcedent[500];
    std::string typeGeoCmd[500];
    std::string point1[500], point2[500], point3[500], point4[500];
    std::string point5[500], point6[500], point7[500], point8[500];
    std::map<std::string, int> variables;
    int result;

    // Run the parser on file 'f'.  Return 0 on success.
    int parse(const std::string& f);
    // The name of the file being parsed.
    std::string file;
    // Whether to generate parser debug traces.
    bool trace_parsing;
    // Handling the scanner.
    void scan_begin();
    void scan_end();
    // Whether to generate scanner debug traces.
    bool trace_scanning;
    // The token's location used by the scanner.
    yy::location location;
};

class FOFtoDB {
private:

public:
    // Read (parse) the FOF conjecture (file) and populate the database
    DBinMemory readFileLoadDB(Driver, DBinMemory);
    // Show database status
    void showDB(DBinMemory);
};

#endif
```

prover.hpp

```
/*
 * prover.hpp
 *
 * GDDM's core: deduction rules and point fixed construction.
 *
 * This file is part of the OGP GDDM prover, which, in turn, is part of
 * the Open Geometry Prover Community Project (OGPCP)
 * <https://github.com/opengeometryprover>.
 *
 * Copyright (C) 2022 Nuno Baeta, Pedro Quaresma
 * Distributed under GNU GPL 3.0 or later
 */


#ifndef PROVER
#define PROVER

#include <string>
```

```
#include "dbRAM.hpp"

class DBinMemory;

class Prover {

private:
    void deriveNewColl(std::string,std::string,std::string);

public:
    // Save fixed point, the 'Facts' table to a file.
    void saveFixedPoint(DBinMemory, std::string);

    /*
     * Find the fixed point of the antecedents
     * --> assume the database of the method, already opened and populated
     * --> proved_t : clock_t, the exact moment when the proof was found
     * <-- databased modified, adding all the new facts derived from the
     * antecedents
     */
  DBinMemory fixedPoint(DBinMemory,clock_t *);

    /*
     * Verify if the conjecture was proved, or not
     * ---> database with all the facts in the fixed point
     * <--- true the consequent is in the facts table, false in the othe case
     */
    bool proved(DBinMemory);

    DBinMemory ruleD01(DBinMemory, std::string, std::string, std::string);
    DBinMemory ruleD02(DBinMemory, std::string, std::string, std::string);
    DBinMemory ruleD03(DBinMemory, std::string, std::string, std::string);
    DBinMemory ruleD04(DBinMemory, std::string, std::string, std::string,
                       std::string);
    DBinMemory ruleD05(DBinMemory, std::string, std::string, std::string,
                       std::string);
    DBinMemory ruleD06(DBinMemory, std::string, std::string, std::string,
                       std::string);
    DBinMemory ruleD07(DBinMemory, std::string, std::string, std::string,
                       std::string);
    DBinMemory ruleD08(DBinMemory, std::string, std::string, std::string,
                       std::string);
    DBinMemory ruleD09(DBinMemory, std::string, std::string, std::string,
                       std::string);
    DBinMemory ruleD10para(DBinMemory,std::string, std::string, std::string,
                           std::string);
    DBinMemory ruleD10perp(DBinMemory,std::string, std::string, std::string,
                           std::string);
    DBinMemory ruleD11(DBinMemory, std::string, std::string, std::string);
    DBinMemory ruleD12(DBinMemory, std::string, std::string, std::string,
                       std::string);
    DBinMemory ruleD13(DBinMemory, std::string, std::string, std::string,
                       std::string);
    DBinMemory ruleD14(DBinMemory, std::string, std::string, std::string,
                       std::string);
    DBinMemory ruleD15(DBinMemory, std::string, std::string, std::string,
```

```
                    std::string);
  DBinMemory ruleD16(DBinMemory, std::string, std::string, std::string,
                    std::string);
  DBinMemory ruleD17(DBinMemory, std::string, std::string, std::string,
                    std::string);
  DBinMemory ruleD18(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string, std::string,
                    std::string);
  DBinMemory ruleD19(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string, std::string,
                    std::string);
  DBinMemory ruleD20(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string, std::string,
                    std::string);
  DBinMemory ruleD21(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string, std::string,
                    std::string);
  DBinMemory ruleD22(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string, std::string,
                    std::string);
  DBinMemory ruleD23(DBinMemory, std::string, std::string, std::string,
                    std::string);
  DBinMemory ruleD24(DBinMemory, std::string, std::string, std::string,
                    std::string);
  DBinMemory ruleD25(DBinMemory, std::string, std::string, std::string,
                    std::string);
  DBinMemory ruleD26(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string, std::string,
                    std::string);
  DBinMemory ruleD27(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string, std::string,
                    std::string);
  DBinMemory ruleD28(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string, std::string,
                    std::string);
  DBinMemory ruleD29(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string, std::string,
                    std::string);
  DBinMemory ruleD30(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string, std::string,
                    std::string);
  DBinMemory ruleD31(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string);
  DBinMemory ruleD32(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string);
  DBinMemory ruleD33(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string);
  DBinMemory ruleD34(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string);
  DBinMemory ruleD35(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string);
  DBinMemory ruleD36(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string);
  DBinMemory ruleD37(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string);
  DBinMemory ruleD38(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string);
```

```
    DBinMemory ruleD39(DBinMemory, std::string, std::string, std::string,
                       std::string, std::string, std::string, std::string,
                       std::string);
    DBinMemory ruleD40(DBinMemory, std::string, std::string, std::string,
                       std::string);
    DBinMemory ruleD41(DBinMemory, std::string, std::string, std::string,
                       std::string);
    DBinMemory ruleD42(DBinMemory, std::string, std::string, std::string,
                       std::string, std::string, std::string,std::string,
                       std::string);
    DBinMemory ruleD43cyclic(DBinMemory, std::string, std::string, std::string,
                             std::string);
    DBinMemory ruleD43eqangle(DBinMemory, std::string, std::string, std::string,
                              std::string, std::string, std::string,
                              std::string, std::string);
    DBinMemory ruleD44(DBinMemory, std::string, std::string, std::string);
    DBinMemory ruleD45coll(DBinMemory, std::string, std::string, std::string);
    DBinMemory ruleD45midp(DBinMemory, std::string, std::string, std::string);
    DBinMemory ruleD45para(DBinMemory, std::string, std::string, std::string,
                           std::string);
    DBinMemory ruleD46(DBinMemory, std::string, std::string, std::string,
                       std::string);
    DBinMemory ruleD47(DBinMemory, std::string, std::string, std::string,
                       std::string, std::string, std::string, std::string,
                       std::string);
    DBinMemory ruleD48circle(DBinMemory, std::string, std::string, std::string,
                             std::string);
    DBinMemory ruleD48perp(DBinMemory, std::string, std::string, std::string,
                           std::string);
    DBinMemory ruleD49circle(DBinMemory, std::string, std::string, std::string,
                             std::string);
    DBinMemory ruleD49eqangle(DBinMemory, std::string, std::string, std::string,
                              std::string, std::string, std::string,
                              std::string, std::string);
    DBinMemory ruleD50circle(DBinMemory, std::string, std::string, std::string,
                             std::string);
    DBinMemory ruleD50midp(DBinMemory, std::string, std::string, std::string);
    DBinMemory ruleD51circle(DBinMemory, std::string, std::string, std::string,
                             std::string);
    DBinMemory ruleD51coll(DBinMemory, std::string, std::string, std::string);
    DBinMemory ruleD51eqangle(DBinMemory, std::string, std::string, std::string,
                              std::string, std::string, std::string,
                              std::string, std::string);
    DBinMemory ruleD52midp(DBinMemory, std::string, std::string, std::string);
    DBinMemory ruleD52perp(DBinMemory, std::string, std::string, std::string,
                           std::string);
    DBinMemory ruleD53circle(DBinMemory, std::string, std::string, std::string,
                             std::string);
    DBinMemory ruleD53coll(DBinMemory, std::string, std::string, std::string);
    DBinMemory ruleD54cyclic(DBinMemory, std::string, std::string, std::string,
                             std::string);
    DBinMemory ruleD54para(DBinMemory, std::string, std::string, std::string,
                           std::string);
    DBinMemory ruleD55midp(DBinMemory, std::string, std::string, std::string);
    DBinMemory ruleD55perp(DBinMemory, std::string, std::string, std::string,
                           std::string);
    DBinMemory ruleD56(DBinMemory, std::string, std::string, std::string,
```

```
                    std::string);
DBinMemory ruleD57cong(DBinMemory, std::string, std::string, std::string,
                       std::string);
DBinMemory ruleD57cyclic(DBinMemory, std::string, std::string, std::string,
                         std::string);
DBinMemory ruleD58a(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string, std::string,
                    std::string);
DBinMemory ruleD58b(DBinMemory, std::string, std::string, std::string,
                    std::string, std::string, std::string, std::string,
                    std::string);
DBinMemory ruleD59(DBinMemory, std::string, std::string, std::string,
                   std::string, std::string, std::string);
DBinMemory ruleD60(DBinMemory, std::string, std::string, std::string,
                   std::string, std::string, std::string);
DBinMemory ruleD61cong(DBinMemory, std::string, std::string, std::string,
                       std::string);
DBinMemory ruleD61simtri(DBinMemory, std::string, std::string, std::string,
                         std::string, std::string, std::string);
DBinMemory ruleD62(DBinMemory, std::string, std::string, std::string,
                   std::string, std::string, std::string);
DBinMemory ruleD63(DBinMemory, std::string, std::string, std::string);
DBinMemory ruleD64midp(DBinMemory, std::string, std::string, std::string);
DBinMemory ruleD64para(DBinMemory, std::string, std::string, std::string,
                       std::string);
DBinMemory ruleD65coll(DBinMemory, std::string, std::string, std::string);
DBinMemory ruleD65para(DBinMemory, std::string, std::string, std::string,
                   std::string);
DBinMemory ruleD66(DBinMemory, std::string, std::string, std::string,
                   std::string);
DBinMemory ruleD67coll(DBinMemory, std::string, std::string, std::string);
DBinMemory ruleD67cong(DBinMemory, std::string, std::string, std::string,
                       std::string);
DBinMemory ruleD68(DBinMemory, std::string, std::string, std::string);
DBinMemory ruleD69(DBinMemory, std::string, std::string, std::string);
DBinMemory ruleD70(DBinMemory, std::string, std::string, std::string);
DBinMemory ruleD71(DBinMemory, std::string, std::string, std::string,
                   std::string, std::string, std::string, std::string,
                   std::string);
DBinMemory ruleD72(DBinMemory, std::string, std::string, std::string,
                   std::string, std::string, std::string, std::string,
                   std::string);
DBinMemory ruleD73eqangle(DBinMemory, std::string, std::string, std::string,
                          std::string, std::string, std::string,
                          std::string, std::string);
DBinMemory ruleD73para(DBinMemory, std::string, std::string, std::string,
                       std::string);
DBinMemory ruleD74eqangle(DBinMemory, std::string, std::string, std::string,
                          std::string, std::string, std::string,
                          std::string, std::string);
DBinMemory ruleD74perp(DBinMemory, std::string, std::string, std::string,
                       std::string);
DBinMemory ruleD75cong(DBinMemory, std::string, std::string, std::string,
                       std::string);
DBinMemory ruleD75eqratio(DBinMemory, std::string, std::string, std::string,
                          std::string, std::string, std::string,
                          std::string, std::string);
```

```
  DBinMemory ruleB1midp(DBinMemory, std::string, std::string, std::string);
    DBinMemory ruleB1cong(DBinMemory, std::string, std::string, std::string,
                          std::string);


    void testDBim(DBinMemory);
};

#endif
```

strs.hpp

```
/*
 * strs.hpp
 *
 * To deal with lists of strings, whether, e.g. point or predictes.
 *
 * This file is part of the OGP GDDM prover, which, in turn, is part of
 * the Open Geometry Prover Community Project (OGPCP)
 * <https://github.com/opengeometryprover>.
 *
 * Copyright (C) 2022 Nuno Baeta, Pedro Quaresma
 * Distributed under GNU GPL 3.0 or later
 */


#ifndef STRS
#define STRS

struct strsList {
    std::string str;
    struct strsList *next;
};

struct strsList *addStr(std::string, struct strsList *);
void showStrs(struct strsList *);

#endif
```

version.hpp

```
/*
 * version.hpp
 *
 * To keep the versions history
 *
 * This file is part of the OGP GDDM prover, which, in turn, is part of
 * the Open Geometry Prover Community Project (OGPCP)
 * <https://github.com/opengeometryprover>.
 *
 * Copyright (C) 2022 Nuno Baeta, Pedro Quaresma
 * Distributed under GNU GPL 3.0 or later
 */


#ifndef OGPGDDMVERSION
#define OGPGDDMVERSION
```

```
#define VERSION "0.6.1"

/*
 * Version 0.6.1 - 2022/11/30, adicionar a regra B1
 * Version 0.6.0 - 2022/11/30, to show the time to prove the
 * conjecture alongside with the overall time to finish the prover
 * (finding the fix-point)
 * .Version 0.5.5 - ...
 */

#endif
```