MSc Module CS612

# Automated Reasoning
# Post-Course Assignment

# (Assessed Work)

Renate Schmidt Room 2.42
email: schmidt@cs.man.ac.uk
Alan Williams Room 2.107
email: alanw@cs.man.ac.uk

November 2005

## 1 Introduction

You will be required to spend one week undertaking a post-teaching week assignment. This will build on the knowledge you have gained during the teaching week and will enable you to explore in more detail a particular area related to automated reasoning.

The assignment contributes 30% towards your final mark.

The topics available are varied. Some involve further practical work, for example using Vampire, MSPASS or other tools to undertake case-studies in automated reasoning, or writing a Prolog program. Others will require you to produce an essay.

In general, for essay topics you will be provided with starting points for references and information sources. You will then be required to write an essay describing the papers or book chapters you have read. You should include a clear and concise description of the topic, so that it is understandable by someone who has, for example, just completed CS612. The aim of the essay is then to demonstrate what you have found out and understood.

Marks are awarded for clarity in style, original text and examples (i.e. not extracted verbatim from the sources) and thoughtful analysis, especially when comparing different techniques or evaluating the advantages/disadvantages of a method—it is not unknown for research papers to include only 'well-chosen' examples, or to only highlight only the positive characteristics of a technique! A list of references used should be given—any cited material should be acknowledged.

### Learning Outcomes

On successfully completing this assignment, a student will

- have studied in more detail one topic related to the CS612 course unit;
- have had some experience of undertaking a literature search;
- have assimilated new material and demonstrated this by producing a summary essay;
- have further experience of writing (Prolog) programs for automated theorem proving.

# 2 List of Topics

You should pick one project from the following lists—it would also be a good idea to have a second or third choice ready in case your first choice is not available.

You should then inform the appropriate course unit lecturer of your choice, who will inform you if it is available and may also discuss the project with you before you begin.

## 2.1 Renate's Project Descriptions

1. **Implementing the model construction of the model existence theorem.** In the lectures, we described the construction of a model for a given set of clauses relative to a given ordering.

   Develop an algorithm which uses this method and will attempt to construct a model for a given set of clauses and will return the model if the set is saturated and satisfiable or return a minimal counter-example otherwise. Assume the ordering is given by a precedence on the symbols.

   Implement this algorithm in Prolog.

   Consider only the propositional case, or if time is available the case of ground clauses.

   **Reference:** Slides of the lecture course.

2. **Efficient transformation into clausal form.** (2 projects.) Transformation into clausal form is an integral part of resolution theorem proving. When done naively the transformation can have a detrimental effect on the performance of provers. Several optimisations have been developed. These include: structural transformation or renaming, optimisations to the Skolemisation, and input reduction.

   (a) (Essay) The purpose of this assignment is to study a number of optimisation techniques that have been developed and write a report about it.

   (b) (Implementation) The aim is to study a number of optimisation techniques that have been developed and implement them. Good marks will be awarded for a well-design implementation with many optimisation techniques and a good report with good demonstration of the effect of the enhancements.

**Reference:** Nonnengart and Weidenbach [1].

3. **Splitting without backtracking.** (2 projects.) The splitting rule of tableau methods is based on the following property: If $C_1$ and $C_2$ are clauses and $N$ is a set of first-order clauses and if $C_1$ and $C_2$ do not have any variables in common, then $N \cup \{C \vee D\}$ is unsatisfiable iff both

$$N \cup \{C\} \text{ and } N \cup \{D\} \text{ are unsatisfiable.}$$

This property can be exploited by an encoding using the introduction of new propositional symbols. Using an appropriate ordering and selection function the branching effected by the tableau splitting rule can then be simulated in a rolution prover without actually implementing branching.

   (a) (Essay) Study this technique and the use of the refinements to change the behaviour of the prover. Write an essay about it.

   (b) (Implementation) Implement the technique and incorporate it in your propositional resolution prover. Good marks will be awarded for a well-design implementation with a number of refinements, and a good report with good demonstration of the effect of the different refinements.

**Reference:** Riazanov and Voronkov [2].

4. **Resolution decision procedures.** (2 projects.) Testing the satsifiability of problems in first-order logic is in general undecidable. There are however numerous ways of defining fragments of first-order logic for which this problem is decidable. Resolution is well-suited to developing decision procedures for decidable fragments of first-order logic. The purpose of this assignment is to learn and write about resolution decision procedures. The aim is to focus on one of the following subtopics.

   (a) Use of ordered resolution in decision procedures, main reference: Joyner [3].

   (b) Deciding the guarded fragment, main reference: Ganzinger and de Nivelle [4].

**General reference:** Fermüller et al. [5].

5. **Reversing Skolemisation.** Skolemisation is a standard technique for eliminating quantifers from formulae. It introduces constants and Skolem functions during the transformation of first-order formulae into clausal form. When transforming resolution proofs into proofs more easily readable by humans (sequent-style proofs or natural deduction style proofs), Skolemisation needs to be reversed by the reintroduction of first-order quantifiers. Write an essay about transformation of clause form back to first-order logic formulae and in particular the process of reversing Skolemisation.

   **Reference:** Chapter 5 in Engel [6].

6. **Exploring first-order logic theorem provers.** (4 projects.) A variety of first-order logic theorem provers exist. These include:

   (a) **SPASS** – a resolution prover, `http://spass.mpi-sb.mpg.de/`

(b) **Vampire** – a resolution prover. References: Riazanov and Voronkov [7, 8], and CASC website `http://www.cs.miami.edu/~tptp/CASC/`, in particular, the entrant's description list and system's sources and executables of CASC-20.

(c) **E** – a resolution prover, `http://www4.informatik.tu-muenchen.de/~schulz/WORK/eprover.html`

(d) **Darwin** – a theorem prover for the model evolution calculus, a first-order version of DPLL. `http://goedel.cs.uiowa.edu/Darwin/`

Write a report about the chosen prover. This could include an overview of the general approach, special inference rules, transformation techniques, optimization techniques, heuristics, the inference loop, any additional features, applications and different uses. Be careful not to make the report too broad. It is better to focus on a few aspects of the prover and discuss them in detail, giving examples and reporting own experience with the prover.

**References:** as specified above and references available from the websites.

7. **Implement the tableau calculus described in lectures.** In the lectures, a semantic tableau calculus for propositional logic and first-order logic was presented. Implement the calculus for propositional logic. The report should describe the implementation, the design choices made and demonstrate the use of the prover on some non-trivial input. For a distinction your system should include techniques for making it fast which should be discussed in the report, *or*, include in your report a *discussion* of ways of extending the implementation to full first-order logic; time will probably not be enough to do the implementation for the first-order case.

   **Reference:** Slides of the lecture course, Fitting [9].

## 2.2 Alan's Project Descriptions

It will be possible to have more than one student working on the some of following projects.

1. **First Order Resolution in Prolog**: In this practical assignment, you will be able to exercise and extend your knowledge of Prolog, by building a complete first order resolution theorem-prover, extending the components which you were developing in the CS612 laboratory exercises. You should aim to include various simplifications and search strategies and you should test your system on a range of example formulae.

2. **Prolog Programming**: In this assignment, you have the opportunity to develop a larger Prolog program of your choice, building on the automated theorem-proving components that you have already constructed during the CS612 labs.

   However, before choosing this option you must first discuss it with Alan Williams, in order to agree a suitable topic.

3. **A Closer Look at Prolog and Resolution**:

   As we stated during the Teaching Week, this was not a Prolog programming course. However, the Post-Course Assignment gives you an opportunity to study more closely particular aspects of the material presented, by utilising texts and research papers on

Prolog, Logic Programming and Automated Deduction. For example, you could further investigate topics such as negation as failure, simplification, unification, or parallel logic programming.

**References:** Several advanced texts on Prolog cover negation as failure and SLDNF models. Please discuss further with Alan Williams.

4. **Under the Bonnet of a Prolog System**:

   In the CS612 course so far, essentially we have only considered the 'in principle' execution model for Prolog. In this assignment, you will look more closely at the techniques used to implement a typical system, such as SWI-Prolog. You will write an essay suitable for inclusion in CS612 in the future.

   [10, 11]

5. **Your Own Topic** There may be an aspect of the course which you have found interesting and wish to explore further, but which is not covered by one of the above assignments. You may also have related interests not covered. We will therefore try to accommodate you undertaking an assignment on a topic of your own choice.

   However, you must discuss this first with one of the course lecturers (in some circumstances, we may not be able to permit you to undertake it).

# 3 Summary

- Either
  - write an essay describing the papers or book chapters you have read,

  or
  - (if appropriate) describe practical work undertaken.
- The aim of an essay is demonstrate what you have found out and understood within the allocated time.
- Include a clear and concise description of the topic understandable by someone who has, for example, just completed CS612. The use of *your own original*, well-chosen examples is crucial here. You can assume the reader knows the material covered in CS612 so there is no need to repeat definitions, etc from the lecture notes, but reference to what was covered is of course sensible in order to provide the appropriate context and facilitate the necessary understanding of your essay topic.
- Be careful about advertising hype in the literature. Try to understand / evaluate / substantiate any claims made in the literature (e.g. with examples), or be critical as appropriate (e.g. give a counter-argument, counter-examples).
- Projects are intended to occupy one week. Some of them are quite open ended though (don't make the mistake of spending too long on them).

## 3.1 Marking Scheme

The basic aim of the assignment is either to learn in more detail about a particular topic, or to undertake further practical work related to automated reasoning. The essay is evidence of sufficient understanding of the subject, i.e. understanding that can be gained in one week by a student who has attended CS612. A report of practical work would describe what has been undertaken and would include results and conclusions.

The assignment contributes 30% towards your final mark.

**Assessment of Essays:**

Marks will be awarded for:

- clarity in style
- 'original' text and description of topic
- good use of original examples (i.e. not extracted verbatim from the sources)
- a well-constructed essay, i.e. including an introduction, conclusions/summary, a list of references
- acknowledgement of any sources quoted directly, i.e. label citations in the main body of the text
- thoughtful analysis (especially when comparing different techniques or evaluating the advantages/disadvantages of a method)

The following is a *provisional* allocation of marks for an essay-style assignment:

| | |
|---|---|
| Overall style and structure: balanced use of material; well-structured (e.g. including introduction, background, description of technical details, discussion/critique, conclusion, references) | 8 |
| Introduction and background to subject | 5 |
| Description of technical details | 12 |
| Examples: original, relevant, correct, adequately described in order to aid understanding | 10 |
| Conclusions | 5 |
| Critical assessment of subject, original ideas etc | 5 |
| References: relevant; all cited material is acknowledged and referenced | 5 |
| **Total** | 50 |

**Assessment of Practical Work:**

For practical work, marks of course will be awarded for successful completion of the assignment, for example in producing a correctly functioning program, or producing a comparison table of the runtimes of on several benchmarks.

Marks will also be awarded for the presentation of the report of the work. This should include an account of any background or context together with overview and any experimental results. In particular it should include analysis and conclusions.

| | |
|---|---|
| Demonstration (if applicable) | 5 |
| Report | 15 |
| Technical Quality | 30 |
| **Total** | 50 |

Clearly, the quality of the work will be ascertained both from the results described in the report and any demonstration. The mark for the report is for the quality of the report itself, and will be considered on similar grounds as an 'Essay' project.

As with an Essay, the report structure should provide a brief background to, and technical description of the work, highlighting aspects of interest (e.g. challenging areas overcome). A summary should give overall conclusions. Appendices should be included giving technical details (e.g. program code, experimental results, formalisation of the examples).

For a pass the implemented system should satisfy some of the requirements and work correctly; the report should satisfactorily document the work and what has been achieved. For a distinction, your system should fullfil all the requirements, work correctly and include some additional features which improve the performance of the system. The quality of the accompanying report should be of a higher standard possibly including suggestions for improvements supported by convincing explanations.

### 3.2 Important Note on Plagiarism

We need to draw your attention to the requirement for proper referencing. You will most likely be utilising several sources of information in your assignment and will therefore need to convey this in your report. In particular, you *must* ensure that any piece of text or other material taken from another source is identifiable and properly referenced. This means that the material must be clearly quoted, for example by placing it in quote marks or in a quotation paragraph, and then its source must be identified, ideally by including a citation label referring to a list of references given at the end of your report.

**If you do include unacknowledged material, then you are plagiarising the work. The minimum penalty for this is that your work will receive a zero mark.**

**When undertaking a practical exercise and writing code, then this should be your own work.**

If you have any doubts or questions at all regarding how to interpret this then please discuss with the course unit lecturers.

## References


[1] A. Nonnengart and C. Weidenbach. Computing small clause normal forms. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 6, pages 335–367. Elsevier Science, 2001.

[2] A. Riazanov and A. Voronkov. Splitting without backtracking. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 611–617. Morgan Kaufmann, 2001.

[3] W. H. Joyner Jr. Resolution strategies as decision procedures. *J. ACM*, 23(3):398–417, July 1976.

[4] H. Ganzinger and H. de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Fourteenth Annual IEEE Symposium on Logic in Computer Science*, pages 295–303. IEEE Computer Society Press, 1999.

[5] C. Fermüller, A. Leitsch, U. Hustadt, and T. Tammet. Resolution decision procedures. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 25, pages 1791–1849. Elsevier, 2001.

[6] T. Engel. Quantifier elimination in second-order predicate logic. Diplomarbeit, Fachbereich Informatik, Univ. des Saarlandes, Saarbrücken, 1996. `http://www.mpi-inf.mpg.de/~ohlbach/scan/diplom.ps.gz`.

[7] A. Riazanov and A. Voronkov. Vampire. In H. Ganzinger, editor, *Automated Deduction—CADE-16*, volume 1632 of *Lecture Notes in Artificial Intelligence*, pages 292–296. Springer, 1999.

[8] A. Riazanov and A. Voronkov. The design and implementation of vampire. *AI Commun.*, 15(2-3):91–110, 2002.

[9] M. Fitting. *First-Order Logic and Automated Theorem Proving*. Texts and Monographs in Computer Science. Springer, 1990.

[10] Hassan Ait-Kaci. *Warren's abstract machine : A Tutorial Reconstruction*. Available in JRULM.

[11] L. Sterling and E. Shapiro. *The Art of Prolog*. MIT, 1986.