MSc Course Unit COMP6012

Automated Reasoning

Introduction, Timetable and Work Schedule

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

October 2006

1 Course Outline

Logic, the study of reasoning, plays an important role in theoretical computing science and many of the practical areas of computer science such as systems development, computer hardware, data bases, cognitive science, AI and logic programming. For example, in web technologies and agent technologies logical and automated reasoning methods are used for the intelligent processing of large ontologies, decision making based on knowledge bases of structured data, and formal specification and verification of multi-agent systems. In particular an important part of the systems development process concerns reasoning about the behaviour of systems in order to verify the correctness of the behaviour.

Broadly, this course unit aims to provide an introduction to classical logic, automated theorem proving and logic programming. In more detail:

- it considers classical logic (propositional, first-order and clause logic),
- it gives an introduction to theoretical concepts and results that form the basis of current state-of-the-art theorem provers (and other theorem proving tools),
- it discusses local reasoning methods (resolution) but also considers global reasoning methods (tableaux),
- it provides an introduction to logic programming and the relationship to resolution.

These notes provide some information about the course unit. They give a brief outline of the structure and assessment. A list of the topics covered can be found in the COMP60121 syllabus.

1.1 Structure

(note: the structure has changed somewhat from previous years, when course teaching was delivered in a single week, with one week each of pre-course and post-course work)

COMP6012 has five single days of teaching during Weeks 2-6 of Period 1 (Semester 1), together with an amount of ‘off-line’ work time.

The course unit neatly divides into two parts:

1. **Teaching Part I**: Introduction to classical propositional logic. Resolution method and logic programming. Introduction to predicate logic:
   - classical propositional logic
   - associated reasoning systems: truth tables, propositional semantic tableaux
   - normal forms, resolution
   - introduction to logic programming and Prolog
   - predicate logic
   - assessed laboratory exercises
   - assessed and non-assessed paper exercises
2. **Teaching Part II**: advanced resolution theorem proving, semantic tableaux
   - elementary set theory (covered during Part I): sets, set operations, binary relations, functions, properties of binary relations
   - orderings, optimised transformation into clause form
   - ordered resolution with selection, redundancy, refinements
   - candidate models, counter-examples, refutational completeness
   - application to Neuman-Stubblebine key exchange protocol
   - semantic tableaux
   - assessed laboratory exercises
   - assessed and non-assessed paper exercises

1.2 Course Assessment

The course will be assessed as follows:

- 60 % laboratory and coursework (30% Part I, 30% Part II)
- 40 % examination

As you can see there will be various exercises to attempt during the Teaching Period, including exercises which will not be formally assessed but are there in order for you to practice the concepts introduced.

1.3 Deadlines

Deadlines for the coursework are as follows:

- Part I Laboratory Exercise 1 deadline: 2:00pm, Monday 9th October.
- Part I Laboratory Exercise 2 deadline: 12:30pm, Monday 16th October 2006

  (extension to 2:00pm 19th October permitted).
- Part I Resolution Exercises (on paper) should be submitted by: 5:00pm Friday 20th October 2006.
- Part II paper Exercises on Sets (not assessed): 5:00pm Monday 9th October 2006.
- Part II Laboratory and other paper Exercises: Deadlines To Be Announced.

Programs implemented as part of the assignments should be archived using the ‘labmail’ system (see §A.6 below).

Please hand in your written coursework as a hard copy to the Student Support Office, Room 2.3 Kilburn Building.

2 Provisional Teaching Period Timetable and Work Schedule

The COMP60121 Teaching Period comprises Mondays in Period I during Semester 1, i.e. starting on 2nd October, 2006, running during Weeks 2 to 6, as follows:

- 5 Teaching Days: Mondays (Weeks 2-6),
- 5 weeks of 1.5 days ‘off-line’ time, to continue study, including work on laboratory and paper-based exercises (Weeks 2–6),
- 2.5 days within the ‘Coursework Completion’ Week (Week 7).

In the first half of the Teaching Period we will look at logic programming and Prolog, with supporting laboratory exercises. We will study resolution, the proof system underlying Prolog, and will introduce predicate logic.

In the second half of the Teaching Period the emphasis is on the foundation of advanced techniques of automated theorem proving including practical aspects.

Venue:

4th Oct 2006 Note: There have been some recent changes to the timetable and venues, in order to accommodate the larger class:

Unless otherwise stated:

- lectures will be in Room 2.19 (**i.e. moved from Room 2.15**).
- labs will be in MSc Labs (Rooms 2.25a and 2.25b).

2.1 Timetable: Part I (Alan Williams)

Mondays	Session 1 (9:00-10:30)	Session 2 (11:00-12:30)	Lunch	Session 3 (2:00-3:30)	Session 4 (4:00-5:00)
2nd Oct	(AW1) Prop log Intro; (lecture)	(AW2) Prop Res I (lecture)		(AW3) Prop Res II (lect & ex) ****Room IT407****	(AW4) Prolog I (lab)
9th Oct	(AW5) Pred Intro (lecture)	(AW6) Pred Res I (lecture)		(AW7) Pred Res II (lect & ex) ****Room IT407****	(AW8) Prolog II (lab)
16th Oct	(AW9) Log Prog (lecture)	(AW10) Prolog III (lab)		(RS1) see separate timetable	(RS2) see separate timetable
Thursday 19th Oct				(AW11) Prolog lab Marking	

2.2 Coursework Part I (Alan Williams)

One of the aims of these tasks is to give you some practice at using the various components of the reasoning systems studied during the course.

It is important that you attempt them so that you gain hands-on experience of actually using the reasoning systems and seeing how they operate. This will also help you to compare the merits of each system, and find out both some of the advantages and limitations of automated reasoning.

Note also that it is the actual *construction* of the various solutions that is the challenge in each case, and not simply knowing what the solution is, i.e. you should always attempt the yourself.

It will be useful for you to construct proofs for formulae using other reasoning systems and tools (probably (M)SPASS or Vampire) in order to compare the structure of the proofs generated with those you have obtained manually.

Note: Those tasks which are assessed and/or that need submitting are indicated below.

2.2.1 Resources

- “Notes on Automated Reasoning” by Rajeev Goré and Martin Peim [G&P]. This is available separately from the course lecturers.
- The course textbook is “The Essence of Logic” [Kelly], available from Blackwells and also from the Departmental library, where multiple copies are available on short-term loan.
- An online introductory Prolog manual [Endriss] available at:

http://staff.science.uva.nl/~ulle/teaching/prolog/prolog.pdf

- Cormen, T. H. and Leiserson, C. E. and Rivest, R. L. (1990), Introduction to Algorithms, MIT Press [CLR].
- These Introductory Notes.

2.2.2 Part I Tasks for Week 2 (2nd October)

1. **As soon as you can**: please ensure that you can successfully log in to one of the machines in the MSc Laboratory[1]. Once logged in, check that you can start SWI Prolog.

   Changed Instruction[2], but this is not now necessary. If you still encounter an error when starting SWI Prolog involving a failure to load the shared library `libreadline.so.4`, then pleas contact Alan Williams. Simply type:

   ```
   pl
   ```

   If you cannot log in or start Prolog, then please **contact Alan Williams immediately**.

   Also, if you are not very familiar with either Linux or the emacs editor, then please refer to the Appendix of these notes.

2. Read chapters 1 and 2 of [Kelly] and/or sections 1 and 2 of [G&P] in order to revise propositional logic.

   In particular ensure you are familiar with:

   (a) syntax and semantics of Propositional Logic
   (b) logical equivalence, (un)satisfiability, tautology
   (c) the meaning of '$\models$' (logical consequence) and '$\vdash$' (logical deduction), and the difference between these relations

3. For the exercises in sections 2.1 and 4.5 of [G&P], use propositional resolution to prove as many of the propositional formulae as you feel you need to in order to familiarize yourself with this reasoning method.

4. Start to look at the Resolution Exercises (see separate sheet) — you should be able to tackle Question 1 and partly do Questions 2 and 4. These do not yet need to be submitted.

5. Prolog: read §3 of [G&P] and undertake the exercises at the end of the section.

6. **(Assessed)** Complete Prolog Lab Exercise 1 — see separate lab script.

   The work you will do will depend upon whether you have previous experience with Prolog programming.

[1] This is especially important for students not taking the Advanced Computer Science MSc, who may not have undertaken laboratory work in the School of Computer Science before, for example those from the School of Mathematics.

[2] Previously, you needed to add '/opt/prolog/bin' to your '$PATH'

7. Recapitulate the basics of sets, relations and functions from Chapter 5.2 in [CLR]. Also useful are:

   - Chapter 1.1 and 1.3 of "Interactive Real Analysis" (http://www.shu.edu/projects/reals/logic/index.html, Website at Seton Hall University). This website includes interactive exercises with answers.
   - Any book on the mathematical foundations of CS or discrete mathematics.

   WORK TO SUBMIT: Exercise 5.2–3 from Chapter 5 of [CLR]. There are different editions of [CLR] and Exercise 5.2–3 is not the same in all editions. The title of Chapter 5 is "Sets, Etc", the title of Section 5.2 is "Relations". Thus here is the exercise we want you to do.

   > Give examples of relations that are
   >
   > a. reflexive and symmetric but not transitive,
   > b. reflexive and transitive but not symmetric,
   > c. symmetric and transitive but not reflexive.

   Please submit your work to the Student Support Office, Room 2.3.

2.2.3 Part I Tasks for Week 3 (9th October)

1. Study sections 6, 7 and 8 of [G&P], undertaking the exercises at the end of each section.
2. Continue work on the Resolution Exercises (see separate sheet) — you should now be able to complete Questions 1, 3 and 4.
3. Prolog: continue work on Prolog Lab Exercise 2 (see separate document)

2.2.4 Part I Tasks for Week 4 (16th October)

1. Study §9 of [G&P] in order to consolidate your understanding of logic programming, Prolog and SLD-resolution. It would be useful to construct further simple Prolog examples, draw the SLD-trees for these and relate these to their execution traces (as required in Question 2 of the Resolution Exercises).
2. **(Assessed)** Complete and submit the Resolution Exercises.
3. **(Assessed)** Complete Prolog Lab Exercise 2 and get this marked.

A The Bare Necessities

Do not switch off/reset/reboot your machine while it is in the middle of running Windows or Linux. This may cause damage! If in doubt, please ask

A.1 Introduction

For the laboratory exercises we are assuming that you will be reasonably familiar with Linux, and that you will be able to make simple use of the emacs editor.

Hopefully this is the case for most students. However, we only need you to use very basic facilities of each, so if you are not too familiar with them then you should be able to learn the basics quite quickly. The following is therefore some brief notes to help you get started. Please contact the course lecturers if you have problems or questions.

A.2 To Log In

- Make sure the machine is running linux (they are dual-boot, so you might need to shut down Windows and reboot). Type your userid and password when prompted. Select the window manager, by typing '`X`'. If you have not already done so, change your password by typing `yppasswd` in an xterm window
- Start an 'xterm' window by clicking the left mouse button on the background and selecting 'New Shell'.

A.3 Basic Text Editing in Emacs

Emacs is an extremely powerful editor with an incredible number of facilities. Yet it is relatively straightforward to start using it. In any case, in the CS612 labs, we will only need to use a very limited number of features.

- Start the emacs editor by typing `emacs &` in the 'xterm'.
- You can exit emacs by typing '`C-x C-c`', where '`C-x`' means hold down the Control or '`Ctrl`', key and typing '`x`' at the same time.
- You can save a file by typing '`C-x C-s`'. A prompt will appear at the bottom of the editor screen for the file name, if you have been editing a new file.
- You can load a file by typing '`C-x C-f`'. You will be prompted for the file name.

  The file will be loaded into a new buffer. Any changes are not made until you save the file. You can edit several buffers at once, and switch between them, by typing '`C-x b`'
- You can type in text by, well, simply typing!
- You can delete text using the `Delete` key or the '←' key

- You can navigate around some text using the arrow keys, `Page Up` and `Page Down` keys.
- You can select a region of text using the mouse, dragging it over the text while holding down the left mouse button. You can copy the text by typing '`M-w`'. You can then paste this text by typing '`C-y`'.

You can find out more about emacs by trying the excellent on-line tutorial, which is available by typing '`C-h t`'

The main reason for learning emacs (apart from it being the best editor by far!) is so that you can use the proof support tools associated with the Lego proof system. You will receive separate instructions via email on how to use these.

A.4 Linux Commands and Logging Off

- **Logging Out**: you can log out by clicking the left mouse button on the background, then select '`Exit Fvwm`' then '`Yes, really quit`'. This will stop your window manager and will also log you out of the machine. You should see the login prompt appear.
- to add '`/opt/prolog/bin`' to your '`$PATH`' type:

  ```
  export PATH=/opt/prolog/bin:$PATH
  ```
- to start the SWI Prolog system type '`pl`'
- **Creating directories:** use the command '`mkdir`'. For example to create a directory called `COMP60121`, type '`mkdir COMP60121`' in an xterm window
- **Moving between directories:** you can move into sub-directory called `COMP60121` of the current directory by typing '`cd COMP60121`'
- You can move up from a sub-directory to its parent by typing '`cd ..`'. To return to your home directory, type '`cd`'.
- **Listing files:** you can list the contents of a directory by typing '`ls -l`'
- You can rename a file called `foo.pl` to call it `bar.pl` by typing '`mv foo.pl bar.pl`'
- **Getting help**: there is an excellent on-line help system with Linux. For example, you can get help on the above `mv` command by typing '`man mv`'

However, if the above is not familiar to you, then you need to find out more about Linux! There is an excellent introduction to School's Linux system available at

`http://www.cs.manchester.ac.uk/Study_subweb/Ugrad/coursenotes/Intro/notes/`

A.5 Saving Output from Prolog

You will need to save the output to a file in various places from the Prolog system.

Apart from using Prolog commands to write output, you can collect standard output in a number of ways — here are two:

- Run SWI Prolog from within a shell in Emacs. To do this, start a 'shell' in emacs by typing '`M-x shell`', followed by the Return key, where '`M-x`' means use the '`Esc`' key followed by typing the letter 'x'. A 'shell' should start, in which you can run commands, including the Prolog system.

  You can then edit the standard output which is now within the '`*shell*`' emacs buffer, and save this to a file of your choosing.

- Use the unix '`tee`' command; for example to copy standard output to the file '`prolog-trace.txt`', type:

```
pl | tee prolog-trace.txt
```

A.6 labmail and ARCADE

We will require you to archive your labwork using the 'labmail' system. This should be straightforward to use.

Assuming you are an xterm in Linux, simply change to the sub-directory for the exercise you wish to archive, e.g. '`cd $HOME/COMP60121/ex1`' and then type:

```
labmail
```

Labmail will warn you if it doesn't recognise the directory names. It looks for the file needed by the exercise, and will warn you if it's not there. It emails a copy of your work to an archive, which serves as a backup record for us, so if all else fails, we should find a dated copy of your work there. To make sure we have this record, the rule is that **your mark will not count if you have not labmailed your program** to the archive.

We will also be using ARCADE to gather attendance and marks information. This will inform you if you have not labmailed your work.