Course 480 Automated Reasoning (Parts I and II)
Lecturer: Krysia Broda (kb@imperial.ac.uk) Room 378
Synopsis of Course 2013/14

Automated Reasoning is used in many aspects of Computer Science, which include, but are not limited to:

logic programming languages; reasoning about hardware design;
reasoning about program correctness and program specifications;
automating logical proofs; problem solving and mathematical reasoning;
model checking;

This course is concerned with the fundamentals of Automated Logical Reasoning. By the end of Parts I and II of the course you will have a good knowledge of various techniques used in theorem proving for propositional logic, first order clausal logic, general first order logic and equational logic, including rewrite rules, as well as practical experience of using the Otter family of theorem provers and some experience of Chaff (Davis Putnam procedure). You will also be aware of various properties expected or desired of theorem provers and the methods used to prove them. Armed with this knowledge you should be able to understand, or at least get the gist of, both historical (ie since about 1960) and state-of-the-art papers in Automated Reasoning, whether for classical logic or non-classical logics, for example modal logics of various kinds.

Each group of lectures will cover a particular theme, briefly described next (note that the Slidesets are not of uniform length). Generally, two slidesets correspond to approximately one week of lectures. Most slidesets include some additional, completely optional, material for interest.

1. *(Slidesets 0 and 1)* A brief overview. The Davis Putnam technique for Clausal Propositional logic is particularly efficient for clausal propositional data. The Davis Putnam procedure was invented in the early 60s and it is still the preferred method for Reasoning with this kind of data. Although implementations have been substantially improved since 1960 (!), the basics remain unchanged. You can try out some propositional provers (eg Chaff). (There is no slideset 2!)

2. *(Slidesets 3-4.)* The main technique used for first order logic is Resolution, invented by Alan Robinson in 1964. There have been many refinements of the original procedure and new ones continue to be implemented to this day. Resolution techniques used in the basic theorem provers have been adapted for dealing with modal logics, and a good knowledge of resolution is essential in order to understand the newer ideas.

3. *(Slideset 5.)* Much of the research in resolution techniques is concerned with soundness and completeness of the refinements. For instance, it is important that a theorem prover should deliver correct results. If it purports to show that

some statement follows from some given information, then that should be the case (Soundness). However, occasionally, a prover will be unable to show that this is so, even though it is. Such a theorem prover is still useful, as long as it manages to prove some things of course! So Completeness, whilst very handy, is not as essential as Soundness. We will look at the way Robinson showed these things for resolution, as well as some other approaches.

4. *(Slidesets 6-8.)* After discussing the basics of resolution, including clausal form logic (which is a convenient normal form in which data is written as disjunctions of atomic data), we look at some of the many refinements that are typical of those in use in the main provers today. In particular you get practical experience of Otter. Although Otter has been improved and replaced by Prover9, Otter is better for learning as it gives a user the chance to control choices.

5. *(Slidesets 9-11.)* A different technique for theorem proving, which can be applied to any data, whether in clausal form or not, uses the idea of semantic tableaux. Although tableaux were invented in 1954 by Beth, they were not automated until 1985-6. But since then, as with resolution, there have been many improvements of the basic method. In fact, many of the refinements were inspired by resolution refinements. Tableau techniques can be programmed very conveniently in Prolog, and new ideas can easily be tried in prototype programs. Tableau methods are often extended and adapted for non-classical logics, such as modal or temporal logic, linguistic logics, etc. It turns out that some very efficient methods can be programmed in case clausal form is used; the technique is called Model Elimination and we investigate this methodology in some detail.

6. *(Slidesets 12 and 13.)* Data may well include use of the equality predicate. For example, I might wish to record that the person I know as Joe Bloggs is, in fact, the prime minister! This might be encoded as JB=PM. Another example is the statement that 1+1=2, or x+x=2*x, for every integer x. Reasoning with equality can greatly increase the search space and special techniques have been devised to control this increase. We cover the paramodulation operation and its incorporation into both resolution provers and tableaux.

7. *(Slidesets 14-17.)* When the data consists only of equations other approaches are possible, including the use of rewriting. The last part of the course covers this and explains the role of the Knuth Bendix Method and rewriting in order to show that two terms are equal modulo some equations. You will be able to use a tool (written by an MSc student in 2010) to guide you on this topic.

Part III is a case study looking at Reasoning in Ontologies and will be given by Graham Deane. Graham will issue his own notes.