EDA132/DATE15, Artificial Intelligence, vt09
Assignment 3: Deduction

Jacek Malec, Computer Science, LU
`jacek.malec@cs.lth.se`

February 24, 2009

Deadline

The report must be handed in for examination before 23:59, Thursday April 2nd, 2009. The report should be put in the box labeled TAI (EDA132 resp. DATE15) in the stairwells outside the secretary offices (LTH resp. NatFak). If you have any questions, mail me (`jacek.malec@cs.lth.se`) or ask them personally.

1 Introduction

In one sentence: you are expected to implement a generic automated proof procedure for propositional logic. If you wish, you may proceed by extending your implementation with some interesting features, like search control, that will make it more efficient and possibly will allow you to prove more interesting theorems.

The following set of tasks is intended to let you master your understanding of the simplest of logical languages used for knowledge representation and reasoning, and the pitfalls of mechanically implementing it. We begin with the language of propositional logic - simple, decidable, but not very expressive, and continue with its extensions interesting from the AI point of view. Although we will limit ourselves to the propositional case, you will have an opportunity to experiment with a number of formalisms, in particular with classical and modal logics. The non-compulsory part of the exercise lets you investigate more advanced issues related to automated reasoning based on logic.

Learning how to use logic is not an easy task. Learning it without making exercises is an impossible task. Therefore I would like to recommend you to solve the exercises in the textbook, at least from Chapter 7 and maybe some more, before attempting to solve the following problems.

In order to get the Assignment passed you need to be able to handle one formal system defined in the Axiomatization section, namely Russell and Whitehead's. In order to get higher grade than pass (4 or 5, alt. VG) you need to

1. implement some speed-up strategy for proof search, describe it in the report, present the results of the speed-up and motivate its use; and
2. address at least one more system defined in the Axiomatization section.

As usual, you should file a written report (paper copy, please note that email submissions are not accepted) describing your solution(s) in sufficient detail to understand it and possibly copy it. You should provide extracts from test runs of your software. You should explicitly address the questions asked in the text of the Assignment. Finally, you should provide the location of your software, both the source code and the executable, so that I may find it and run it without problems. Please DO NOT include the code in your report, unless you want to illustrate some specific topic in the text — let's save some trees.

Even if you do not intend to solve more than the basic task (Russel and Whitehead), please read this document to the end: there are some hints and suggestions in the final section that might appear useful while solving the problem.

2 Motivation

As this question has been asked a number of times before, it should probably be answered straight away: Why such task and what do I expect you to learn while solving it? Let me begin with saying what I don't want you to achieve: an efficient theorem prover for propositional logics of various sorts. This problem has been studied for many years and to achieve a decent outcome one needs to exploit theoretical results well beyond what we can cover here in this course. I rather see it as a paradigmatical one for the whole field of AI: a hard (NP-complete or worse) search problem, computationally hopeless without using heuristics. In a sense, I want you to feel this hopelesness by trying to solve the problem in a straightforward way, and then realise that a heuristics, even a simple one, is always better than nothing (provided, of course, that it is introduced in a reasonable way). So the outcome of this assignment is not any particular insight into how theorem proving should be done, but rather into the need of heuristics in any non-trivial search, and an attempt to create one out of the problem descripton, without knowing much more about the domain. I expect you to think out some trivial (in the sense quite obvious) heuristics, to pick one that seems reasonable and to compare it with the brute force approach.

3 The obligatory part of the assignment

Theorem proving has been considered a challenge for AI already at the birth of this area of research. The first program to prove theorems, LOGIC THEORIST, has been written in 1957 by Newell, Shaw and Simon (n.b., Simon was to become later the Nobel Prize winner) and tested on the propositional calculus formal system, as defined in the book of Russell and Whitehead *Principia Mathematica*. You may find more information about this program, together with some interesting considerations of proof strategies, in the book edited by Feigenbaum and Feldman *Computers and Thought*, available in our library. You are expected to make now exactly the same breakthrough in AI research!

3.1 Russell and Whitehead's Axiomatisation

As we have said earlier, the language of propositional logic consists of a number (in principle arbitrary) of propositional variables (denoted here as p, q, r, p_1,

$p_2, \ldots$), some logical connectives ($\neg$ standing for negation, $\vee$ for disjunction, $\wedge$ for conjunction, $\rightarrow$ for implication, $\leftrightarrow$ for equivalence, and possibly more) and parentheses. A *well-formed formula* (wff) of the propositional calculus is built according to the following rules:

1. Every propositional variable is a wff;
2. If A and B are wffs then $(\neg A), (A \vee B), (A \wedge B), (A \rightarrow B)$ and $(A \leftrightarrow B)$ are wffs;
3. All wffs are built according to the rules 1 and 2 above.

We need only a few connectives to provide a minimal language — the remaining ones may be defined using the primitive ones. E.g., if we choose $\neg$ and $\vee$ as the primitive logical connectives, then we can define others as follows:

$$(A \rightarrow B) \overset{df}{=} ((\neg A) \vee B) \tag{1}$$
$$(A \wedge B) \overset{df}{=} (\neg((\neg A) \vee (\neg B))) \tag{2}$$
$$(A \leftrightarrow B) \overset{df}{=} ((A \rightarrow B) \wedge (B \rightarrow A)) \tag{3}$$

To get a formal system we need a set of axioms and a number of rules of inference. Principia Mathematica assumed the following axioms:

$$((A \vee A) \rightarrow A) \tag{4}$$
$$(A \rightarrow (B \vee A)) \tag{5}$$
$$((A \vee B) \rightarrow (B \vee A)) \tag{6}$$
$$((A \vee (B \vee C)) \rightarrow (B \vee (A \vee C))) \tag{7}$$
$$((A \rightarrow B) \rightarrow ((C \vee A) \rightarrow (C \vee B))) \tag{8}$$

It may be surprising, but these five axioms suffice to obtain all the theorems of the propositional logic.

The rules of inference assumed in Principia are as follows:

Substitution In any theorem, any propositional variable may be substituted by any wff, provided that all the occurrences of this variable are substituted.

E.g., Prove $((p \vee q) \rightarrow (q \vee (p \vee q)))$.

1. $(A \rightarrow (B \vee A))$ (Axiom 5)
2. $((p \vee q) \rightarrow (B \vee (p \vee q)))$ Substitution $A \Rightarrow (p \vee q)$
3. $((p \vee q) \rightarrow (q \vee (p \vee q)))$ Substitution $B \Rightarrow q$

End of the proof!

Definition substitution Every connective may be substituted by its definition and vice versa.

E.g., Prove $((\neg p) \vee (q \vee p))$.

1. $(A \rightarrow (B \vee A))$ (Axiom 5)
2. $(p \rightarrow (B \vee p))$ Substitution $A \Rightarrow p$

3. $(p \to (q \lor p))$ Substitution $B \Rightarrow q$

4. $((\neg p) \lor (q \lor p))$ Definition Substitution $(A \to B) \Rightarrow ((\neg A) \lor B)$, with $A = p$ and $B = (q \lor p)$.

End of the proof!

Modus Ponens If A and $A \to B$ are theorems then B is a theorem as well.

E.g., Prove $((\neg p) \lor p)$.

1. $((A \to B) \to ((C \lor A) \to (C \lor B)))$ Axiom 8
2. $(((p \lor p) \to B) \to ((C \lor (p \lor p)) \to (C \lor B)))$ Substitution $A \Rightarrow (p \lor p)$
3. $(((p \lor p) \to p) \to ((C \lor (p \lor p)) \to (C \lor p)))$ Substitution $B \Rightarrow p$
4. $(((p \lor p) \to p) \to (((\neg p) \lor (p \lor p)) \to ((\neg p) \lor p)))$ Substitution $C \Rightarrow (\neg p)$
5. $((A \lor A) \to A)$ Axiom 4
6. $((p \lor p) \to p)$ Substitution $A \Rightarrow p$ in 5.
7. $(((\neg p) \lor (p \lor p)) \to ((\neg p) \lor p)))$ Modus Ponens 6 and 4
8. $((p \to (p \lor p)) \to ((\neg p) \lor p)))$ Definition substitution for $\to$
9. $(A \to (B \lor A))$ Axiom 5
10. $(p \to (B \lor p))$ Substitution $A \Rightarrow p$ in 9.
11. $(p \to (p \lor p))$ Substitution $B \Rightarrow p$ in 10.
12. $((\neg p) \lor p))$ Modus Ponens 11 and 8.

End of the proof!

As you can see, even apparently simple theorems may require pretty long proofs. But a principled application of the three inference rules together with the axioms (and previous theorems) allows one to prove **every** theorem of the propositional logic.

If you are interested in more examples of proofs in this system then please consult the following web site: `http://www.qedeq.org/`, and in particular `http://www.qedeq.org/propositional.html`.

3.2 Your task:

Ask your program to prove the following theorems (their numbers come from the original *Principia Mathematica*:

$$((p \to (\neg p)) \to (\neg p)) \quad (2.01)$$
$$(p \to (p \lor q)) \quad (2.02)$$
$$((p \to q) \to ((q \to r) \to (p \to r))) \quad (2.06)$$
$$(p \to (p \lor p)) \quad (2.07)$$
$$(p \to (\neg(\neg p))) \quad (2.12)$$
$$(((\neg p) \to q) \to ((\neg q) \to p)) \quad (2.15)$$
$$((p \to q) \to ((\neg q) \to (\neg p))) \quad (2.16)$$
$$(((\neg p) \to (\neg q)) \to (q \to p)) \quad (2.17)$$
$$((p \lor (q \lor r)) \to ((p \lor q) \lor r)) \quad (2.31)$$
$$((\neg(p \lor q)) \to (\neg p)) \quad (2.45)$$

Please make sure that your theorem prover numbers the lines of the proof and states what rule has been used (and how, if that matters).

You may get some problems with some theorems - they require rather large search spaces. If necessary, you might already need to introduce some heuristics that cut the search tree appropriately. Or maybe backward chaining would help? How does your pattern matching procedure (used for substitution finding) look like? Can you improve the search by choosing a more "intelligent" matching strategy?

Please note that you are allowed to introduce previously proven theorems as axioms in the next proof. E.g., when attempting to prove (2.31) you are allowed to use (2.01)–(2.17) (or other theorems you might have obtained on the way) as axioms.

Some more theorems to test your system with (proper handling of double negation is a good sign):

$$(p \to p)$$
$$(p \lor (\neg p))$$
$$(p \to (\neg(\neg p)))$$
$$(p \lor (\neg(\neg(\neg p))))$$
$$((\neg(\neg p)) \to p)$$

A bit more advanced theorems (if you would like to do more experiments) can be found at `http://www.qedeq.org/propositional.html` in the documents `prophilbert1`, `prophilbert2`, and `prophilbert3` (a local copy may be found on the course page, `http://www.cs.lth.se/EDA132`).

4 The non-obligatory part of the assignment

Below you will find five more formal systems of propositional logic. For better grade make sure that your program can handle at least one of them. Actually, if it can, then it should be able to handle all of them.

4.1 Hilbert and Ackermann

The axioms from the previous section are exchanged by the following ones:

$$((A \lor A) \to A) \tag{9}$$
$$(A \to (A \lor B)) \tag{10}$$
$$((A \lor B) \to (B \lor A)) \tag{11}$$
$$((A \to B) \to ((C \lor A) \to (C \lor B))) \tag{12}$$

You may note that the difference consists of omitting Axiom 7 and slightly changing Axiom 5. Can your theorem prover show all the original theorems? Please note that for this to hold you just need to prove the Axioms 5 and 7. If you consult `http://www.qedeq.org/propositional.html` then you might see that the proof of Axiom 7 requires about 60 steps. Can your program handle that?

4.2 Mendelson

In this system the only primitive connectives are $\neg$ and $\to$. Disjunction and conjunction are defined as follows:

$$(A \vee B) \overset{df}{=} ((\neg A) \to B) \tag{13}$$

$$(A \wedge B) \overset{df}{=} (\neg(A \to (\neg B))) \tag{14}$$

and the axioms are

$$(A \to (B \to A)) \tag{15}$$

$$((A \to (B \to C)) \to ((A \to B) \to (A \to C))) \tag{16}$$

$$(((\neg B) \to (\neg A)) \to (((\neg B) \to A) \to B)) \tag{17}$$

The first exercise in Mendelson's textbook is to prove the following theorems, so your prover might as well begin with these:

$$(((\neg p) \to p) \to p) \tag{18}$$

$$(((\neg p) \to (\neg q)) \to (q \to p)) \tag{19}$$

Try to prove other theorems as well. E.g., you may try to prove the axioms of the other systems, or Russell and Whitehead's theorems from Principia. Remember that the definition substitution has to be changed appropriately.

4.3 Kleene

In this system the primitive connectives are $\neg$, $\wedge$, $\vee$ and $\to$. The only defined connective is equivalence.

The axioms in the Kleene's system are

$$(A \to (B \to A)) \tag{20}$$

$$((A \to (B \to C)) \to ((A \to B) \to (A \to C))) \tag{21}$$

$$((A \wedge B) \to A) \tag{22}$$

$$((A \wedge B) \to B) \tag{23}$$

$$(A \to (B \to (A \wedge B))) \tag{24}$$

$$(A \to (A \vee B)) \tag{25}$$

$$(B \to (A \vee B)) \tag{26}$$

$$((A \to C) \to ((B \to C) \to ((A \vee B) \to C))) \tag{27}$$

$$((A \to B) \to ((A \to (\neg B)) \to (\neg A))) \tag{28}$$

$$((\neg(\neg A)) \to A) \tag{29}$$

Can you prove the theorems stated earlier using this system? Remember that the definition substitution has to be appropriately limited here even further.

Just for fun, completely outside the scope of the assignment: You might wish to know that if we replace the axiom $((\neg(\neg A)) \to A)$ with $((\neg A) \to (A \to B))$ then we obtain so called *intuitionistic logic*, behaving slightly differently than the standard one. The intuitionistic logic has very big importance in philosophy. Can you prove some intuitionistic theorems? Which of the previous theorems do not hold any longer?

4.4 Rasiowa and Sikorski

In this system the primitive connectives are $\neg$, $\wedge$, $\vee$ and $\rightarrow$. The only defined connective is equivalence.

The axioms in the Rasiowa and Sikorski's system are

$$((A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))) \tag{30}$$
$$(A \rightarrow (A \vee B)) \tag{31}$$
$$(B \rightarrow (A \vee B)) \tag{32}$$
$$((A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow ((A \vee B) \rightarrow C))) \tag{33}$$
$$((A \wedge B) \rightarrow A) \tag{34}$$
$$((A \wedge B) \rightarrow B) \tag{35}$$
$$((C \rightarrow A) \rightarrow ((C \rightarrow B) \rightarrow (C \rightarrow (A \wedge B)))) \tag{36}$$
$$((A \rightarrow (B \rightarrow C)) \rightarrow ((A \wedge B) \rightarrow C)) \tag{37}$$
$$(((A \wedge B) \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C))) \tag{38}$$
$$((A \wedge (\neg A)) \rightarrow B) \tag{39}$$
$$((A \rightarrow (A \wedge (\neg A))) \rightarrow (\neg A)) \tag{40}$$
$$(A \vee (\neg A)) \tag{41}$$

Can you prove the theorems stated earlier using this system? Remember that the definition substitution has to be appropriately limited here.

Rasiowa and Sikorski's system leads us easily to the formalization of so called *modal logic*, equipped with one more unary connective.

4.5 Modal logic

We begin with extending our language with one more connective, namely $\Box$ (sometimes read as "necessarily", often referred to as simply "box", but sometimes interpreted as "it is known that".) The definition of a wff is now as follows:

1. Every propositional variable is a wff;
2. If A and B are wffs then $(\neg A), (\Box A), (A \vee B), (A \wedge B), (A \rightarrow B)$ and $(A \leftrightarrow B)$ are wffs;
3. All wffs are built according to the rules 1 and 2 above.

The set of rules of inference is extended by the following one: If $(A \rightarrow B)$ is a theorem then $((\Box A) \rightarrow (\Box B))$ is a theorem as well.

Finally, we extend the set of axioms (30)–(41) with the following ones:

$$(((\Box A) \wedge (\Box B)) \rightarrow (\Box(A \wedge B))) \tag{42}$$
$$((\Box A) \rightarrow A) \tag{43}$$
$$((\Box A) \rightarrow (\Box(\Box A))) \tag{44}$$
$$(\Box(A \vee (\neg A))) \tag{45}$$

Can you prove $((\Box(A \wedge B)) \rightarrow ((\Box A) \wedge (\Box B)))$, one of the most famous theorems in elementary modal logic (sometimes taken as an axiom instead)?

Some modal logic theorems for testing purposes (with $(\Diamond A)$ defined as $(\neg(\Box(\neg A)))$):

$$(((\Diamond A) \to (\Box B)) \to (A \to B)) \tag{46}$$
$$(A \to (\Diamond A)) \tag{47}$$
$$((\Box A) \to (\Diamond A)) \tag{48}$$
$$((\Box(A \wedge B)) \to ((\Box A) \wedge (\Box B))) \tag{49}$$
$$((\Diamond(A \vee B)) \to ((\Diamond A) \vee (\Diamond B))) \tag{50}$$
$$((\Diamond(A \vee B)) \to (\Diamond A)) \tag{51}$$
$$(((\Box A) \vee (\Box B)) \to (\Box(A \vee B))) \tag{52}$$

The utility of modal logic lies in its enormous potential for applications. You can read $\Box A$ as "I know A", or "The robot knows A" — the necessity operator becomes the "knowledge" operator, and suddenly you may reason about your own (or your robot's) knowledge, or lack thereof! Another possibility is to interpret $\Box$ as the "Always in the future" operator, and you get a temporal logic instead. The applications are multiple, in particular within the more theoretically-founded AI. But we will leave this topic out here. For more details please consult the literature, which is enormous.

5 Final notes. Important!

- One of the most popular questions I have got last year was "How should I represent formulae in my program"? Of course, there are lots of possible choices, starting with pure strings, for example. A representation that might happen to be particularly useful is probably the *abstract syntax tree* (AST) of the logical formula. Remember, however, that you have to manipulate original formulae, without transforming them to any of the "normal forms" (like CNF or DNF).

- The simple solution would already need to make some simple choices: e.g., substitution may create infinite branching in the search tree!

- The simple solution would have methods for performing substitutions, definition substitutions and modus ponens. Then it would iterate over them, growing the set of formulae (and storing the history in order to print the proof at the end).

- Putting more than three days of work into this is an overkill!

- You may find examples of previous solutions (reports) and other suggestions at the page `http://www.cs.lth.se/home/Jacek_Malec/eda132/`.

- Result to be handed in (for each task you chose to solve, except if stated otherwise):

  1. A presentation of the assignment.
  2. A presentation of your implementation and the user interface.
  3. The results of your investigations and your reflections.

4. Information of where to find your source and executables and how to run the executable.

- The report must be handed in on paper, neatly written in Swedish or English. Neither e-mail nor references to web pages will be accepted. The resulting programs should remain intact in your directory until you have been notified of the result, e.g., on the notice board or by e-mail. You may expect that your report and implementation will be examined within two weeks. If your report or implementation is unsatisfactory you will be given *one* chance to make the corrections and then to hand it in within a week after you have been notified (on the notice board and/or by e-mail). Your final report will be kept until your final grade of the course has been determined. You may then retrieve it if you wish.
- Your program must be implemented and runnable at the department's UNIX machines (e.g. `login-1.student.lth.se`). Remember to make your programs and all the directories in their path read and execute accessible to 'others' (`chmod 705` *filename*).

Have Fun!

Acknowledgement

The formulation of this assignment has been much improved thanks to comments of Hans Gylling.