

## CS 2502/486 2008 Assignment 2

Due: November 7

**Note:** Question 4 below requires programming. Further instructions will be provided in a tutorial.

1. **[40 pts]** In class we saw an efficient procedure for determining the satisfiability of a set of propositional Horn clauses. In this question, we generalize the procedure. We call a set of clauses *generalized Horn* or *GHorn* for short, if we can flip the polarity of some of the atoms and produce a set of Horn clauses. For example, the clauses  $\{[\bar{p}, \bar{q}, \bar{r}], [p, q, \bar{r}], [\bar{p}, s, r]\}$  are GHorn since if we flip  $q$  and flip  $s$  we get a set of Horn clauses. A naive procedure for deciding the satisfiability of a set of GHorn clauses with  $n$  atoms would be to go through all  $2^n$  possible sets of flips until one was found to be Horn, and then use Horn satisfiability. Here we consider a much more efficient procedure:

**Input:** a finite set  $S$  of propositional clauses

**Output:** yes or no

- (1) At the start, no literal in  $S$  is marked.
- (2) If there is a clause in  $S$  such that the complement of every literal in that clause is marked, return no.
- (3) Otherwise, if there is a clause in  $S$  that contains a literal  $m$  that is not marked, but such that the complement of every other literal in the clause is marked, then mark  $m$  and go to step 2.
- (4) Otherwise, return yes.

Observe that the procedure returns **yes** iff every clause either has a literal that is marked whose complement is not marked or contains at least two literals whose complements are not marked.

- (a) Give an example of a (non-GHorn) set of clauses where this procedure does not correctly determine if the clauses are satisfiable.
  - (b) Explain why this procedure always terminates in time that is polynomial in the size of  $S$ . (Aside: with some care in the use of data structures, it can be made to run in linear time overall.)
  - (c) Prove that a GHorn set of clauses where every clause has at least 2 literals must be satisfiable. *Hint:* consider the Horn clauses after flipping and the interpretation that makes all atoms false.
  - (d) Let  $S$  be a GHorn set of clauses and  $m$  be any literal that is marked at some step of the above procedure. Let  $S \bullet m$  be defined as in Question 5 of Chapter 4. Prove that  $S$  is satisfiable iff  $S \bullet m$  is satisfiable. *Hint:* Use induction on the number of iterations it takes to mark the literal.
  - (e) Use the observation above and parts (b), (c) and (d) to prove that for any GHorn set of clauses, the procedure returns **yes** iff the set is satisfiable.
2. **[30 pts]** This question also concerns generalizing the forward chaining procedure for Horn clauses, but this time, to include *negation as failure*, as described at the start of the Exercises of Chapter 6 in the book. In this case, instead of marking literals as above, we mark atoms with Y or N.
    - (a) Question 1a) of Chapter 6.
    - (b) Question 1b) of Chapter 6.
    - (c) Consider the following iterative variant of the forward chaining procedure:

1. If all atoms are labelled then exit.
2. Otherwise, run the forward chaining procedure until it stops.
3. Choose some atom that is not labelled, and label it N.
4. Go to step 1.

Give an example of a KB where simple forward chaining fails, but where this iterative version works correctly. Give another example of a KB where this procedure does not do the appropriate thing.

3. **[30 pts]** In this question, we will use production rules to solve a *blocks-world problem*, where we have stacks of blocks on a table, and it is possible to move the block at the top of a stack to the top of some other stack. Specifically, imagine we have three stacks, *A*, *B* and *C*, and an equal number of red and blue blocks, all located initially in stack *A*. Our goal is to move them so that the stacks *A* and *B* are empty and the blocks in *C* alternate in colour, with red at the bottom and blue at the top.

Assume we have a working memory with elements of the following form:

- (STACK name:*x* top:*y*), where *x* is A, B or C and *y* is either the name of a block, or the special constant TABLE (when the stack is empty).
- (BLOCK name:*x* colour:*y* loc:*z*), where *x* is the name of a block, *y* is either RED or BLUE, and *z* is either the name of block (when *x* is on *z*) or TABLE (when *x* is the bottom of a stack).
- (START) or (DONE) to indicate an initial or final state.

So if the blocks in stack *A* at the start are blue, blue, red, and red (from top to bottom) our initial working memory would consist of

```
(START)
(STACK name:C top:TABLE)
(STACK name:B top:TABLE)
(STACK name:A top:B1)
(BLOCK name:B1 colour:BLUE loc:B2)
(BLOCK name:B2 colour:BLUE loc:B3)
(BLOCK name:B3 colour:RED loc:B4)
(BLOCK name:B4 colour:RED loc:TABLE)
```

and the final working memory after 6 block moves should consist of

```
(DONE)
(STACK name:A top:TABLE)
(STACK name:B top:TABLE)
(STACK name:C top:B1)
(BLOCK name:B1 colour:BLUE loc:B4)
(BLOCK name:B4 colour:RED loc:B2)
(BLOCK name:B2 colour:BLUE loc:B3)
(BLOCK name:B3 colour:RED loc:TABLE)
```

Production rules can of course delete, modify and add to these lists. In addition, rules can add a WME of any other form desired to keep track of information as the problem is being solved, so long as they clean up after themselves at the end.

Write a collection of production rules that will transform a legal initial state as above into a goal state by a sequence of legal block moves. Your rules should work for any number of red blocks with an

equal number of blue ones. State what conflict resolution strategy you are assuming, including what happens if more than one WME can match a rule. Show the execution of your rules when the initial stack  $A$  from the top is blue, red, blue, red. (Five block moves are required.)

4. [60 pts] Question 4a) and 4b) of Chapter 9.

5. [20 pts] Question 3 of Chapter 9.

6. [20 pts] Consider the following assertions:

Henry is a course instructor.

Henry is a university person.

A university person is typically a student.

A course instructor is typically not a student.

A student is typically young.

A university person is typically not young.

(a) Represent the assertions in an inheritance network.

(b) What are the two preferred extensions of the network with respect to Henry?

(c) Would a skeptical reasoner conclude that Henry is young? How about an ideally skeptical reasoner?