CSC 486/2502, 2000 – Assignment 2

Due: Friday, November 10

---

1. **[10 Marks]** (Resolution)
   Determine whether the following sentence is a theorem of first-order logic using resolution:

$$\exists x.\forall y.\forall z.((P(y) \supset Q(z)) \supset (P(x) \supset Q(x)))$$

2. **[25 Marks]** (Negation as Failure)
   This question involves generalising Horn derivations to incorporate *negation as failure*. Assume that a knowledge base KB consists of a set of rules of the form $(q \leftarrow a_1,\ \ldots,\ a_n)$ where $n \geq 0$, q is an atom, and each a_i is either of the form p or $\mathsf{NOT}(p)$, where p is an atom.

   The backward-chaining derivation procedure presented in lectures might be extended to handle negation as failure as follows:

```
SOLVE[a1, ..., an]:
if n = 0 then return YES
else if a1 = NOT(q) then
    if SOLVE[q] then return NO
        else return SOLVE[a2, ..., an]
else if there is a rule (a1 ← b1, ..., bm) in the KB
    such that SOLVE[b1, ..., bm, a2, ..., an]
then return YES else return NO
```

   [*Note: this procedure my loop indefinitely like the one given in lectures.*]

   (a) Analogously extend the forward-chaining procedure presented in class to handle negation as failure. Atoms should be marked incrementally with either YES (if they are known to be solved), or with NO (if they are known to be insoluble).

   (b) Implement your algorithm from 2a. You should provide documentation in the form of program comments and any additional documentation necessary to understand how your program works and the form of input that it accepts.

   (c) Show how your procedure would label the atoms in the following example KB:

```
a ← b
b ← NOT(h)
c ← d, e
e ←
d ← NOT(b), f
f ← NOT(g), NOT(h), NOT(j)
f ← j
g ← NOT(j)
h ← e
i ← i
```

   (d) Show semantically that KB$\not\models$ g when NOT is understood as ordinary (classical) negation.

3. **[35 Marks]** (Negation as Failure)

(a) There have been a number of attempts at giving negation as failure a clear and intuitive semantics. One way is to use the *closed world assumption*: that is, KB $\models p$ using negation as failure if and only if $\mathrm{KB} \cup \mathrm{KB}_{\mathrm{CWA}} \models p$, where $\mathrm{KB}_{\mathrm{CWA}} = \{\neg q \mid q \text{ is a ground atom such that KB} \not\models q\}$ and with NOT understood as ordinary negation. Another method is to use what is known as the *Clark completion* of a KB:

Suppose KB is a list of rules $(q \leftarrow a_1, \ldots, a_n)$ as in Question 2. The Clark completion of the KB, *comp*(KB), is defined as

$$\mathrm{KB} \cup \{(\neg q \vee \alpha_1 \vee \ldots \vee \alpha_n) \mid (q \leftarrow \alpha_i) \text{ is a rule in the KB}\}.$$

In this case, we would say that KB $\models p$ using negation as failure if and only if $comp(\mathrm{KB}) \models p$, with NOT again understood as ordinary negation.

i. Show that both accounts of negation as failure agree on the ground atoms entailed by the following KB:

```
a ← b
a ← c, d
c.
```

ii. Consider two KBs, each containing a single rule:

(1): `a ← NOT(a).`
(2): `a ← NOT(b).`

Show that in one case, the closed world assumption leads to inconsistency but the Clarke completion does not, and in the other case the reverse is true.

iii. Show that neither way of formalising negation as failure works properly for the KB of Question 2.

(b) i. Suppose that KB is a set of consistent Horn clauses (potentially both positive and negative). Let KB′ be the clauses in KB less one of the negative clauses in KB (if one exists). For any ground atom p, show that KB $\models p$ if and only if KB′ $\models p$.

ii. Suppose that KB is a set of consistent Horn clauses as in part 3(b)i. Show that the set $\mathrm{KB}_{\mathrm{CWA}}$, as defined above, is not affected by deleting a negative clause from the KB.

iii. Again, suppose that KB is a set of consistent Horn clauses. Show that the set $\mathrm{KB} \cup \mathrm{KB}_{\mathrm{CWA}}$ is inconsistent if and only if there are ground atoms $p_1, \ldots, p_n$ such that KB $\models p_1 \vee \ldots \vee p_n$ but KB $\not\models p_i$ for $1 \leq i \leq n$.

4. **[30 Marks]** (Production Systems)

A simple algorithm for multiplying two unsigned binary numbers is to use the "shift and add" method:

Initially you have the two operands and an accumulator (that acts like a temporary variable) which is set to zero. Starting with the leftmost bit of the second operand, if the current bit is 1, add the first operand to the accumulator otherwise (it is 0) do nothing. If there are bits remaining in the second operand, move to the right and shift the accumulator. When there are no bits remaining to be processed in the second operand, the answer will be contained in the accumulator.

(a) In this question you are to implement this algorithm using a production system. As part of this you will also need to implement procedures for performing binary addition (this cannot be assumed as given) and for binary shift. Assume that the working memory (WM) initially contains the two operands, bit by bit, of the multiplication problem using instances of the following elements:

(OP1 pos i bit b)
(OP2 pos i bit b)

where b is a binary digit (0 or 1) and i is an index indicating the position of the binary digit.

For example, if the first operand were 100, the WM would contain:

(OP1 pos 0 bit 0)
(OP1 pos 1 bit 0)
(OP1 pos 2 bit 1)

Furthermore, the WM contains the goal (START). You are to write a collection of production rules that remove (START) and eventually stop with elements in the WM of the form (RESULT pos i bit b) indicating the result of multiplying the two operands bit by bit. Be sure to specify which conflict resolution strategy you are using; you may use any strategy described in lectures. You may not use any arithmetic operators in rules beyond those for manipulating indices.

5. [**30 Marks**] (Frames)
The International Olympic Committee has gotten hold of a paper describing Frames and wishes to investigate this new technology as a possible way of assisting the smooth running of the next Olympic games. They have assigned you the task of implementing a prototype system based on Frames as described in lectures (frames, slots, attached procedures, etc.). In particular, you are to select an event of your own choice and write a program to deal with that event including facilities for handling the preliminary rounds/heats and finals. Slots of frames might be used to record athletes in a heat/final, the location and time of that heat/final, etc. and IF-ADDED/IF-NEEDED and other procedures might be used to encode constraints as well as fill in implied values when the knowledge base is updated.

We particularly wish to consider several ways of updating the knowledge base: (1) asserting that a heat will take place with certain athletes. The system should add this and determine what time and the location of the venue the athletes need to be at for their heat, etc; (2) asserting that a particular semi-final/final should take place, the system should determine the participating athletes; and, (3) asserting that the medal ceremony should take place at a particular time and location, the system should add this and provide the medallists plus appropriate national anthem when queried.

[To simplify matters you can assume that an athlete takes part in only the event you choose. It is sufficient to write procedures using pseudocode.]

(a) Design a set of frames and slots to represent the schedule and any ancillary information needed by the scheduling assistant.

(b) For all slots of all frames, write in English the IF-ADDED or IF-NEEDED procedures that would appear there. Annotate these procedures with comments explaining why they are there (e.g., what constraints they are enforcing)

(c) Briefly explain how your system would work (what procedures would fire and what they would do) on concrete examples of your choosing illustrating each of the three situations (1, 2, and 3) mentioned above.

6. [**40 Marks**] (Description Logic)
Consider the formal semantics of a description logic in the style given in lectures. This involves specifying the extensions of concepts or roles in terms of the extensions of their arguments, as in:

$$\Phi[[(\mathsf{AND}\ c_1\ \ldots\ c_n)]] = \Phi[[c_1]] \cap\ \ldots\ \cap\ \Phi[[c_n]]$$
$$\Phi[[(\mathsf{ALL}\ r\ c)]] = \{x \in \mathcal{D} \mid \text{ if } (x,\ y) \in \Phi[[r]], \text{ then } y \in \Phi[[c]]\}$$

(a) Give a formal semantics in this style for the following concept-forming operators:

- (AT-MOST n r) Maximum role cardinality: Something with at most n r's.
- (SOME r) Role existence: Something with at least one r.

(b) In lectures we only considered concept-forming operators. Give a formal semantics in this style for the following role-forming operators:

- (INVERSE r) Role inverse.
  For example, the CHILD role could be defined as (INVERSE parent).

- (COMPOSE r_1 … r_n) Role composition: the r_n's of the r_{n-1}'s …of the r_1's.
  For example, (ALL (COMPOSE parent brother-in-law) RICH) would mean something all of whose uncles are rich (where an uncle is a brother-in-law of a parent).

(c) Use this semantic specification to show that for any roles r, s, and t, the concept

(ALL (COMPOSE r s) (SOME t))

subsumes the concept

(ALL r (AND (ALL s (EXISTS 2 t)) (ALL s (AT-MOST 2 t))))

where (EXISTS n r) is defined as in the text (p. 129).

(d) Use the semantic specification again to show that the concept

(ALL (COMPOSE r s) (SOME t))

does not subsume

(ALL r (SOME (COMPOSE s t)))

7. [**30 Marks**] (Inheritance Networks)
Consider the following assertions:

Bob lives in the Arctic Circle
Bob comes from Antarctica
People who live in the Arctic Circle typically like cold weather
People who come from Antarctica normally like the cold
People from Antarctica like to play cricket
People who like the cold normally don't like to play cricket
People who like cold weather usually like to play baseball
People who like to play cricket normally don't like to play baseball and vice versa
People who like to play cricket or who like to play baseball like to play sport

(a) Represent the assertions in an inheritance network.

(b) What are the credulous extensions of the network?

(c) Are any of the credulous extensions determined in 7b not preferred extensions? State why.

(d) Give an example of a conclusion that comes out differently depending on whether we use credulous or sceptical reasoning.

(e) Give an example of a conclusion that comes out differently depending on whether we use sceptical or ideally sceptical reasoning. Also give an example of a conclusion that comes out the same.