Assignment 3

CSU520, Spring 2008
Due: Wednesday, Feb. 20

Part I. Propositional Logic Inference Using Exhaustive Model Checking

(Taken from Problem 7.9, p. 238, in the textbook.) Consider the following knowledge base:

- If the unicorn is mythical, then it is immortal.
- If the unicorn is not mythical, then it is a mortal mammal.
- If the unicorn is either immortal or a mammal, then it is horned.
- If the unicorn is horned, then it is magical.

1. Create propositional symbols to represent each of the relevant assertions, (e.g., "the unicorn is mythical"), and then represent the knowledge given above as a single propositional logic sentence. You may use either Lisp-style syntax or the mathematical logic syntax used in the book.

2. Use the function *exhaustive-check-satisfiability* in the file "satisfiability.lisp" (from the `/programs/logic/proposit` subdirectory under the course web page) to determine whether each of the following assertions are entailed by the above knowledge base:

- The unicorn is mythical.
- The unicorn is magical.
- The unicorn is horned.

(To run this program you will, of course, need to express your answer to problem 1 using Lisp-style syntax.) Print out and turn in the dribble output. (Also, be sure to state explicitly for each of the 3 assertions whether it is or is not entailed by the given knowledge base, which you will determine by interpreting this output appropriately.)

3. Convert the knowledge base from problem 1 into CNF. (Hint: The result should contain 6 clauses.)

Part II. Finding Satisfying Assignments Using WalkSAT

4. Consider the following simple logic puzzle: Jones, Smith, and Clark hold the jobs of programmer, knowledge engineer, and manager (not necessarily in that order). Jones owes the programmer $10. The manager's spouse prohibits borrowing money. Smith is not married. Your task is to figure out which person has which job.

Use nine propositional symbols to represent the possible person/job assignments, and represent the given facts in propositional logic, together with all the constraints that you know must hold since there is a 1-1 correspondence between names and jobs. For example, you might use the symbol SM to represent the assertion "Smith is the manager." You do not need to represent the relation between owing and borrowing, or being married and having a spouse; you can just use these to draw conclusions. (E.g., from "Smith is not married" and the mention of "the manager's spouse" we know that Smith can't be the manager, which we

would represent in Lisp-style syntax as (not SM).) Express this knowledge base as a collection of clauses (i.e., in CNF form with an implicit AND joining the clauses).

Hint: Your CNF knowledge base should contain 24 clauses, most of which are required to represent the 1-1 correspondence constraint.

5. Use the WalkSAT program in the `/programs/logic/propositional` subdirectory, with $p = 0.5$, to find a satisfying assignment for these clauses. Is this the only satisfying assignment? (You may run the exhaustive search program to check this since the truth table is not impractically large in this case.) Hand in corresponding dribble output, and summarize clearly the solution found. Comment on how hard or difficult it is for WalkSAT to find a solution to this puzzle (based on the number of iterations it took), compared to how much effort must be expended by the exhaustive search program.

Part III. Using First-Order (Predicate) Logic

6. Translate each of the following English sentences into predicate calculus sentences in explicit quantifier form The only predicates you should use are these five:

```
car        new        person        owns        drives
```

Your answers may use either Lisp-style or mathematical syntax, whichever you prefer. The meaning of these predicates is as follows:

```
car(x)        or  (car x)         means    ‘‘x is a car’’
new(x)        or  (new x)         means    ‘‘x is new’’
person(x)     or  (person x)      means    ‘‘x is a person’’
owns(x,y)     or  (owns x y)      means    ‘‘x owns y’’
drives(x,y)   or  (drives x y)    means    ‘‘x drives y’’
```

The only constants you should use are `mary` and `john`.

To get you started, the Lisp-style translation of “Mary drives a new car” should begin `(exists (x) ...`

a. “Mary drives a new car.”

b. “Mary drives a car but John does not drive a car.”

c. “Mary owns a car but does not drive it.”

d. “Everybody who owns a new car drives it.”

e. “Not everybody drives a car”

7. Skolemize the following predicate calculus sentences:

a. `(forall (x y) (if (larger x y) (smaller y x)))`

b. `(exists (y) (forall (x) (larger x y)))`

c. `(forall (x) (exists (y) (larger x y)))`

d. `(not (forall (x) (exists (y) (larger x y))))`

e. `(forall (x) (if (exists (y) (larger x y)) (not (small x))))`

8. For each of the following pairs of predicate calculus sentences, give a MGU (most general unifier) for them if they unify. If they do not unify, indicate this instead.

```
a1:   (older ?x ?y)
a2:   (older (father-of john) sally)

b1:   (older (father-of ?z) ?z)
b2:   (older (father-of john) sally)

c1:   (older ?x ?y)
c2:   (older (father-of john) john)

d1:   (older (father-of ?x) ?y)
d2:   (older (father-of john) john)

e1:   (older ?x ?y)
e2:   (older (father-of ?z) john)
```