CIS 730 Artificial Intelligence
CIS 530 Principles of Artificial Intelligence
Fall 2007

Homework 4 of 10: Machine Problem (MP4)
Constraints and Logic, Part II: Clausal Form and Logic Programming

Assigned: Sat 29 Sep 2007
Due: Sat 13 Oct 2007 (before midnight)

The purpose of this assignment is to exercise your basic understanding of logic programming and rule-based expert systems through a simple implementation.

This homework assignment is worth a total of 20 points.
Each problem is worth 4 points for CIS 730 students and 7 points for CIS 530 students.

Use `gprolog` on `fingolfin.user.cis.ksu.edu` to solve all five problems. Each problem should have a source code file mp4_i.pl. Put any additional notes in your README.txt file.
Upload to your KSOL drop box a .zip file named MP2-XYZ.zip, with your initials in place of XYZ.

1. **(530/730) Prolog Programming.** (From p. 37-41, *Logic, Programming and Prolog* by Ulf Nilsson and Jan Małuszyński.) Consider the axioms:

   Rules

   i. *Every good student taught by a logician knows logic.*
   ii. *Tom is a good student.*
   iii. *Peter is a logician.*
   iv. *Tom is Peter's student.*

   Use Prolog to prove that Tom knows logic.
   Turn in your source (`mp4_1.pl`) and a problem solving trace (`mp4_1-out.txt`).

2. **(730 only) Term Project.** Write at least two Prolog rules containing representative axioms from your term project

   a) Computer Role-Playing Game (*Angband*)
   b) Trading Agent Competition – Supply Chain Management (*TAC-SCM*)
   c) Protégé, DIP and BIND-based evidence ontology for protein interaction

Conjunctive Normal Form (CNF) converter

In this assignment you will write a program to partially parse first order predicate calculus (FOPC), *aka* first-order logic (FOL) sentences, into CNF, i.e., clausal form. This will be completed and applied in Machine Problem 6.

Refer to the following specification for Problems 4-5.

```
The hash mark denotes comments.
# Your program should ignore all input on a line containing the '#'
# symbol.
#
# The first non-comment line contains a single integer
# denoting the number of sentences.
```

```
3
# The second non-comment line starts the sentences.
# Syntax:
#     \A denotes UNIVERSAL QUANTIFICATION (\forall)
#     \E denotes EXISTENTIAL QUANTIFICATION (\exists)
#     => denotes IMPLICATION (\rightarrow)
#     & denotes CONJUNCTION, i.e., AND
#     | denotes DISJUNCTION, i.e., OR (\vee)
#     ! denotes NEGATION, i.e., NOT (\not)
#     = denotes EQUALITY (NOTE: this is OPTIONAL in the regular MP)
#     . separates each quantified variable from its scoped expression
# Conventions:
#     - All variables are lowercase
#     - All constants are alphanumeric names in ALL CAPS
#     - All predicates contain alphanumeric characters or _, begin with
#       a capital letter, and enclose their arguments in parentheses
#     - All functions contain alphanumeric characters or _, begin with
#       a lowercase letter, and enclose their arguments in parentheses
#     - Use C/C++ precedence for & and |
\A x . \A y . P(x, y) => \E z . Q(x, z) & !R(y, z)
Husband_Of (JOE1, SUSAN)
Longer (left_leg_of(RICHARD), left_leg_of(JOHN)) & (Foo() | bar = baz)
#
# Correct answers:
# 1. {!P(x_1, y_1), Q(x_1, sf1(x_1, y_1))},
#    {!P(x_2, y_2), !R(x_2, sf1(x_2, y_2))}
# 2. {Husband_Of (JOE1, SUSAN)}
# 3. {{Longer (left_leg_of(RICHARD), left_leg_of(JOHN))},
#     {Foo(), bar = baz}}
```

3. **(530/730) Scanning and parsing FOL expressions.** Write a Java or C++ program to scan and parse the above expressions into an internal *expression tree* format. You may use any of the following:

   a) Java string tokenizer
   http://java.sun.com/j2se/1.4.2/docs/api/java/util/StringTokenizer.html
   b) C Standard Library (`string.h`)
   http://www.cs.cf.ac.uk/Dave/C/node19.html
   c) C++ Standard Template Library (STL) string library
   http://www.processdoc.com/doc/cppstl/string.html
   d) `lex/flex, ml-lex`
   http://cs.wwc.edu/~aabyan/464/Book/LexFlex.html
   See also: http://dinosaur.compilertools.net
   http://www.smlnj.org/doc/ML-Lex/
   e) `yacc/bison, ml-yacc`
   http://cs.wwc.edu/~aabyan/464/Book/YaccBison.html
   See also: http://dinosaur.compilertools.net
   http://smlnj.cs.uchicago.edu/doc/ML-Yacc/
   f) *Antlr*.
   http://www.antlr.org

   As output, print the expression tree in a *prefix traversal*. For example, P | Q should be printed out as Or (P, Q). Turn in `mp4_3` (prefix any Lex, Yacc, or Antlr spec thusly) and include instructions in your README.txt file for generating your actual scanner/parser source, compiling it, and running it on test input.

4. **(530/730) Implications, deMorgan's Theorem, and standardization of variable names.** Extend your parser code by adding code to perform the "INS" part of the "INSEUDOR" procedure for conversion to clausal form. Print the resulting intermediate

version of expressions given as standard input. Turn in `mp4_4`, a revised version of your scanner/parser code or spec.

5. **(730) Skolemization.** Perform full Skolemization as explained in Russell and Norvig 2^{e}, and in the examples given in class. Turn in `mp4_5`, a revised version of your scanner/parser code or spec.

Extra Credit (4 points for 730, 7 points for 530). Document your Skolemization code and test it on the sample inputs, printing the "INSE" results.

Class Participation (required)

After going over your **Read And Explain Pairs** exercise with your assigned partner, post a short paragraph summarizing *unification* and a second containing any questions you may have on search to the class mailing list (**CIS730-L@listserv.ksu.edu**). That is, ask questions about any knowledge representation topic for which your understanding is unclear. This includes propositional logic, first-order logic, description logic, or theorem proving (forward and backward chaining, resolution strategies)

Midterm Exam Review

A review guide will be posted next weekend with examples not in this homework.

Coming Up Next

Problem Set 5 (due Wed 17 Oct 2007) – Constraints and Logic, Part III: Decidability, Situational Calculus, First-Order Planning

Machine Problem 6 (due Fri 26 Oct 2007) – Constraints and Logic, Part IV: More Clausal Form and Prolog with Applications to Planning

Problem Set 7 (due Fri 02 Nov 2007) – Artificial Neural Networks and Probabilistic Reasoning

Machine Problem 8 (due Fri 09 Nov 2007) – Probabilistic Reasoning

Problem Set 9 (due Fri 16 Nov 2007) – Machine Learning

Machine Problem 10 (due Fri 30 Nov 2007)– Term Project Experiments