Instructor: R. Greiner  
Due Date: Tuesday, 30 October 2007 at start of class

The following exercises are intended to further your understanding of logic (propositional and first order), efficient reasoning, and planning (and a question on $\alpha - \beta$ pruning). Of course, be sure to explain your answers, etc etc etc.

Total points: $85 + 45 = 130$

**Submission:** You should hand-in hardcopies of Problems 1 to 6. For Problem 7: use ASTEP to submit a gzipped tar file named

```
  as2.tgz
```

such that unpacking it creates a directory called `as2` with the following subdirectories that contain your solutions:

<table>
<thead>
<tr>
<th>Subdirectory</th>
<th>Contains your solution to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>as2/p7ad</td>
<td>7(a) to 7(d) [as raw text, or *.pdf]</td>
</tr>
<tr>
<td>as2/p7e</td>
<td>Problem 7(e)</td>
</tr>
<tr>
<td>as2/p7f</td>
<td>Problem 7(f)</td>
</tr>
<tr>
<td>as2/p7g</td>
<td>Problem 7(g)</td>
</tr>
</tbody>
</table>

(If you wish, you may instead hand in hardcopies of 7(a)–7(d). If so, the `as2/p7ad` subdirectory should include a single file stating this. This hardcopy should be distinct from the paper for Problems 1–6, and contain your name / student id, etc.)

Note each coding sub-question is in its own sub-directory. Typing `make` in each subdirectory should compile the program. The name of the executable should be the same as the name of the subdirectory.

---

**Problem 1 [6 points] Unification**

For each of the following pairs of literals, find a most-general unifier (MGU), or explain why the expressions do not unify. (Note that CAPITAL letters correspond to variables.)

<table>
<thead>
<tr>
<th>A</th>
<th>h( X, g(3) )</th>
<th>h( g(3), X )</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>h( X, Y )</td>
<td>f( a, Z )</td>
</tr>
<tr>
<td>C</td>
<td>f( X, q(X) )</td>
<td>f( q(3), Y )</td>
</tr>
<tr>
<td>D</td>
<td>h( g(X), g(7) )</td>
<td>h( Z, Z )</td>
</tr>
<tr>
<td>E</td>
<td>h( g(X), X )</td>
<td>h( Z, g(Z) )</td>
</tr>
<tr>
<td>F</td>
<td>h( X, X )</td>
<td>h( g(q(W)), g(Z) )</td>
</tr>
</tbody>
</table>

**Problem 2 [10 points] Game Playing — Alpha-Beta Pruning**

Consider the game tree shown in Figure 1; the static scores (in parentheses under each leaf node) each reflect the first player, who is maximizing.
Figure 1: Game Tree

a [3]: What move should the first player choose?
(To receive any partial credit, you should show the values of the intermediate nodes.)

b [3]: What nodes would not need to be examined using the alpha-beta algorithm —
assuming the nodes are examined in the left-to-right order?

c [4]: Is this “left-to-right” ordering optimal for alpha-beta? If so, give a simple argument
explaining why. If not, give an ordering that would require examining fewer nodes.
(While you get to change the order, of course you may not alter the connectivity — e.g., C’s children
must remain H and I — nor can you modify the static scores — e.g., the value of T must remain
5.)

To simplify the notation, use “g(E,F)” for “E ≥ F”.

Problem 4 [19 points] In class, we defined

HappyMother = Woman whose children are all rich, and all married to pediatricians.
SuccessfulParent = Person whose sons are all married to doctors.

a [5]: Write this out, using the vocabulary:

child(x, c) : c is a child of x
hm(x) : x is a happy mother
married(x, y) : x is married to y
md(x) : x is a doctor
person(x) : x is a person

ped(x) : x is a pediatrician
rich(x) : x is rich
son(x, c) : c is a son of x
sp(x) : x is a successful parent
woman(x) : x is a woman

Also include propositions defining the obvious relationships: every woman is a person, every
son is a child, and every pediatrician is a doctor.

b [10]: Is every HappyMother a SuccessfulParent? You should use (refutation)
resolution to answer this.

c [4]: Suppose consider some slight variants:
\[ \text{HMa} = \text{Woman whose children are all rich and famous, and all married to pediatricians.} \]

\[ \text{SPb} = \text{Person whose sons are all married to lawyers.} \]

(Note that pediatrician is NOT a subset of lawyer.)

Is every HMa a SP?

Is every HM a SPb?

You need not give a complete derivation; instead you should provide a succinct argument for your answer, in terms of the derivation for part (b) above.

**Problem 5 [10 points] Sussman Anomaly**

[Russell/Norvig: Exercise 11.11, p414]

**Problem 6 [30 points] Control of Reasoning**

Consider \( \text{Ask}(KB_0, \text{lender(me, u_1)}) \), where

\[
KB_0 = \begin{cases}
R1: \text{relative}(X, Y) :& - \text{cousin}(X, Y). \\
R2: \text{relative}(X, Y) :& - \text{uncle}(X, Y). \\
R3: \text{relative}(X, Y) :& - \text{brother}(X, Y). \\
\vdots \\
R4: \text{lender}(X, Y) :& - \text{relative}(X, Y), \text{rich}(Y). \\
R5: \text{lender}(X, Y) :& - \text{gullible}(Y), \text{rich}(Y). \\
R6: \text{rich}(Y) :& - \text{own}(Y, H), \text{house}(H). \\
R7: \text{rich}(Y) :& - \text{ceo}(Y, C), \text{company}(C). \\
\vdots \\
\text{brother}(me, u_1). \quad \text{company}(c_7). \quad \text{house}(h_2). \quad \text{ceo}(u_1, c_7).
\end{cases}
\]

a [5]: Assuming the rules are considered in the order shown here, (and the “…”s contain only irrelevant information), describe how Prolog would answer this query — ie, specify the rules that would be considered, when the system would backtrack, etc.

Also give the answer that Prolog will return.

b [5]: One way to improve the efficiency of the reasoner, at least for this query, is by re-ordering the rules. Give an ordering that would be more efficient — that is, which would return the same answer(s), but in fewer steps.

c [5]: Now consider “chaining together” the rules R4, R3 and R7, to form

\[ R_{\text{new}}: \text{lender}(X, Y) : - \text{brother}(X, Y), \text{ceo}(Y, C), \text{company}(C). \]

Let \( KB_1 = KB_0 \cup \{ R_{\text{new}} \} \) be the knowledge base formed by adding this new rule. Does this addition change the “contents” (read “deductive closure”) of the knowledge — ie, will \( \text{Ask}(KB_0, \sigma) \) always equal \( \text{Ask}(KB_1, \sigma) \), for all \( \sigma \)?

d [5]: If our goal is to speed up the reasoner, for this \( \text{lender(me, u_1)} \) query, where should we add this \( R_{\text{new}} \) rule? In particular, is it better to add this new rule to the FRONT of the knowledge base (so it will be used first), or to the end?

e [5]: Give a query \( \sigma \) that is entailed, but will take longer to answer using \( KB_1 \) than using \( KB_0 \) — ie, \( \text{Ask}(KB_0, \sigma) = Yes = \text{Ask}(KB_1, \sigma) \), but \( \text{Ask}(KB_0, \sigma) \) requires fewer inference
steps than \texttt{Ask}(KB_1, \sigma).

Here, \( KB_1 \) is the version that added the rule as specified by (d).

You may assume that \( KB_0 \) includes other facts, which allow it to entail that others are lenders, or perhaps other types of propositions.

Compare the efficiency of answering a query \( \sigma \) that is \textbf{not entailed}, from \( KB_1 \) versus from \( KB_0 \).

\textbf{f [5]:} Now consider the query \texttt{lender(me, L)} — i.e., find all lenders. (For Prolog-experts, think
\[
| ?- \text{setof( L, lender(me, L), S).}
\]
Can either rule-ordering, or rule-chaining, help here? That is, do either improve the time required to answer this query?

\textbf{Problem 7 [45 points] CSP/SAT — Problem Formulation}

\textit{Sudoku} has become a very prominent game today, with millions of people solving puzzles that appear in hundreds of papers and websites; see \url{http://en.wikipedia.org/wiki/Sudoku}. In general, each Sudoku instance begins with a 9 \times 9 board, each of whose 81 squares are either blank, or contain a digit \( \{1, 2, \ldots, 9\} \). The goal is to fill in the blank squares with the digits \( \{1, 2, \ldots, 9\} \), with the constraints that each row must contain exactly one of each digit, as must every column, as must each of the 9 “regions”, each a 3 \times 3 square centered whose upper left position is \( (x, y) \) for each \( x, y \in \{1, 4, 7\} \).

This problem considers a simpler variant, dealing with a smaller 6 \times 6 board; \textit{e.g.,}

\begin{center}
\begin{tabular}{|c|c|c|}
\hline
6 & 4 & 1 \\
\hline
3 & & \\
\hline
\end{tabular}
\begin{tabular}{|c|c|c|}
\hline
4 & 3 & 5 \\
\hline
3 & 2 & 6 \\
\hline
\end{tabular}
\begin{tabular}{|c|c|}
\hline
5 & 1 \\
\hline
\end{tabular}
\end{center}

(\url{http://www.dailysudoku.com/sudoku/kids}) where the goal now is to fill in the blank squares with the digits \( \{1, 2, \ldots, 6\} \), such that each row, column and 2 \times 3 “region” (whose upper left position is \( (x, y) \) for some \( x \in \{1, 4\} \) and \( y \in \{1, 3, 5\} \)) contains exactly one instance of each digit.

We consider this challenge of solving such a game — \textit{i.e.}, finding a satisfying assignment. One obvious approach involves \( 6 \times 6 = 36 \) variables, with one \( B_{x,y} \) for each \( (x, y) \in \{1..6\}^2 \).

This question, however, will use a different representation, based on \( 6 \times 6 \times 6 = 216 \) \textit{binary} variables \( \{\alpha_{x,y,v}\} \), where each \( \alpha_{x,y,v} \) corresponds to assigning the value \( v \in \{1..6\} \) to position \( (x, y) \in \{1..6\}^2 \).

\textbf{a [10]:} Using these variables, describe the constraints (“clauses”) for a \textit{completely blank} board. Be sure to argue that you have included all of the constraints — that is, make sure that every assignment consistent with the constraints is a legal board.
Do not include redundant clauses. That is, every constraint in this set should be \textit{required} to specify a legal board.

How many clauses, of what arities, do you need to include? (Be careful not to double count.)

In your description, feel free to use elipses, or “meta-variables”, after explaining the meaning.

\textbf{b} [5]: Note that each specific assignment corresponds to asserting an additional constraint; \textit{e.g.}, stating that position \((3, 4)\) has value 1 corresponds to asserting the unit clause \(\alpha_{3,4,1}\).

Show the immediate (1-step) entailments from this \(\alpha_{3,4,1}\) assertion. Also discuss which clauses are subsumed...and what happens to such clauses. (Hint: \(\alpha_{3,4,1} \Rightarrow \varphi \lor \alpha_{3,4,1}\) for any formula \(\varphi\).)

\textbf{c} [5]: Using these clauses, describe what happens when row\#1 of the initial board contains

\[
\begin{array}{ccc}
1 & 2 & \square \\
4 & 5 & 6 \\
\end{array}
\]

that is after asserting the 5 unit clauses \(\alpha_{1,1,1}, \alpha_{2,1,2}, \alpha_{4,1,4}, \alpha_{5,1,5},\) and \(\alpha_{6,1,6}\). In particular, show how this entails \(\alpha_{3,1,3}\).

Be sure to focus on the essential parts of the derivation. Again, feel free to use elipses, or “meta-variables”, after making sure the meaning is clear.

\textbf{d} [5]: Consider the initial board:

\[
\begin{array}{ccc}
\hline
1 & 5 & \ast & 6 \\
\hline
\hline
4 & & \\
\hline
5 & 6 & 3 \\
\hline
1 & 4 & \\
\hline
\hline
\end{array}
\]

What is the value of position \((4, 1)\) \textit{i.e.}, the “\ast” between the cells labeled 5 and 6? Show a short proof of how this was derived, in terms the the \(\alpha_{x,y,v}\) values.

\textbf{e} [10]: Implement a “Sudoku \(\mapsto\) boolean-CSP” translator, whose input is the name of a file, which should contain 36 (white-space-delimited) integers corresponding to a \(6 \times 6\) matrix. Each \((i, j)\) entry is in \(\{0, 1, \ldots, 6\}\), where the value 0 corresponds to a blank. (See for example \texttt{http://www.cs.ualberta.ca/~greiner/C-366/6x6Sudoku/simple6x6.data}.) This program should compute a representation of the board in CNF format \textit{i.e.}, a list of clauses, where each clause is a list of literals, where each literal is a variable \(\alpha_{x,y,v}\) or its negation.

You should remove any clause that is subsumed (based on the input array) but (for now) not do any other reductions. Your program should then print out the CNF, in the form used as input to the \texttt{gsat} problem in HW\#1.
c this is a comment!
c The following formula is (easily) satisfiable
p cnf 5 7
-1 2 -3 0
1 -2 3 0
2 3 5 0
-1 2 -5 0
-3 -4 -5 0
-1 -3 4 0
-1 3 -4 0

Of course, your formulae will probably be very different from this.

You should encode each $a_{x,y,v}$ variable with the number $v + 6 \times [(y - 1) + 6(x - 1)] = v + 6y + 36x - 42$. (Note these values are always positive. Recall that 0 is used to mean the end of a list.)

If you wish, you may use the C code in http://www.cs.ualberta.ca/~greiner/C-366/6x6Sudoku/6x6-sudoku.c to help with the input and some parts of the output.

f [5]: Rather than just print out the formula, instead pass it to a SAT solver (perhaps your own from HW#1, or another), that will then produce a satisfying assignment. You should also write the code that prints out the board corresponding to this assignment.

You are allowed to use code from other sources for your SAT solver, provided you explicitly acknowledge the source of this code.

Be sure to try it out on various problems; check the course website for some relevant $6 \times 6$ Sudoku instances.

g [5]: Write your “Best Sudoku solver”, called p7g, using whatever techniques you want — either based on your boolean version, or perhaps based on the $B_{x,y}$ encoding mentioned above. Feel free to look over http://www.norvig.com/sudoku.html for ideas in general.

You must also describe your algorithm: e.g., does it do forward checking? Or include MRV? Or...

We will then compare your algorithm with the ones written by others in the class, and announce the winner(s). As with HW#1...while bragging is fun, please note this problem is worth only 5 points, out of 130 for this assignment, and allocate your time accordingly.

Good luck!