CS 4804 Homework 4
Solution Sketches

---

1. **(10 points)** The desired expression can be given by:

$$\begin{aligned}\forall x\ \text{Politician(x)} \Rightarrow\ & \\ & ((\exists y\ \forall t\ \text{Person(y)} \wedge \text{Fools(x,y,t)}) \wedge \\ & (\exists t\ \forall y\ \text{Person(y)} \Rightarrow \text{Fools(x,y,t)}) \wedge \\ & \neg (\forall t\ \forall y\ \text{Person(y)} \Rightarrow \text{Fools(x,y,t)}))\end{aligned}$$

2. **(10 points)** Thinking of PROLOG as an inference engine, the cut (!) is a way for the programmer to alter the default search strategy built into the inference engine. For instance, we can use the cut as a mechanism to prevent consideration of alternate solutions. Remember that PROLOG uses depth first search with backtracking, and so the cut could be encountered either when PROLOG is navigating down the search tree or while backtracking.

   It is easiest to describe the operation of the cut when it is encountered during backtracking. When this happens, the siblings of the current node are pruned out and the search continues with the 'grandparent' node. For instance, consider the following sequence of rules:

```
getanAin4804(X) :- didwellin4804(X), !.
getanAin4804(X) :- niceperson(X).
```

   PROLOG initially sets up two branches under `getanAin4804`, corresponding to the two ways of satisfying this predicate. When the cut is encountered, it means that `didwellin4804` was satisfied, so there is no need to consider the second rule (involving `niceperson`); this effect is achieved by placing a cut strategically in the first rule (at the point where we are sure we want to prune the tree). There is no grandparent node in this case, so the inference algorithm terminates with the one solution. The above sequence of rules mimics an if-then-else operation, namely:

```
if (didwellin4804) then getanAin4804
else if (niceperson) then getanAin4804;
```

3. **(20 points)** The first situation happens because PROLOG omits the occurs-check in its unification algorithm. So, it doesn't notice that the variable `X` itself occurs in the term it is being instantiated to. With the given information, it is clear that the goal predicate is true only if there is a number that is its own successor. However, while PROLOG thinks that such a number exists, attempting to print this number out reveals that PROLOG is thinking in an infinite loop.

   The second situation happens due to PROLOG's implementation of the `not` operator. PROLOG uses *negation by failure*, which means that `not (X)` is true whenever goal `X` fails. So the goal `not (not (human (X)))` succeeds because `not (human (X))` fails with `X` unified to `bob`. However, then PROLOG backtracks and this unification is *released*, causing `X` to be uninstantiated. PROLOG prints out a cryptic value denoting this aspect.

4. **(30 points = 20 points + 10 points)** Let us assume the following vocabulary:

   - esbcmember(x): true when person x is a member of the Elm St. Bridge Club.
   - marriedto(x, y): true when person x is married to person y.
   - brother(x, y): true when person y is brother of person x.
   - esbcmeetingat(x): true when the last meeting of the Club was held at place x.
   - house(x, y): true when x's house is place y.
   - equal(x, y): true when x is equal to y.

   Then the various statements can be asserted as:

   - The members of the Elm St. Bridge Club are Joe, Sally, Bill, and Ellen.
     esbcmember(Joe).
     esbcmember(Sally).
     esbcmember(Bill).
     esbcmember(Ellen).
     $\forall x$ esbcmember(x) $\Rightarrow$ (equal(x, Joe) $\vee$ equal(x, Sally) $\vee$ equal(x, Bill) $\vee$ equal(x, Ellen))
   - Joe is married to Sally.
     marriedto(Joe, Sally).
   - Bill is Ellen's brother.
     brother(Ellen, Bill).
   - The spouse of every married person in the club is also in the club.
     $\forall x, y$ esbcmember(x) $\wedge$ marriedto(x, y) $\Rightarrow$ esbcmember(y).
   - The last meeting of the club was at Joe's house.
     $\forall x$ esbcmeetingat(x) $\Rightarrow$ house(Joe, x).

   In addition, we will need to assert some obvious axioms:

   - Spouses live in the same house.
     $\forall x, y, z$ marriedto(x, y) $\wedge$ house(x, z) $\Rightarrow$ house(y, z).
   - People do not get married to their brothers.
     $\forall x, y$ marriedto(x, y) $\Rightarrow \neg$ brother(x, y).
   - People do not marry themselves.
     $\forall x$ $\neg$ marriedto(x, x).
   - Marriage is a symmetric relationship.
     $\forall x, y$ marriedto(x, y) $\Rightarrow$ marriedto(y, x).
   - Every person is married to only one other person.
     $\forall x, y, z$ marriedto(x, y) $\wedge$ marriedto(z, y) $\Rightarrow$ equal(x, z).
   - We can substitute equal people in a marriage.
     $\forall x, y, z$ marriedto(x, y) $\wedge$ equal(y, z) $\Rightarrow$ marriedto(x, z).

   Converting these statements to clauses gives a whopping 15 sentences! We also take care to use new variables for every universally quantified sentence, to prevent confusion.

   (a) esbcmember(Joe)

   (b) esbcmember(Sally)

(c) $\text{esbcmember}(\text{Bill})$

(d) $\text{esbcmember}(\text{Ellen})$

(e) $\neg\,\text{esbcmember}(r) \vee \text{equal}(r, \text{Joe}) \vee \text{equal}(r, \text{Sally}) \vee \text{equal}(r, \text{Bill}) \vee \text{equal}(r, \text{Ellen})$

(f) $\text{marriedto}(\text{Joe}, \text{Sally})$

(g) $\text{brother}(\text{Ellen}, \text{Bill})$

(h) $\neg\,\text{esbcmember}(x) \vee \neg\,\text{marriedto}(x, y) \vee \text{esbcmember}(y)$

(i) $\neg\,\text{esbcmeetingat}(z) \vee \text{house}(\text{Joe}, z)$

(j) $\neg\,\text{marriedto}(l, m) \vee \neg\,\text{house}(l, n) \vee \text{house}(m, n)$

(k) $\neg\,\text{marriedto}(s, t) \vee \neg\,\text{brother}(s, t)$

(l) $\neg\,\text{marriedto}(u, u)$

(m) $\neg\,\text{marriedto}(v, w) \vee \text{marriedto}(w, v)$

(n) $\neg\,\text{marriedto}(a, b) \vee \neg\,\text{marriedto}(c, b) \vee \text{equal}(a, c)$

(o) $\neg\,\text{marriedto}(d, e) \vee \neg\,\text{equal}(e, f) \vee \text{marriedto}(d, f)$

We are now ready to do our proofs. The first statement we need to prove is

$$\forall\, x\ \ \text{esbcmeetingat}(x) \Rightarrow \text{house}(\text{Sally}, x).$$

Negating this statement and adding it to our database gives two clauses:

$$\text{esbcmeetingat}(H1), \neg\,\text{house}(\text{Sally}, H1)$$

where $H1$ is a Skolem constant. Lets call the first clause as $G1$ and the second clause as $G2$. Then the resolution refutation proceeds as shown below.

```
G2
|-- (j)
|-- (f)
|-- (i)
|-- (G1)
NIL
```

To prove the second statement, we add the following clause to our database:

$$\text{marriedto}(\text{Ellen}, P1)$$

and lets call it $G3$. Then the resolution refutation proceeds as follows:

```
G3
|-- (h)
|-- (d)
|-- (e)
|-- (o)
|-- (G3)
|-- (l)
|-- (o)
```

```
|-- (G3)
|-- (n)
|-- (m)
|-- (f)
|-- (o)
|-- (G3)
|-- (n)
|-- (f)
|-- (o)
|-- (G3)
|-- (k)
|-- (g)
NIL
```

Notice that the proof has three sequences of resolving with (o) followed by $(G3)$. This is because it is slowly eliminating each member one by one. For instance, Ellen cannot be married to Joe because Joe is already married to Sally. Next, Ellen cannot be married to Sally because Sally is already married to Joe. Finally, Ellen cannot be married to Bill because he is her brother. That leaves nobody else and gives us the empty clause. Along the way you will notice that we have to assert the falsehood of equal(Ellen, Sally) and also equal(Ellen, Joe).

The reason why this proof is so long is because we have had to reason about equality, in addition to concepts like marriage and brotherhood. The purpose of this exercise is to show that after some time, this can get very tedious. Your textbook proposes a solution to this, namely adding new inference rules such as *demodulation* and *paramodulation*. See the section on 'Dealing with equality' in AIMA for more information.

As both proofs show, input resolution is sufficient and hence so is linear resolution. Unit resolution is not sufficient to complete these proofs. Set of support resolution is complete only if the set of support is chosen properly.

5. **(30 points = 4 points each for parts (a)-(e); 5 points each for parts (f) and (g))** (a) The first sentence means that for every natural number, there is some other natural number that is lesser than or equal to it. The second sentence means that there is a specific natural number that is lesser than or equal to any natural number. (b&c) Both sentences are obviously true given our understanding of natural numbers. (d) No. (e) Yes. (f) To prove that A follows from B, we add B and ¬A to our initial knowledge base and try to prove that the latter is inconsistent. But before that we need to convert each into clausal form (in the following, *gte* denotes 'greater than or equal to.'):

$$\begin{aligned}
B &\Leftrightarrow \exists y \, \forall x \; gte(x, y) \\
&\Leftrightarrow \forall x \; gte(x, F2) \\
&\Leftrightarrow gte(x, F2) \\
\neg A &\Leftrightarrow \neg (\forall x \, \exists y \; gte(x, y)) \\
&\Leftrightarrow \exists x \, \forall y \, \neg gte(x, y) \\
&\Leftrightarrow \forall y \, \neg gte(F1, y) \\
&\Leftrightarrow \neg gte(F1, y)
\end{aligned}$$

where $F1$ and $F2$ are Skolem constants. The resolution goes through, with the substitution $\{x/F1, y/F2\}$. (g) In this case, however, we will be unable to show this by resolution-refutation. Consider:

$$\begin{aligned}
\neg B &\Leftrightarrow \neg\,(\exists\, y\ \forall\, x\ gte(x,y)) \\
&\Leftrightarrow \forall\, y\ \exists\, x\ \neg\, gte(x,y) \\
&\Leftrightarrow \forall\, y\ \neg\, gte(F2(y),y) \\
&\Leftrightarrow \neg\, gte(F2(y),y) \\
A &\Leftrightarrow \forall\, x\ \exists\, y\ gte(x,y) \\
&\Leftrightarrow \forall\, x\ gte(x,F1(x)) \\
&\Leftrightarrow gte(x,F1(x))
\end{aligned}$$

Here, $F1$ and $F2$ are Skolem functions. If you think the substitution $\{x/F2(y), y/F1(x)\}$ should work, notice that it is actually equivalent to $\{x/F2(y), y/F1(F2(y))\}$, which means that y is being bound to an expression containing y. The occurs-check would rule out the unification, meaning that that this is not provable.