Hints for Solving Logic Problems

These guidelines will help in the translation of ``story problems'' into predicate calculus and solution of the problems by resolution. Following these will help to avoid common errors.

1. *Terms* (constants, variables, and functions) are always objects in the domain about which the formulas are written. *Predicates* are true/false properties or relations of these objects.

2. It is a good idea to write down, at the side, type descriptions of predicates, e.g.,
   DRIVES(x,y) where *x* is a person and *y* is a car.
   A use of the predicate that involves objects of the wrong types is likely either to be wrong or not part of a solution.

3. A statement ``for all'' or ``every'' in English usually translates to a $\forall$ quantifier and uses a $\rightarrow$ connective. ``There exists'' or ``some'' usually translates to $\exists$ and uses a $\wedge$ connective.

4. Use your knowledge of the real-world meaning of a statement to guide the translation into predicate calculus. There may be multiple correct ways of translating a statement, but they all result in the same clause form. For example, the statement ``No cat likes any dog'' could be written:
   1. $\forall x\ (CAT(x) \rightarrow \forall y\ (DOG(y) \rightarrow \neg\ LIKES(x,y)))$
   2. $\forall x\ (CAT(x) \rightarrow \neg\ \exists y\ (DOG(y) \wedge LIKES(x,y)))$
   3. $\neg\ \exists x\ (CAT(x) \wedge \exists y\ (DOG(y) \wedge LIKES(x,y)))$
   4. $\neg\ \exists x\ \exists y\ (CAT(x) \wedge DOG(y) \wedge LIKES(x,y))$

   All of these produce the clause form: $\neg\ CAT(x) \vee \neg\ DOG(y) \vee \neg\ LIKES(x,y)$

5. After translating a sentence into clause form, read it back into English and see whether it has the same meaning: ``Either *x* is not a cat or *y* is not a dog or *x* does not like *y* .''

6. Any form *P(x)* or $\neg$ *P(x)* that appears as a clause by itself is wrong, whether it appears as a clause resulting from translation of an English statement, or as the result of a resolution step. (Note that *x* is a universally quantified variable; *P(a)* , where *a* is a constant, is okay.) The rationale for this rule is that if *P(x)* is true, then *P* is true of *everything* and therefore cannot be useful for reasoning.

7. More generally, the literals of a clause should be connected by the variables they contain. Imagine that lines are drawn between occurrences of each variable, i.e., all the occurrences of *x* are connected, etc. The clause should be completely connected; if any literal is disconnected from the rest of the clause, it probably indicates an error. Moreover, each variable should appear at least twice.

8. Constants other than Skolem constants are generally numbers or things that would be written as proper names in English: *3 John Austin* . A common noun written as a constant is probably incorrect. For example, ``Every boy loves some dog'' can be written: $\forall x\ (BOY(x) \rightarrow \exists y\ (DOG(y) \wedge LOVES(x,y)))$ but not: $\forall x\ (BOY(x) \rightarrow LOVES(x,dog))$. Note that in the latter form there is nothing to say that the constant *dog* is a dog.

9. Every time a Skolem constant or Skolem function is created, it must be a new one. By convention, Skolem constants are represented by letters $a\ b\ c$ and Skolem functions by $f()\ g()\ h()$.

10. Different Skolem constants and functions cannot be unified. If you find yourself wanting to unify two different Skolems, don't do it! Instead, use this information to find your bug: one of the Skolems should be a universally quantified variable.

11. Certain common forms can be translated to clause form by inspection. ``Every P is Q'' will translate to:
    $\neg P(x) \vee Q(x)$.
    More generally, ``Every P that is R and S and ... is Q'' will translate to:
    $\neg P(x) \vee \neg R(x) \vee \neg S(x) \vee \ldots \vee Q(x)$.

12. Statements with an ``and'' in the conclusion will result in multiple clauses. For example, ``Every cat is soft and furry'' results in clauses:
    1. $\neg CAT(x) \vee SOFT(x)$
    2. $\neg CAT(x) \vee FURRY(x)$

13. Use brackets liberally to avoid making mistakes in algebraic manipulations. For example, in writing the negated conclusion,
    1. Write the conclusion in positive form.
    2. Put large brackets around it.
    3. Put a negation sign in front of the bracket.

14. An ``If [...] then [...]'' should be translated as: $[\ [\ldots] \rightarrow [\ldots]\ ]$. Use brackets to make sure your manipulations are correct.

15. If the conclusion is of the form ``If *condition* then *result* '' you can often simplify the algebra by:
    1. Writing the *condition* as additional positive axioms.
    2. Negating the *result*.

    Justification: ``If C then R'' is written $[\ C \rightarrow R]$, which is negated:
    $$\neg [\ C \rightarrow R] = \neg [\ \neg C \vee R\ ] = [\ C \wedge \neg R\ ] .$$
    C and R must have no variables in common to be separated in this manner.

16. Use your knowledge of the real-world situation to guide the resolution steps you make, so that the resulting search will be short. Use the strategies *set of support* (use the negated conclusion, which will often involve constants) and *unit preference* (resolve with shorter clauses).

Gordon S. Novak Jr.