Example of Resolution

The example we present here is one to which all AI students should be exposed at some point in their studies. It is the famous "monkey and bananas problem," another one of those complex real life problems solvable with AI techniques. We envision a room containing a monkey, a chair, and some bananas that have been hung from the center of the ceiling, out of reach from the monkey. If the monkey is clever enough, he can reach the bananas by placing the chair directly below them and climbing on top of the chair. The problem is to use FOPL to represent this monkey-banana world and, using resolution, prove the monkey can reach the bananas.

In creating a knowledge base, it is essential first to identify all relevant objects which will play some role in the anticipated inferences. Where possible, irrelevant objects should be omitted, but never at the risk of incompleteness. For example, in the current problem, the monkey, bananas, and chair are essential. Also needed is some reference object such as the floor or ceiling to establish the height relationship between monkey and bananas. Other objects such as windows, walls or doors are not relevant.

The next step is to establish important properties of objects, relations between them, and any assertions likely to be needed. These include such facts as the chair is tall enough to raise the monkey within reach of the bananas, the monkey is dexterous, the chair can be moved under the bananas, and so on. Again, all important properties, relations, and assertions should be included and irrelevant ones omitted. Otherwise, unnecessary inference steps may be taken.

The important factors for our problem are described below, and all items needed for the actual knowledge base are listed as axioms. These are the essential facts and rules. Although not explicitly indicated, all variables are universally quantified.

Relevant factors for the problem

CONSTANTS

{floor, chair, bananas, monkey}

VARIABLES

{x, y, z}

PREDICATES

{can_reach(x,y)	; x can reach y
dexterous (x)	; x is a dexterous animal
close(x,y)	; x is close to y
get_on(x,y)	; x can get on y
under(x,y)	; x is under y
tall(x)	; x is tall
in_room(x)	; x is in the room
can_move(x,y,z)	; x can move y near z
can_climb(x,y)}	; x can climb onto y

AXIOMS

{in_room(bananas)
in_room(chair)
in_room(monkey)
dexterous(monkey)
tall(chair)
close(bananas,floor)
can_move(monkey,chair,bananas)
can_climb(monkey,chair)
(dexterous(x) & close(x,y) → can-reach(x,y)
((get_on(x,y) & under(y,bananas) & tall(y) →
close(x,bananas))
((in_room(x) & in_room(y) & in_room(z) & can_move(x,y,z))
→ close(z,floor) V under(y,z))
(can_climb(x,y) → get_on(x,y))}

Using the above axioms, a knowledge base can be written down directly in the required clausal form. All that is needed to make the necessary substitutions are the equivalences

$$P \rightarrow Q = \bar{}P \vee Q$$

and De Morgan's laws. To relate the clauses to a LISP program, one may prefer to think of each clause as being a list of items. For example, number 9, below, would be written as

(or (¯can_climb(?x ?y) get_on(?x ?y))

where ?*x* and ?*y* denote variables.

Note that clause 13 is not one of the original axioms. It has been added for the proof as required in refutation resolution proofs.

Clausal form of knowledge base

1. in_room(monkey)
2. in_room(bananas)
3. in_room(chair)
4. tall(chair)
5. dexterous(monkey)
6. can_move(monkey, chair, bananas)
7. can_climb(monkey,chair)
8. ¯close(bananas,floor)
9. ¯can_climb(x,y) V get_on(x,y)
10. ¯dexterous(x) V ¯close(x,y) V can_reach(x,y)
11. ¯get_on(x,y) V ¯under(y,bananas) V ¯tall(y) V close(x,bananas)
12. ¯in_room(x) V ¯in_room(y) V ¯in_room(z) V ¯can_move(x,y,z) V close(y,floor) V under(y,z)
13. ¯can_reach(monkey,bananas)

Resolution proof. A proof that the monkey can reach the bananas is summarized below. As can be seen, this is a refutation proof where the statement to be proved (can_reach(monkey,bananas)) has been negated and added to the knowledge base (number 13). The proof then follows when a contradiction is found (see number 23, below).

14. ¯can_move(monkey,chair,bananas) V close(bananas,floor) V under (chair,bananas) ; 14 is a resolvent of 1,2,3 and 12 with substitution {monkey/*x*, chair/*y*, bananas/*z*}

15.	close(bananas,floor) V under (chair,bananas)	; this is a *resolvent of 6 and 14*
16.	under(chair,bananas)	; this is a resolvent of 8 and 15
17.	~get_on(x,chair) V ~tall(chair) V close(x,bananas)	; this is a resolvent of 11 and 16 with substitution (chair/*y*}
18.	~get_on(x,chair) V close(x,bananas)	; a resolvent of 4 and 17
19.	get_on(monkey,chair)	; a resolvent of 7 and 9
20.	close(monkey,bananas)	; a resolvent of 18 and 19 with substitution (monkey/*x*}
21.	~close(monkey,y) V can_reach (monkey,y)	; a resolvent of 10 and 5 with substitution (monkey/*x*}
22.	reach(monkey,bananas)	; a resolvent of 20 and 21 with substitution {bananas/*y*}
23.	[ ]	; a resolvent of 13 and 22

In performing the above proof, no particular strategy was followed. Clearly, however, good choices were made in selecting parent clauses for resolution. Otherwise, many unnecessary steps may have been taken before completing the proof. Different forms of resolution were completed in steps 14 through 23. One of the exercises requires that the types of resolutions used be identified