"a bluebird is a small blue-colored bird and a bird is a feathered flying vertebrate."

This may be represented as a set of logical predicates:

hassize(bluebird,small).

hascovering(bird,feathers).

hascolor(bluebird,blue).

hasproperty(bird,flies).

isa(bluebird,bird).

isa(bird,vertebrate).

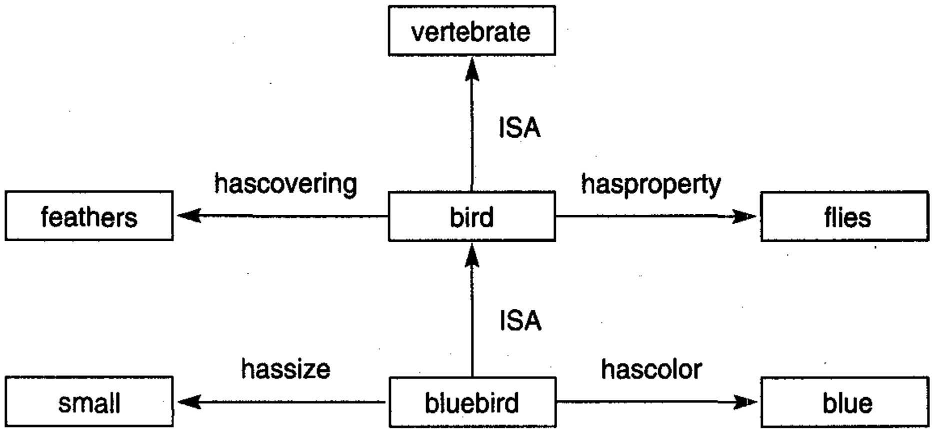


Figure II.4 Semantic network description of a bluebird.

If it doesn't rain tomorrow, Tom will go to the mountains.
$\neg$ weather(rain,tomorrow) $\Rightarrow$ go(tom,mountains)

Emma is a Doberman pinscher and a good dog.
gooddog(emma) $\wedge$ isa(emma,doberman)

All basketball players are tall.
$\forall$ X (basketball_player(X) $\Rightarrow$ tall(X))

Some people like anchovies.
$\exists$ X (person(X) $\wedge$ likes(X,anchovies)).

If wishes were horses, beggars would ride.
equal(wishes,horses) $\Rightarrow$ ride(beggars).

Nobody likes taxes.
$\neg \exists$ X likes(X,taxes).

2.3.3 A Unification Example

The behavior of the preceding algorithm may be clarified by tracing the call

unify((parents X (father X) (mother bill)), (parents bill (father bill) Y)).

When unify is first called, because neither argument is an atomic symbol, the function will attempt to recursively unify the first elements of each expression, calling

unify(parents, parents).

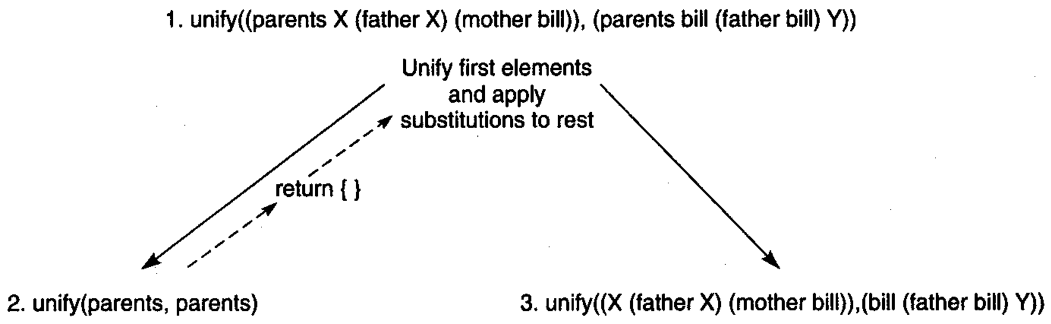

1. unify((parents X (father X) (mother bill)), (parents bill (father bill) Y))
Unify first elements and apply substitutions to rest
return { }
2. unify(parents, parents)
3. unify((X (father X) (mother bill)),(bill (father bill) Y))

Figure 2.5 Initial steps in the unification of (parents X (father X) (mother bill)) and (parents bill (father bill) Y).

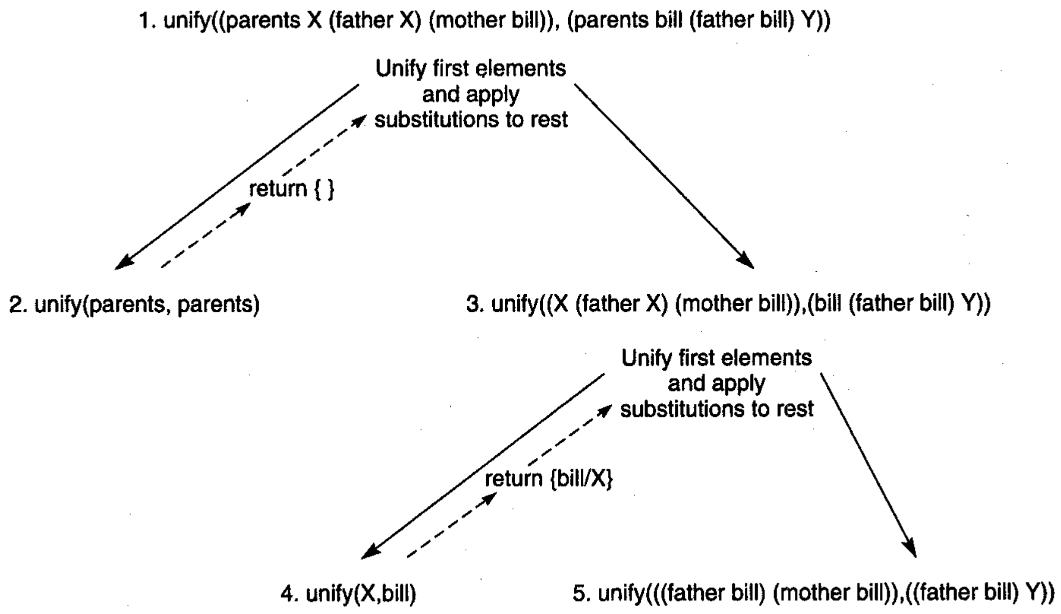

1. unify((parents X (father X) (mother bill)), (parents bill (father bill) Y))
Unify first elements and apply substitutions to rest
return { }
2. unify(parents, parents)
3. unify((X (father X) (mother bill)),(bill (father bill) Y))
Unify first elements and apply substitutions to rest
return {bill/X}
4. unify(X,bill)
5. unify(((father bill) (mother bill)),((father bill) Y))

Figure 2.6 Further steps in the unification of (parents X (father X) (mother bill)) and (parents bill (father bill) Y).

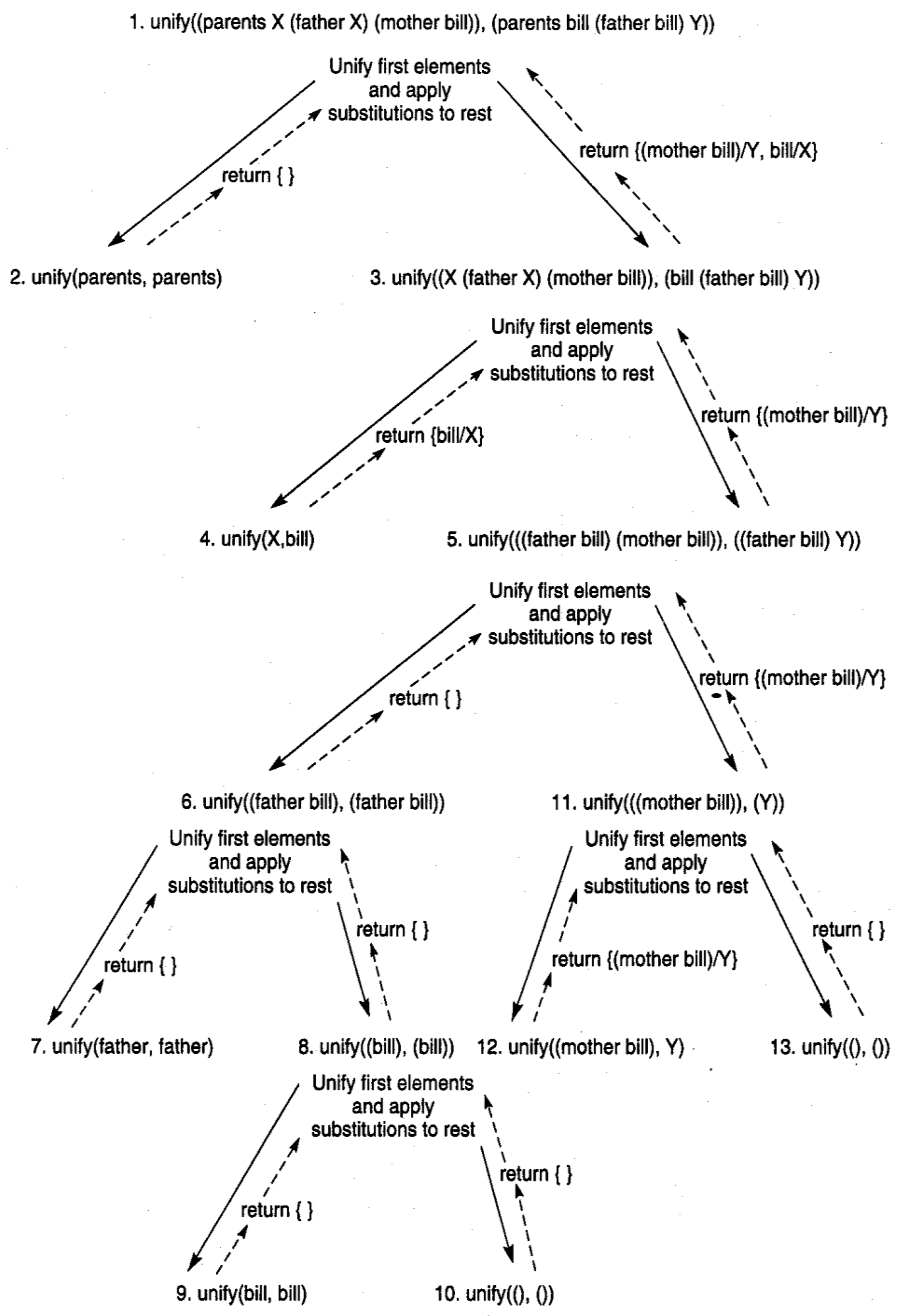

1. unify((parents X (father X) (mother bill)), (parents bill (father bill) Y))
Unify first elements and apply substitutions to rest
return { }
return {(mother bill)/Y, bill/X}
2. unify(parents, parents)
3. unify((X (father X) (mother bill)), (bill (father bill) Y))
Unify first elements and apply substitutions to rest
return {bill/X}
return {(mother bill)/Y}
4. unify(X,bill)
5. unify(((father bill) (mother bill)), ((father bill) Y))
Unify first elements and apply substitutions to rest
return { }
return {(mother bill)/Y}
6. unify((father bill), (father bill))
11. unify(((mother bill)), (Y))
Unify first elements and apply substitutions to rest
return { }
return { }
Unify first elements and apply substitutions to rest
return {(mother bill)/Y}
return { }
7. unify(father, father)
8. unify((bill), (bill))
12. unify((mother bill), Y)
13. unify((), ())
Unify first elements and apply substitutions to rest
return { }
return { }
9. unify(bill, bill)
10. unify((), ())

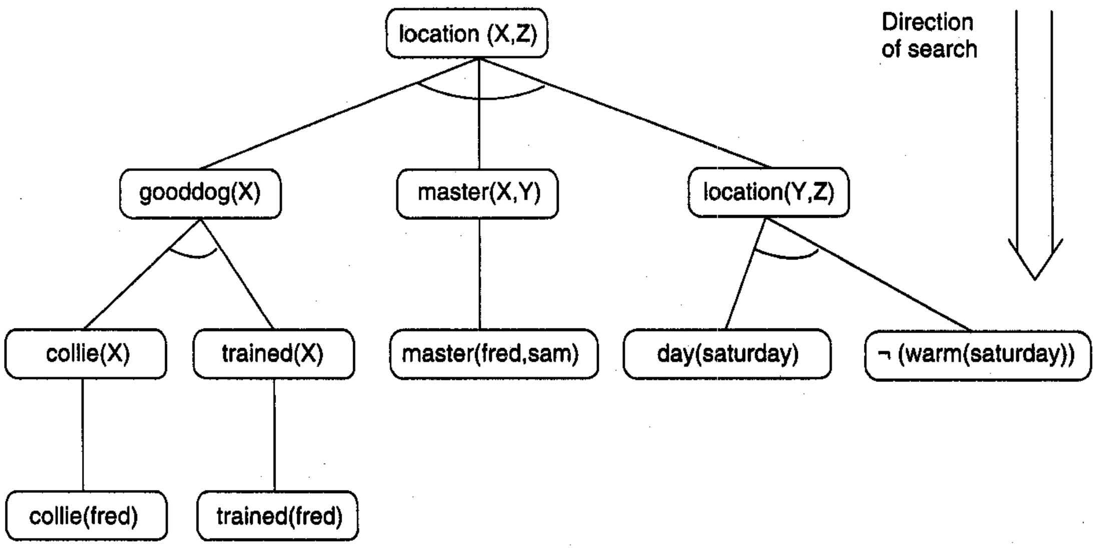
Figure 2.7 Final trace of the unification of (parents X (father X) (mother bill)) and (parents bill (father bill) Y).

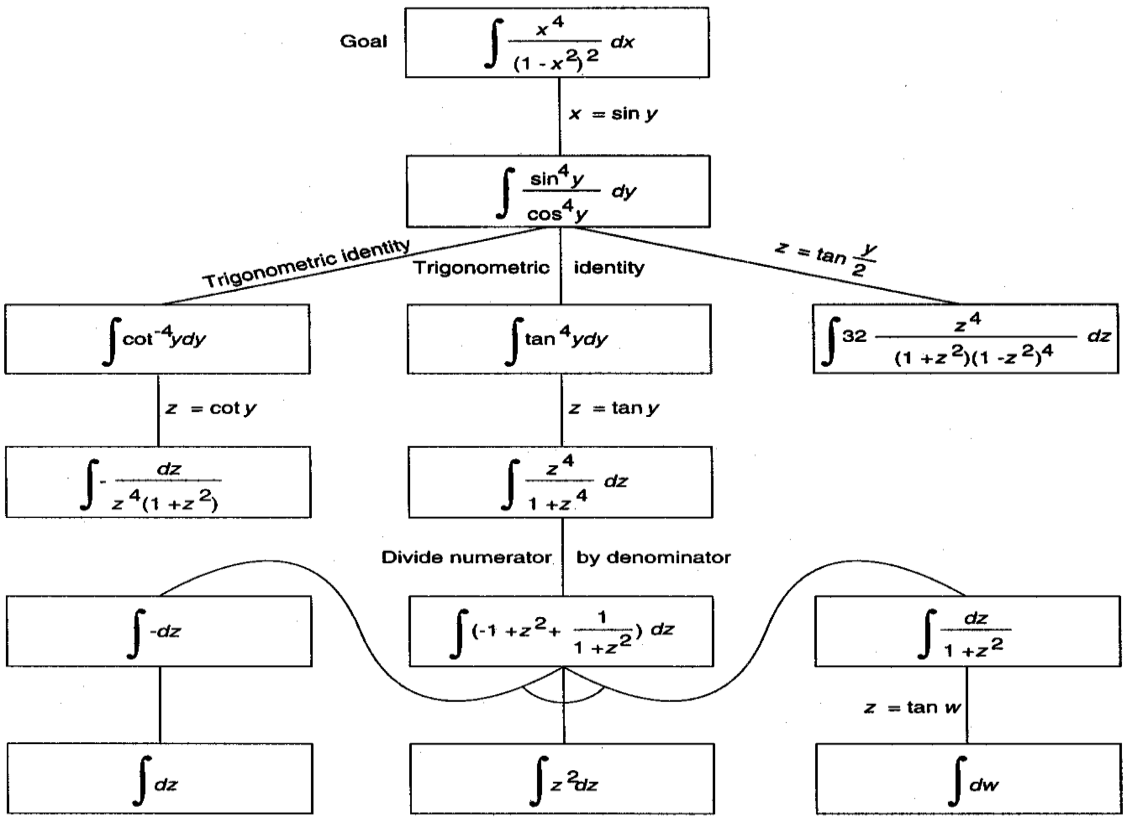
EXAMPLE 3.3.4

This example is taken from the predicate calculus and represents a goal-driven graph search where the goal to be proved true is a predicate calculus expression, often containing variables. The axioms are the logical descriptions of a relationship between a dog, Fred, and his master, Sam.

The facts and rules of this example are given as English sentences followed by their predicate calculus equivalents:

1. Fred is a collie
   collie(fred).
2. Sam is Fred's master.
   master(fred,sam).
3. It is Saturday.
   day(saturday).
4. It is cold on Saturday.
   ¬ (warm(saturday)).
5. Fred is a trained dog.
   trained(fred).
6. Spaniels or collies that are trained are good dogs.
   ∀ X[spaniel(X) ∨ (collie(X) ∧ trained(X)) ⇒ gooddog(X)]
7. If a dog is a good dog and has a master then he will be with his master.
   ∀ (X,Y,Z) [gooddog(X) ∧ master(X,Y) ∧ location(Y,Z) ⇒ location(X,Z)]
8. If it is Saturday and warm, then Sam is at the park.
   day(saturday) ∧ warm(saturday) ⇒ location(sam,park).
9. If it is Saturday and not warm, then Sam is at the museum.
   day(saturday) ∧ ¬ (warm(saturday)) ⇒ location(sam,museum).

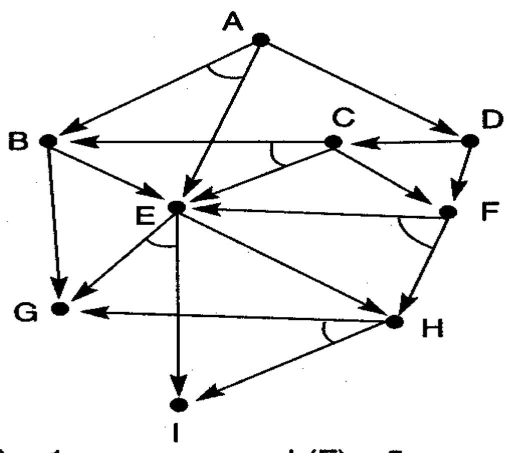




Goal
$\int \frac{x^4}{(1-x^2)^2} dx$
$x = \sin y$
$\int \frac{\sin^4 y}{\cos^4 y} dy$
Trigonometric identity
Trigonometric identity
$z = \tan \frac{y}{2}$
$\int \cot^{-4} y dy$
$\int \tan^4 y dy$
$\int 32 \frac{z^4}{(1+z^2)(1-z^2)^4} dz$
$z = \cot y$
$z = \tan y$
$\int - \frac{dz}{z^4(1+z^2)}$
$\int \frac{z^4}{1+z^4} dz$
Divide numerator by denominator
$\int -dz$
$\int (-1+z^2+\frac{1}{1+z^2}) dz$
$\int \frac{dz}{1+z^2}$
$z = \tan w$
$\int dz$
$\int z^2 dz$
$\int dw$

Figure 3.22 And/or graph representing part of the state space for integrating a function.

b. Use this algorithm to search the graph in Figure 4.25.



h(A) = 1	h(F) = 5
h(B) = 2	h(G) = 1
h(C) = 5	h(H) = 3
h(D) = 3	h(I) = 1
h(E) = 2	

Figure 4.25

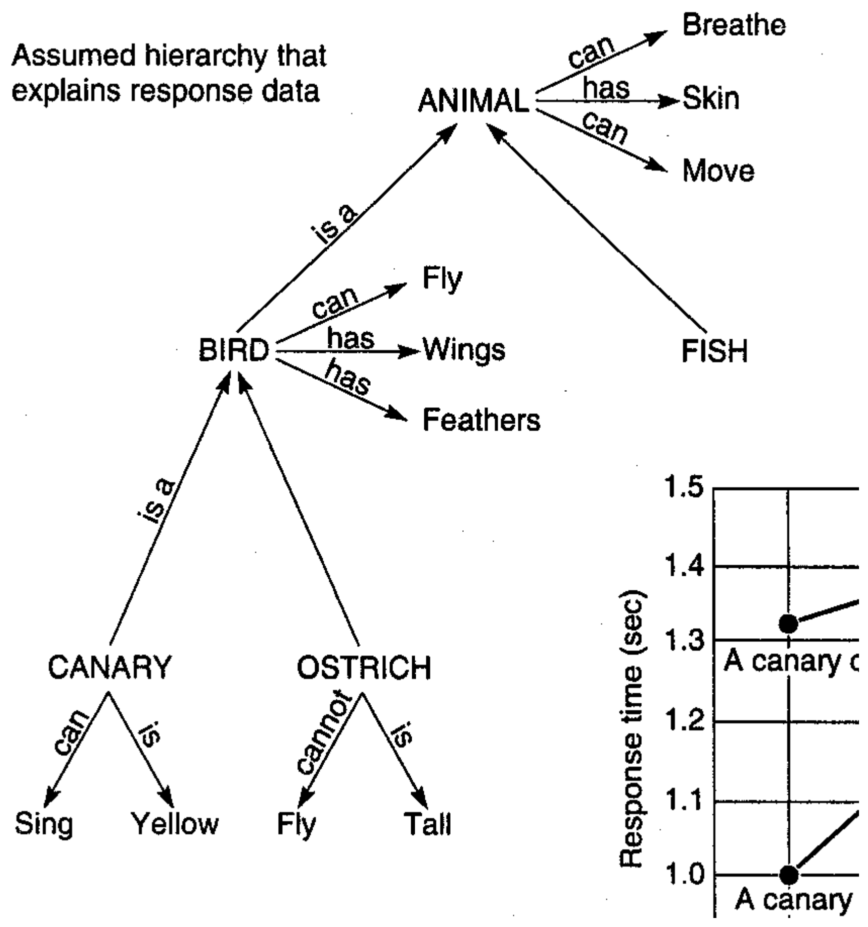
∀ X good_student(X) ∧ M study_hard(X) ⇒ graduates(X)
∀ Y party_person(Y) ∧ M not(study_hard(Y)) ⇒ not(graduates(Y))
good_student(peter)
party_person(peter)

∀ Y very_smart(Y) ∧ M not(study_hard(Y)) ⇒ not(study_hard(Y))
∀ X not(very_smart(X)) ∧ M not(study_hard(X)) ⇒ not(study_hard(X))

From these clauses we can infer a new clause:

∀ Z M not(study_hard(Z)) ⇒ not(study_hard(Z))

∀ X good_student(X) ∧ M study_hard(X) ⇒ study_hard(X)
∀ Y party_person(Y) ⇒ not (study_hard(Y))
good_student(peter)
party_person(peter)



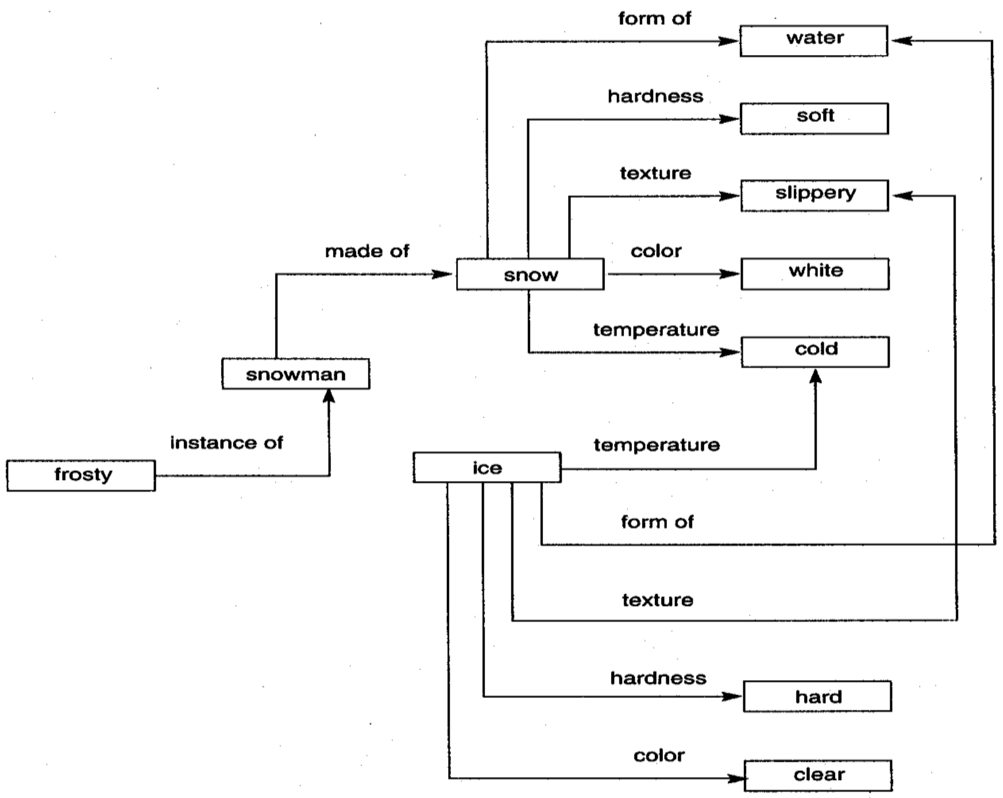


Figure 8.2 Network representation of properties of snow and ice.

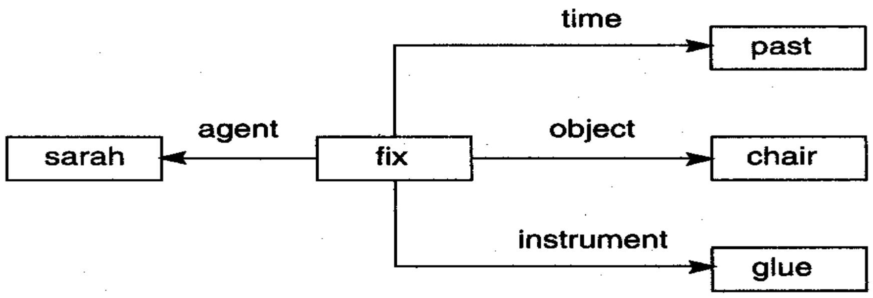
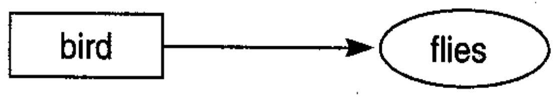
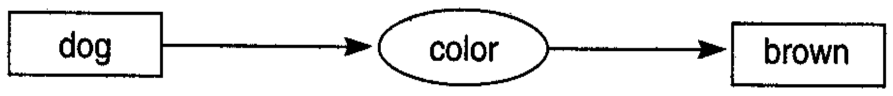


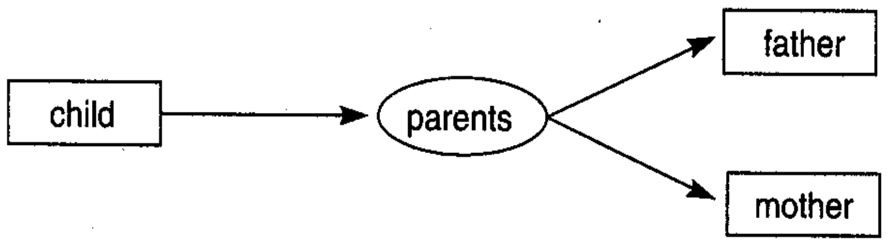
Figure 8.5 Case frame representation of the sentence "Sarah fixed the chair with glue."



Flies is a 1-ary relation.



Color is a 2-ary relation.



Parents is a 3-ary relation.

Figure 8.11 Conceptual relations of different arities.

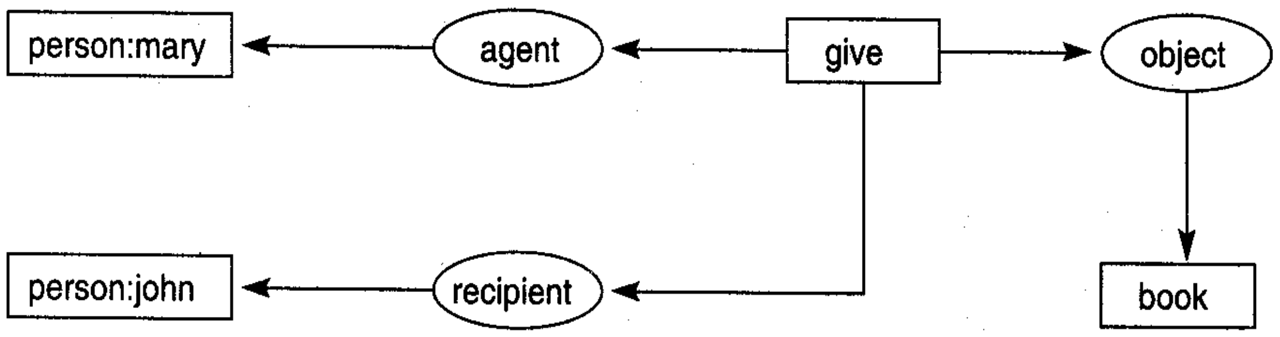


Figure 8.12 Conceptual graph of the sentence "Mary gave John the book."

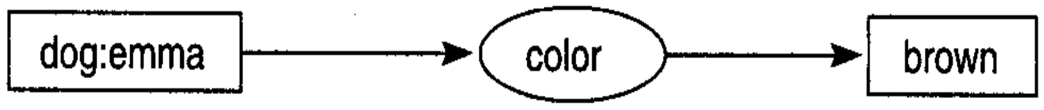


Figure 8.13 Conceptual graph indicating that the dog named **emma** is brown.

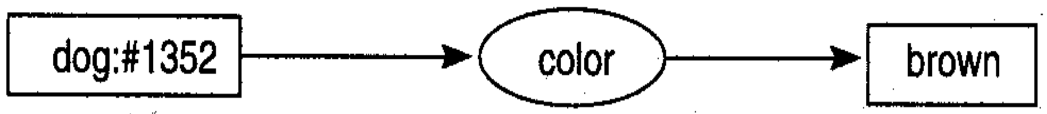


Figure 8.14 Conceptual graph indicating that a particular (but unnamed) dog is brown.

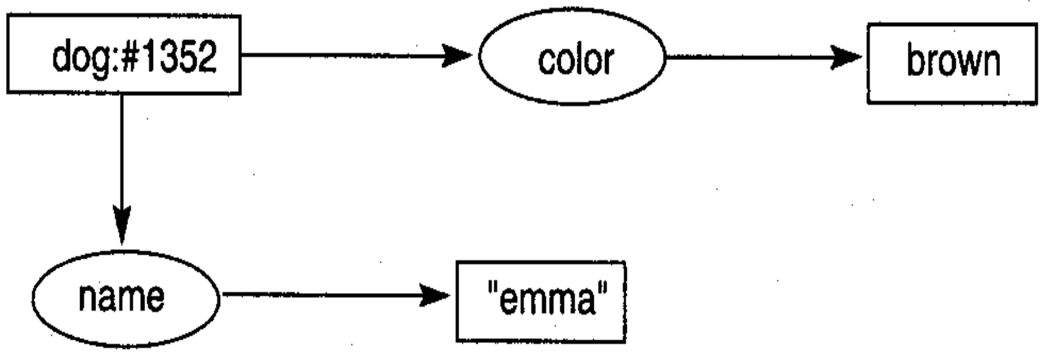


Figure 8.15 Conceptual graph indicating that a dog named **emma** is brown.

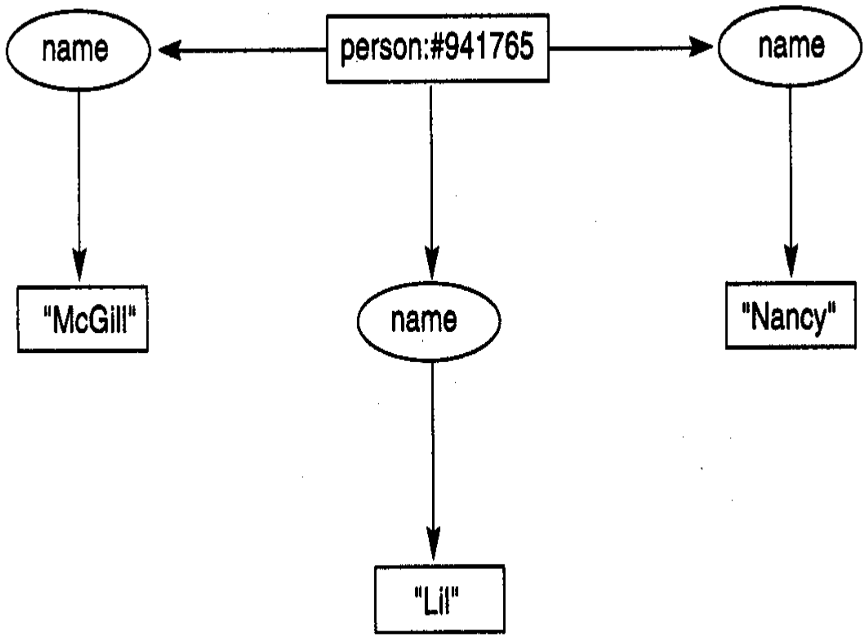


Figure 8.16 Conceptual graph of a person with three names.

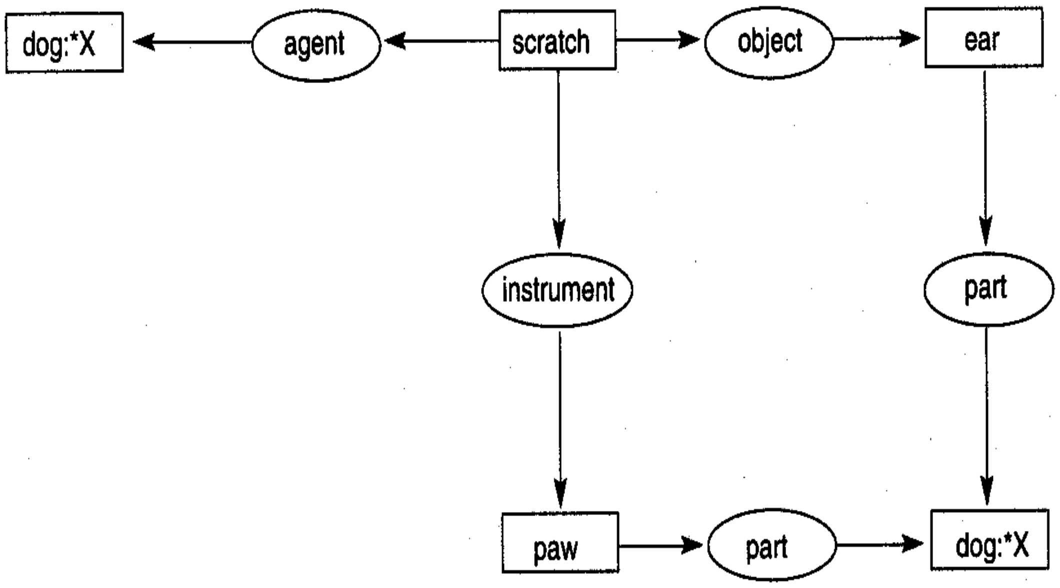


Figure 8.17 Conceptual graph of the sentence "The dog scratches its ear with its paw."

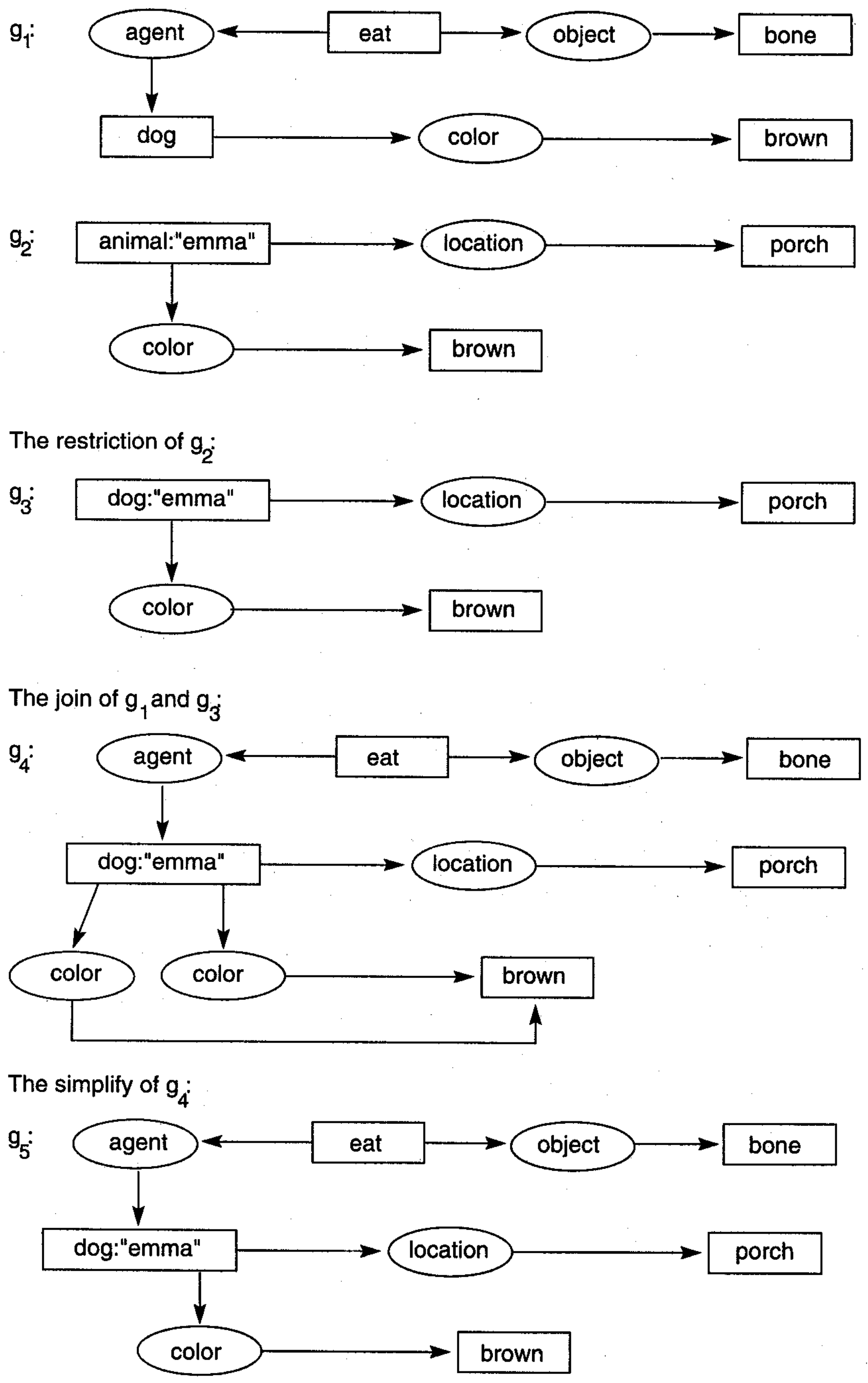


Figure 8.19 Examples of restrict, join, and simplify operations.

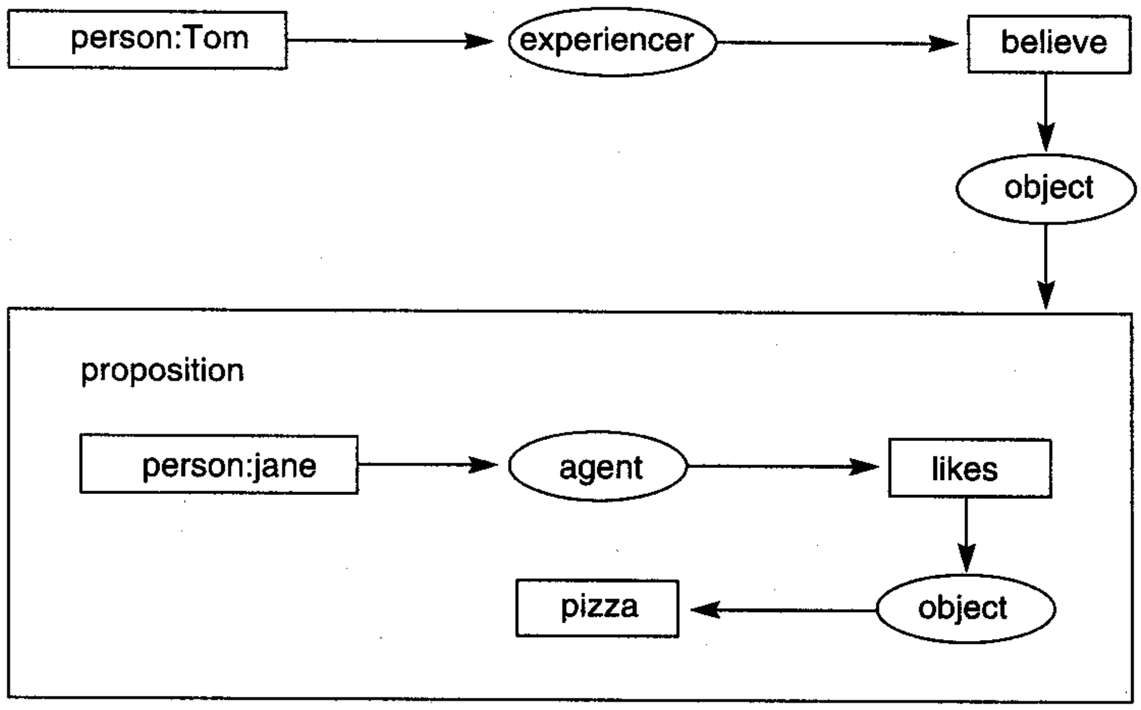


Figure 8.21 Conceptual graph of the statement "Tom believes that Jane likes pizza," showing the use of a propositional concept.

$$\exists X \exists Y\ (\mathrm{dog}(X) \wedge \mathrm{color}(X,Y) \wedge \mathrm{brown}(Y)).$$

$$\forall X \forall Y\ (\neg\ (\mathrm{dog}(X) \wedge \mathrm{color}(X,Y) \wedge \mathrm{pink}(Y))).$$

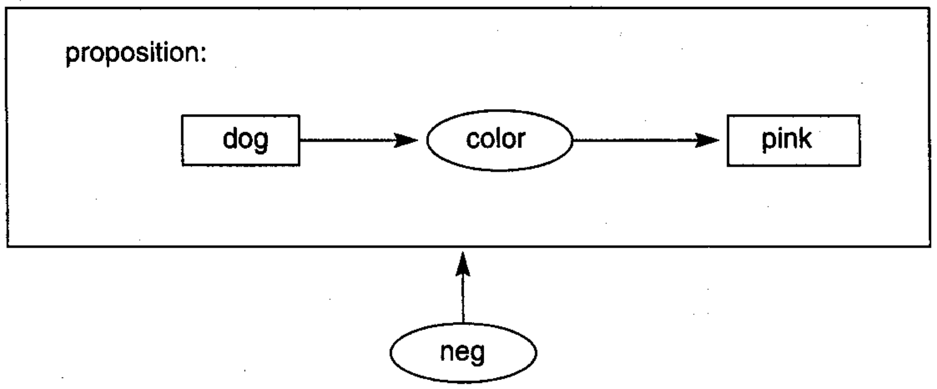


Figure 8.22 Conceptual graph of the proposition "There are no pink dogs."

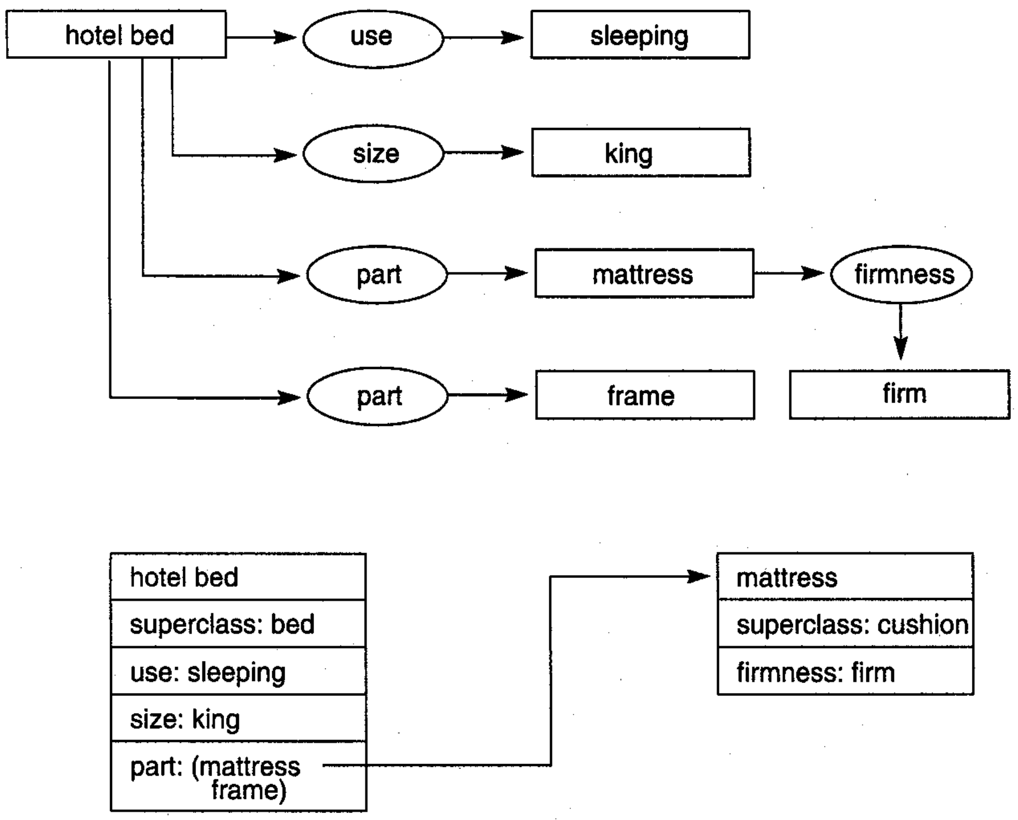


Figure 8.23 Conceptual graph and frame descriptions of a hotel bed.

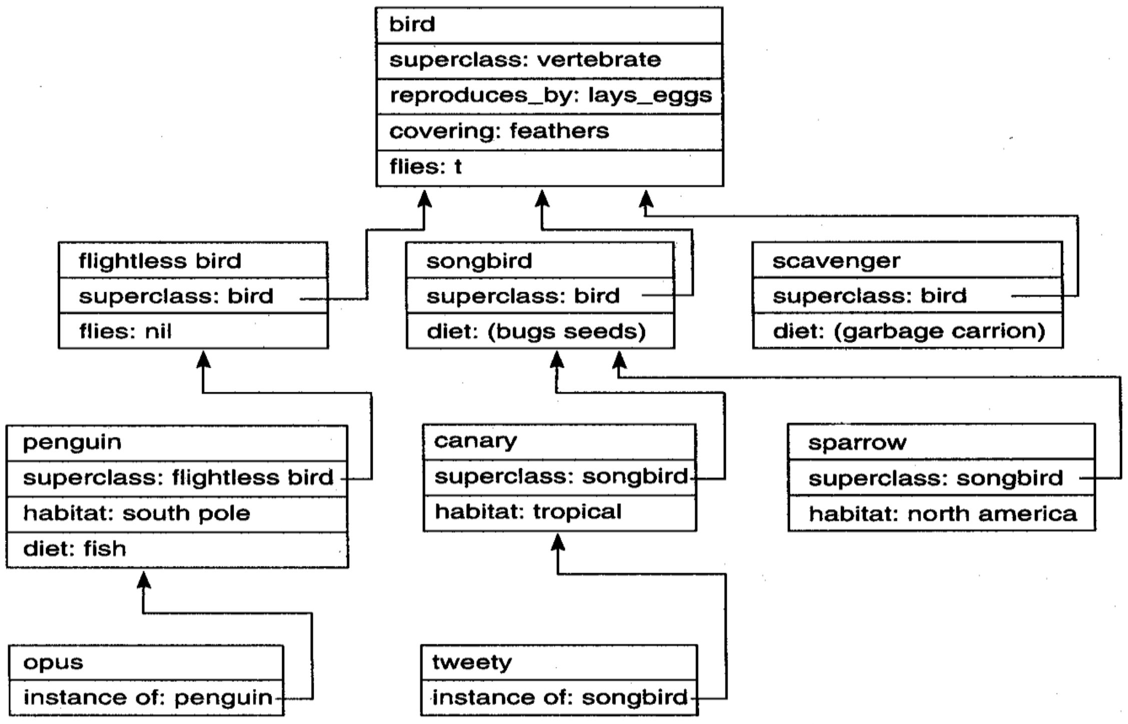
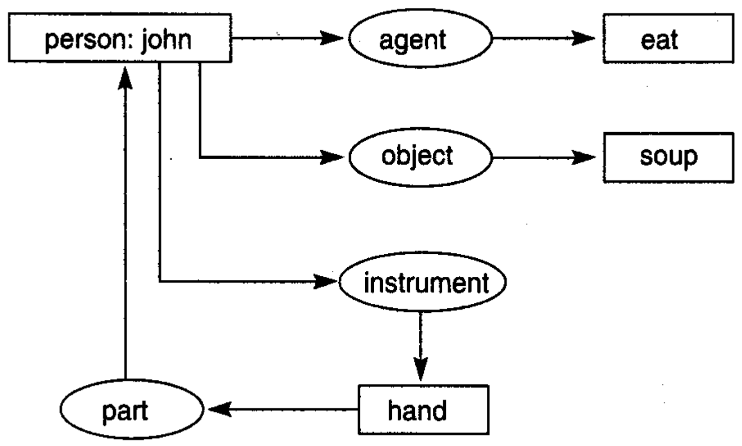


Figure 8.27 Inheritance system description of birds.



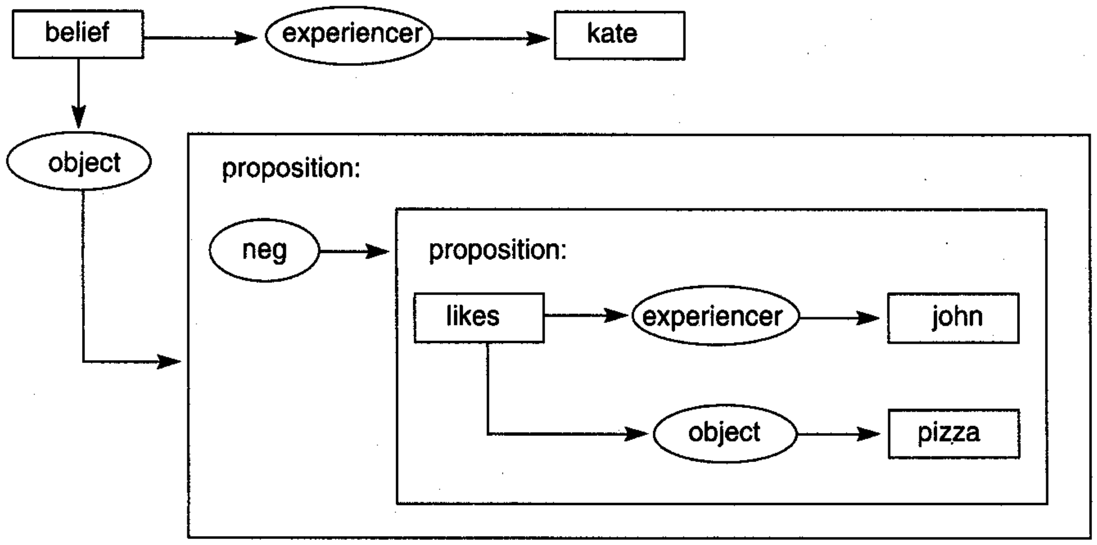


Figure 8.31 Two conceptual graphs to be translated into English.

```
isa (canary, bird).                  isa (robin, bird).
isa (ostrich, bird).                 isa (penguin, bird).
isa (bird, animal).                  isa (fish, animal).
isa (opus, penguin).                 isa (tweety, canary).
```

```
hasprop (tweety, color, white).      hasprop (robin, color, red).
hasprop (canary, color, yellow).     hasprop (penguin, color, brown).
hasprop (bird, travel, fly).         hasprop (fish, travel, swim).
hasprop (ostrich, travel, walk).     hasprop (penguin, travel, walk).
hasprop (robin, sound, sing).        hasprop (canary, sound, sing).
hasprop (bird, cover, feathers).     hasprop (animal, cover, skin).
```

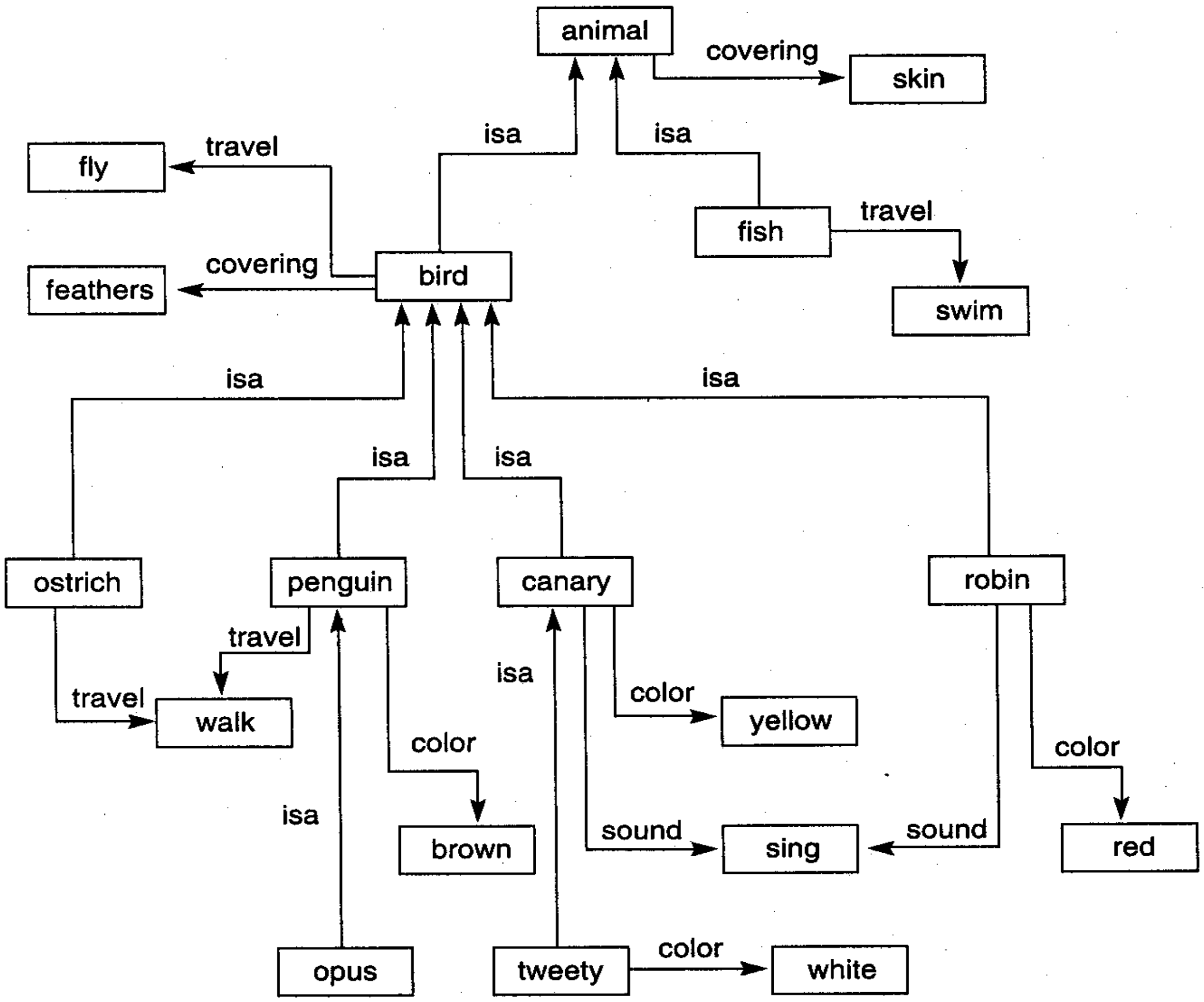


Figure 9.7 Portion of a semantic network describing birds and other animals.

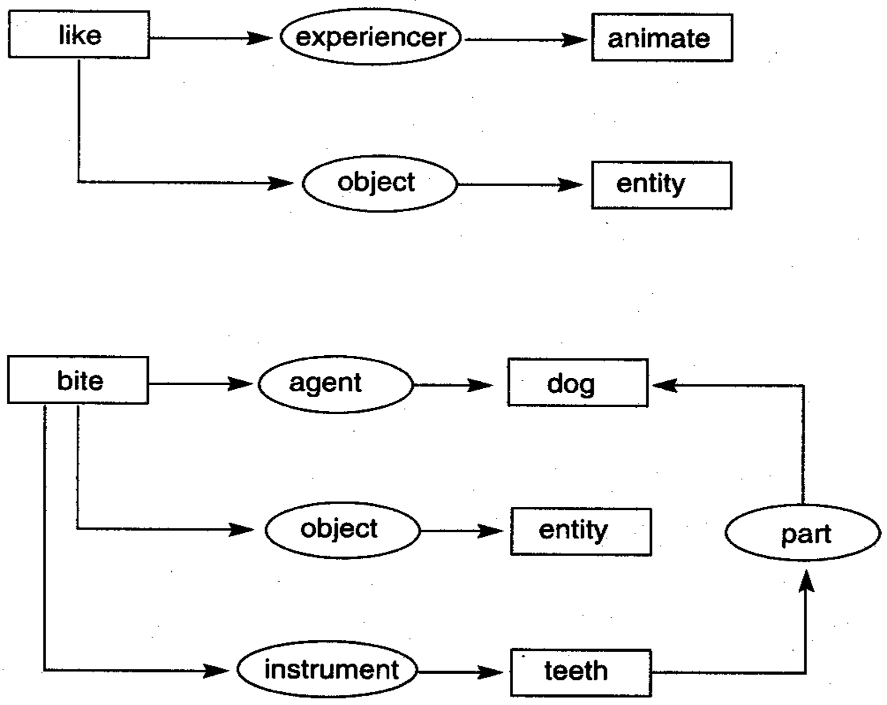
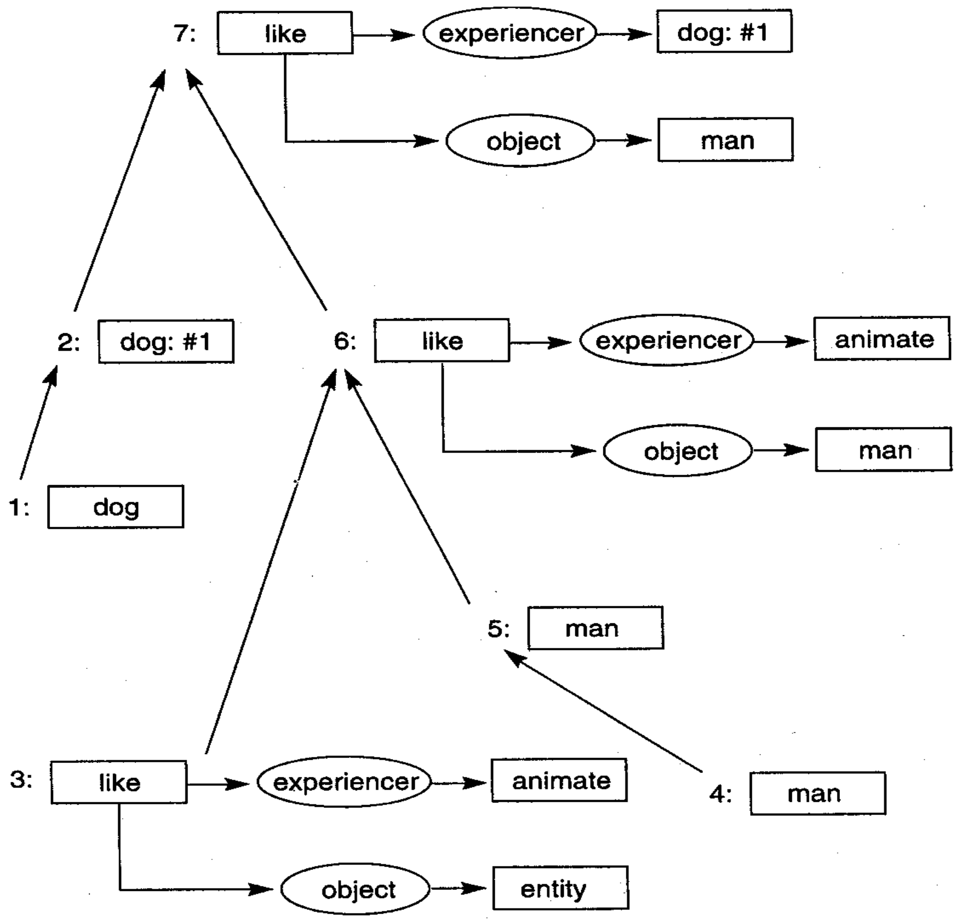


Figure 11.11 Case frames for the verbs "like" and "bite."



we use two types of brackets: ( ) and [ ]. Where possible in the derivation, we remove redundant brackets: The expression we will reduce to clause form is:

(i) $(\forall X)([a(X) \wedge b(X)] \rightarrow [c(X,I) \wedge (\exists Y)((\exists Z)[c(Y,Z)] \rightarrow d(X,Y))]) \vee (\forall X)(e(X))$

1. First we eliminate the $\rightarrow$ by using the equivalent form proved in Chapter 2: $a \rightarrow b \equiv \neg a \vee b$. This transformation reduces the expression in (i) above:

   (ii) $(\forall X)(\neg [a(X) \wedge b(X)] \vee [c(X,I) \wedge (\exists Y)(\neg (\exists Z)[c(Y,Z)] \vee d(X,Y))]) \vee (\forall X)(e(X))$

2. Next we reduce the scope of negation. This may be accomplished using a number of the transformations of Chapter 2. These include:

   $\neg (\neg a) \equiv a$
   $\neg (\exists X)\ a(X) \equiv (\forall X) \neg a(X)$
   $\neg (\forall X)\ b(X) \equiv (\exists X) \neg b(X)$
   $\neg (a \wedge b) \equiv \neg a \vee \neg b$
   $\neg (a \vee b) \equiv \neg a \wedge \neg b$

   Using the second and fourth equivalences (ii) becomes:

   (iii) $(\forall X)([\neg a(X) \vee \neg b(X)] \vee [c(X,I) \wedge (\exists Y)((\forall Z)[\neg c(Y,Z)] \vee d(X,Y))]) \vee (\forall X)(e(X))$

3. Next we standardize by renaming all variables so that variables bound by different quantifiers have unique names. As indicated in Chapter 2, because variable names are "dummies" or "place holders," the particular name chosen for a variable does not affect either the truth value or the generality of the clause. Transformations used at this step are of the form:

   $((\forall X)a(X) \vee (\forall X)b(X)) \equiv (\forall X)a(X) \vee (\forall Y)b(Y)$

   Because (iii) has two instances of the variable X, we rename:

   (iv) $(\forall X)([\neg a(X) \vee \neg b(X)] \vee [c(X,I) \wedge (\exists Y)((\forall Z)\ [\neg c(Y,Z)] \vee d(X,Y))]) \vee (\forall W)(e(W))$

4. Move all quantifiers to the left without changing their order. This is possible because step 3 has removed the possibility of any conflict between variable names. (iv) now becomes:

   (v) $(\forall X)(\exists Y)(\forall Z)(\forall W)([\neg a(X) \vee \neg b(X)] \vee [c(X,I) \wedge (\neg c(Y,Z) \vee d(X,Y))] \vee e(W))$

   After step 4 the clause is said to be in *prenex normal* form, because all the quantifiers are in front as a *prefix* and the expression or *matrix* follows after.

5. At this point all existential quantifiers are eliminated by a process called *skolemization.* Expression (v) has an existential quantifier for Y. When an expression contains an existentially quantified variable, for example, $(\exists Z)(foo(\ldots, Z, \ldots))$, it may be concluded that there is an assignment to Z under which foo is true. Skolemization identifies such a value. Skolemization does not necessarily show *how* to produce such a value; it is only a method for giving a name to an assignment that *must* exist. If k represents that assignment, then we have $foo(\ldots, k, \ldots)$. Thus:

   $(\exists X)(dog(X))$ may be replaced by $dog(fido)$

   where the name fido is picked from the domain of definition of X to represent that individual X. fido is called a *skolem constant.* If the predicate has more than one argument and the existentially quantified variable is within the scope of universally quantified variables, the existential variable must be a function of those other variables. This is represented in the skolemization process:

   $(\forall X)\ (\exists Y)\ (mother(X,Y))$

   This expression indicates that every person has a mother. Every person is an X and the existing mother will be a function of the particular person X that is picked. Thus skolemization gives:

   $(\forall X) mother(X, m(X))$

   which indicates that each X has a mother (the m of that X). In another example:

   $(\forall X)(\forall Y)(\exists Z)(\forall W)(foo(X,Y,Z,W))$

   is skolemized to:

   $(\forall X)(\forall Y)(\forall W)(foo(X,Y,f(X,Y),W))$.

   We note that the existentially quantified Z was within the scope (to the right of) universally quantified X and Y. Thus the skolem assignment is a function of X and Y but not of W. With skolemization (v) becomes:

   (vi) $(\forall X)(\forall Z)(\forall W)([\neg a(X) \vee \neg b(X)] \vee [c(X,I) \wedge (\neg c(f(X),Z) \vee d(X,f(X)))]) \vee e(W))$

   where f is the skolem function of X that replaces the existential Y. Once the skolemization has occurred, step 6 can take place, which simply drops the prefix.

6. Drop all universal quantification. By this point only universally quantified variables exist (step 5) with no variable conflicts (step 3). Thus all quantifiers can be dropped, and any proof procedure employed assumes all variables are universally quantified. Formula (vi) now becomes:

(vii) $[\neg a(X) \vee \neg b(X)] \vee [c(X,I) \wedge (\neg c(f(X),Z) \vee d(X,f(X)))] \vee e(W)$

7. Next we convert the expression to the conjunct of disjuncts form. This requires using the associative and distributive properties of $\wedge$ and $\vee$. Recall from Chapter 2 that

   $a \vee (b \vee c) = (a \vee b) \vee c$

   $a \wedge (b \wedge c) = (a \wedge b) \wedge c$

   which indicates that $\wedge$ or $\vee$ may be grouped in any desired fashion. The distributive property of Chapter 2 is also used, when necessary. Because

   $a \wedge (b \vee c)$

   is already in clause form, $\wedge$ is not distributed. However, $\vee$ must be distributed across $\wedge$ using:

   $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

   The final form of (vii) is:

   (viii) $[\neg a(X) \vee \neg b(X) \vee c(X,I) \vee e(W)] \wedge$
   $[\neg a(X) \vee \neg b(X) \vee \neg c(f(X),Z) \vee d(X,f(X)) \vee e(W)]$

8. Now call each conjunct a separate clause. In the example (viii) above there are two clauses:

   (ixa) $\neg a(X) \vee \neg b(X) \vee c(X,I) \vee e(W)$

   (ixb) $\neg a(X) \vee \neg b(X) \vee \neg c(f(X),Z) \vee d(X,f(X)) \vee e(W)$

9. The final step is to *standardize the variables apart* again. This requires giving the variable in each clause generated by step 8 different names. This procedure arises from the equivalence established in Chapter 2 that

   $(\forall X)(a(X) \wedge b(X)) \equiv (\forall X)\, a(X) \wedge (\forall Y)\, b(Y)$

   which follows from the nature of variable names as place holders. (ixa) and (ixb) now become, using new variable names U and V:

   (xa) $\neg a(X) \vee \neg b(X) \vee c(X,I) \vee e(W)$

   (xb) $\neg a(U) \vee \neg b(U) \vee \neg c(f(U),Z) \vee d(U,f(U)) \vee e(V)$

The importance of this final standardization becomes apparent only as we present the unification steps of resolution. We find the most general unification to make two predicates within two clauses equivalent, and then this substitution is made across all the variables of the same name within each clause. Thus, if some variables (needlessly) share names with

Anyone passing his history exams and winning the lottery is happy.

$\forall$ X (pass (X,history) $\wedge$ win (X,lottery) $\rightarrow$ happy (X))

Anyone who studies or is lucky can pass all his exams.

$\forall$ X $\forall$ Y (study (X) $\vee$ lucky (X) $\rightarrow$ pass (X,Y))

John did not study but he is lucky.

$\neg$ study (john) $\wedge$ lucky (john)

Anyone who is lucky wins the lottery.

$\forall$ X (lucky (X) $\rightarrow$ win (X,lottery))

These four predicate statements are now changed to clause form (Section 12.2.2):

1. $\neg$ pass (X, history) $\vee$ $\neg$ win (X, lottery) $\vee$ happy (X)
2. $\neg$ study (Y) $\vee$ pass (Y, Z)
3. $\neg$ lucky (W) $\vee$ pass (W, V)
4. $\neg$ study (john)
5. lucky (john)
6. $\neg$ lucky (U) $\vee$ win (U, lottery)

Into these clauses is entered, in clause form, the negation of the conclusion:

7. $\neg$ happy (john)

The resolution refutation graph of Figure 12.5 shows a derivation of the contradiction and, consequently, proves that John is happy.

As a final example for this subsection, suppose:

All people who are not poor and are smart are happy. Those people who read are not stupid. John can read and is wealthy. Happy people have exciting lives. Can anyone be found with an exciting life?

We assume $\forall$X (smart (X) $\equiv$ $\neg$ stupid (X)) and $\forall$Y (wealthy (Y) $\equiv$ $\neg$ poor (Y)), and get:

$\forall$X ($\neg$ poor (X) $\wedge$ smart (X) $\rightarrow$ happy (X))

$\forall$Y (read (Y) $\rightarrow$ smart (Y))

read (john) $\wedge$ $\neg$ poor (john)

$\forall$Z (happy (Z) $\rightarrow$ exciting (Z))

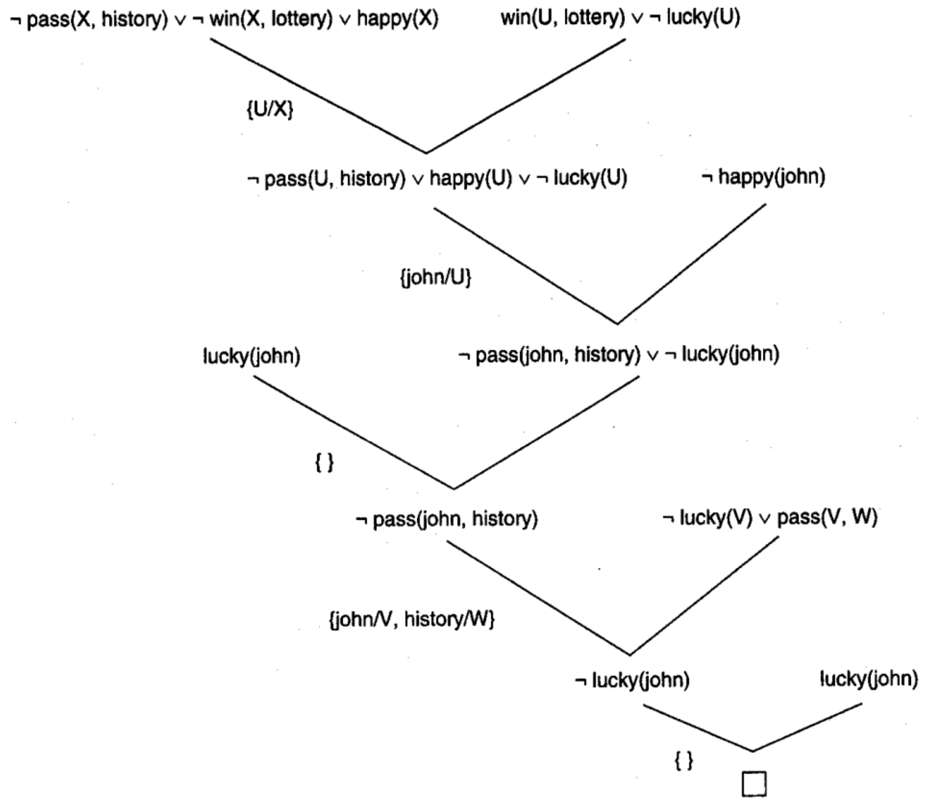

¬ pass(X, history) ∨ ¬ win(X, lottery) ∨ happy(X)
win(U, lottery) ∨ ¬ lucky(U)
{U/X}
¬ pass(U, history) ∨ happy(U) ∨ ¬ lucky(U)
¬ happy(john)
{john/U}
lucky(john)
¬ pass(john, history) ∨ ¬ lucky(john)
{ }
¬ pass(john, history)
¬ lucky(V) ∨ pass(V, W)
{john/V, history/W}
¬ lucky(john)
lucky(john)
{ }
□

Figure 12.5 One resolution refutation for the "happy student" problem.

The negation of the conclusion is:

$\neg \exists$ W (exciting (W))

These predicate calculus expressions for the happy life problem are transformed into the following clauses:

poor (X) $\vee \neg$ smart (X) $\vee$ happy (X)
$\neg$ read (Y) $\vee$ smart (Y)
read (john)
$\neg$ poor (john)
$\neg$ happy (Z) $\vee$ exciting (Z)
$\neg$ exciting (W)

The resolution refutation for this example is found in Figure 12.6.

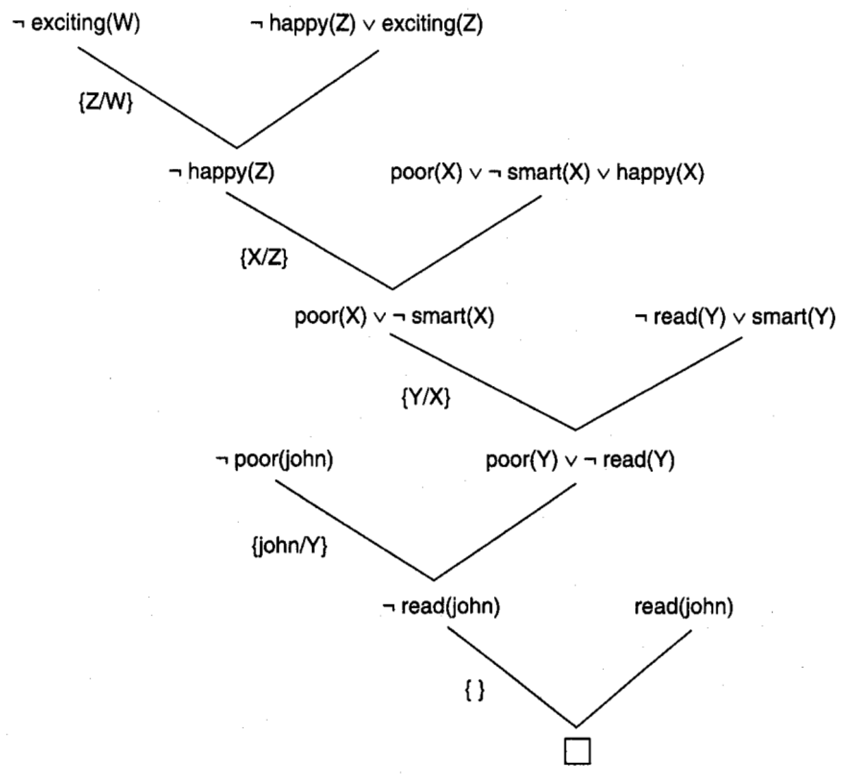


Figure 12.6 Resolution proof for the "exciting life" problem.

12.2.4 Strategies and Simplification Techniques for Resolution

A different proof tree within the search space for the problem of Figure 12.6 appears in Figure 12.7. There are some similarities in these proofs; for example, they both took five resolution steps. Also, the associative application of the unification substitutions found that John was the instance of the person with the exciting life in both proofs.

However, even these two similarities need not have occurred. When the resolution proof system was defined (Section 12.2.3) no order of clause combinations was implied. This is a critical issue: when there are N clauses in the clause space, there are N^2 ways of combining them or checking to see whether they can be combined at just the first level! The resulting set of clauses from this comparison is also large; if even 20% of them produce new clauses, the next round of possible resolutions will contain even more combinations than the first round. In a large problem this exponential growth will quickly get out of hand.

For this reason search heuristics are very important in resolution proof procedures, as they are in all weak method problem solving. As with the heuristics we considered in

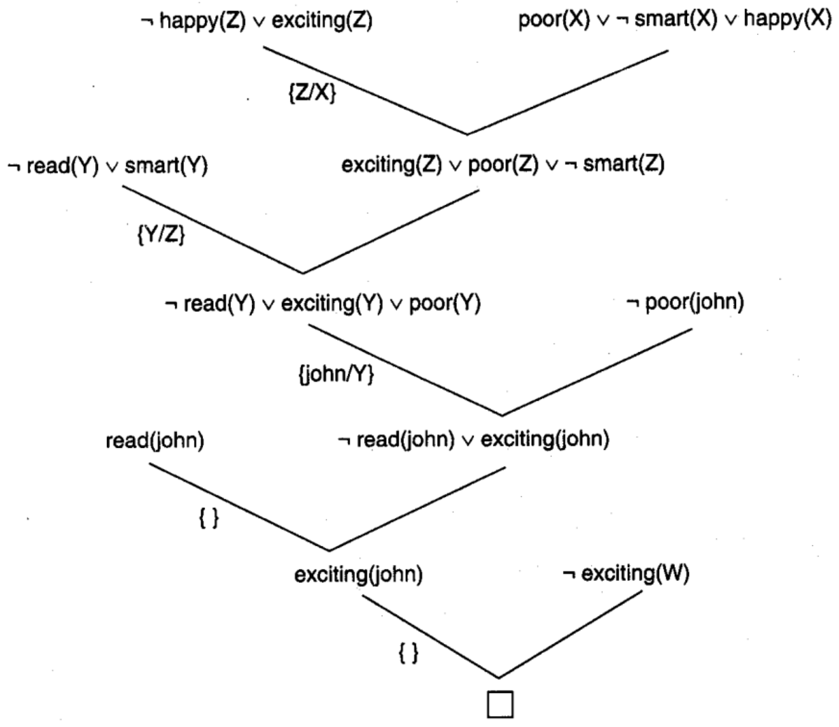

¬ happy(Z) ∨ exciting(Z)
poor(X) ∨ ¬ smart(X) ∨ happy(X)
{Z/X}
¬ read(Y) ∨ smart(Y)
exciting(Z) ∨ poor(Z) ∨ ¬ smart(Z)
{Y/Z}
¬ read(Y) ∨ exciting(Y) ∨ poor(Y)
¬ poor(john)
{john/Y}
read(john)
¬ read(john) ∨ exciting(john)
{ }
exciting(john)
¬ exciting(W)
{ }
□

Figure 12.7 Another resolution refutation for the example of Figure 12.6.

Chapter 4, there is no science that can determine the best strategy for any particular problem. Nonetheless, some general strategies can address the exponential combinatorics.

Before we describe our strategies, we need to make several clarifications. First, based on the definition of unsatisfiability of an expression in Chapter 2, a *set of clauses is unsatisfiable* if no interpretation exists that establishes the set as satisfiable. Second, an inference rule is *refutation complete* if, given an unsatisfiable set of clauses, the unsatisfiability can be established by use of this inference rule alone. Resolution with factoring has this property (Chang and Lee 1973). Finally, a *strategy is complete* if by its use with a refutation-complete inference rule we can guarantee finding a refutation whenever a set of clauses is unsatisfiable. *Breadth-first* is an example of a complete strategy.

The *Breadth-First* Strategy The complexity analysis of exhaustive clause comparison just described was based on breadth-first search. Each clause in the clause space is compared for resolution with every clause in the clause space on the first round. The clauses at the second level of the search space are generated by resolving the clauses produced at the first level with all the original clauses. We generate the clauses at the nth level by resolving all clauses at level $n - 1$ against the elements of the original clause set and all clauses previously produced.

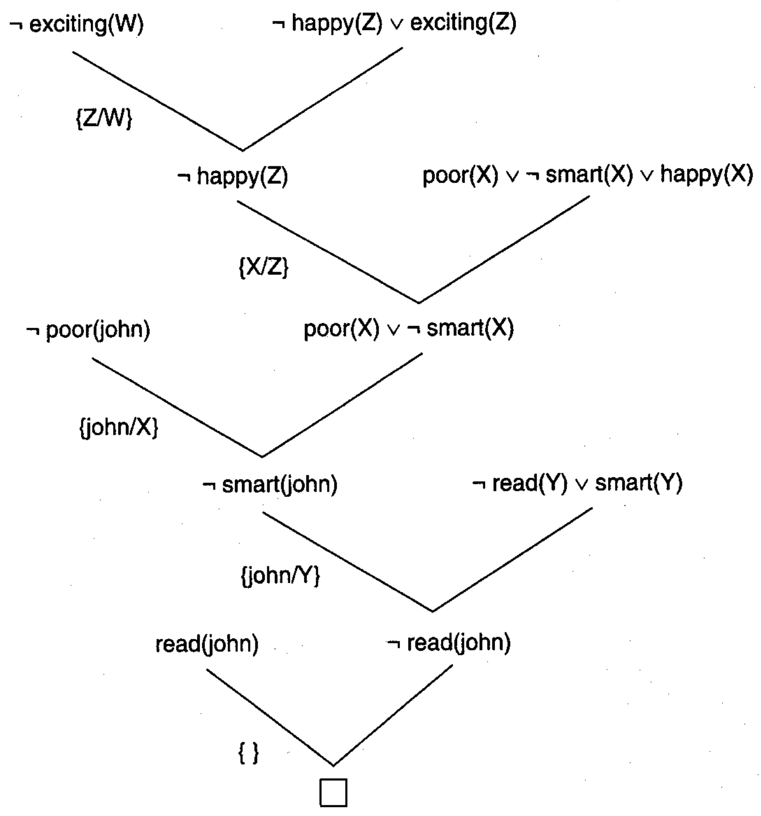


Figure 12.9 Using the unit preference strategy on the "exciting life" problem.

we produce a resultant clause that has fewer literals than the clauses that are resolved to create it, we are closer to producing the clause of no literals. In particular, resolving with a clause of one literal, called a *unit* clause, will guarantee that the resolvent is smaller than the largest parent clause. The unit preference strategy uses units for resolving whenever they are available. Figure 12.9 uses the unit preference strategy on the exciting life problem. The unit preference strategy along with the set of support can produce a more efficient complete strategy.

Unit resolution is a related strategy that requires that one of the resolvents always be a unit clause. This is a stronger requirement than the unit preference strategy. We can show that unit resolution is not complete using the same example that shows the incompleteness of linear input form.

The *Linear Input Form* Strategy The linear input form strategy is a direct use of the negated goal and the original axioms: take the negated goal and resolve it with one of the axioms to get a new clause. This result is then resolved with one of the axioms to get another new clause, which is again resolved with one of the axioms. This process continues until the empty clause is produced.

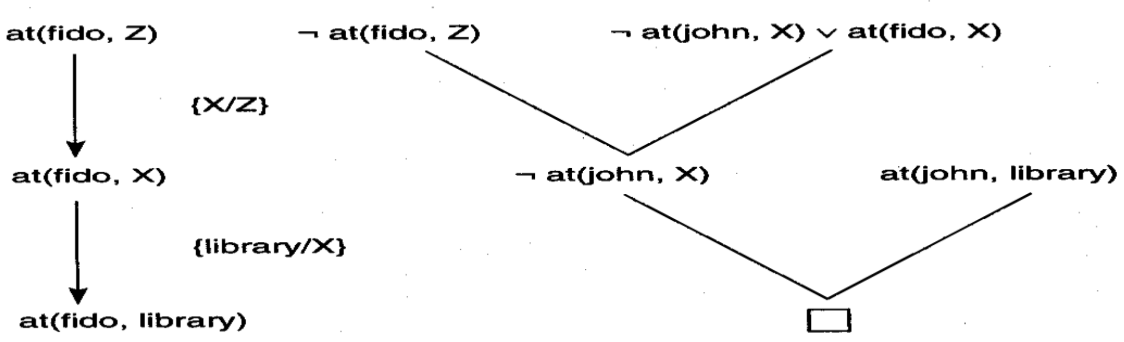


Figure 12.11 Answer extraction process on the "finding fido" problem.

Everyone has a parent. The parent of a parent is a grandparent. Given the person John, prove that John has a grandparent.

The following sentences represent the facts and relationships in the situation above. First, Everyone has a parent:

$$(\forall X)(\exists Y)\ p(X,Y)$$

A parent of a parent is a grandparent.

$$(\forall X)(\forall Y)(\forall Z)\ p(X,Y) \wedge p(Y,Z) \rightarrow gp(X,Z)$$

The goal is to find a W such that gp(john,W) or $\exists$ (W)(gp(john,W)). The negation of the goal is $\neg \exists$ (W)(gp(john,W)) or:

$$\neg\ gp(john,W)$$

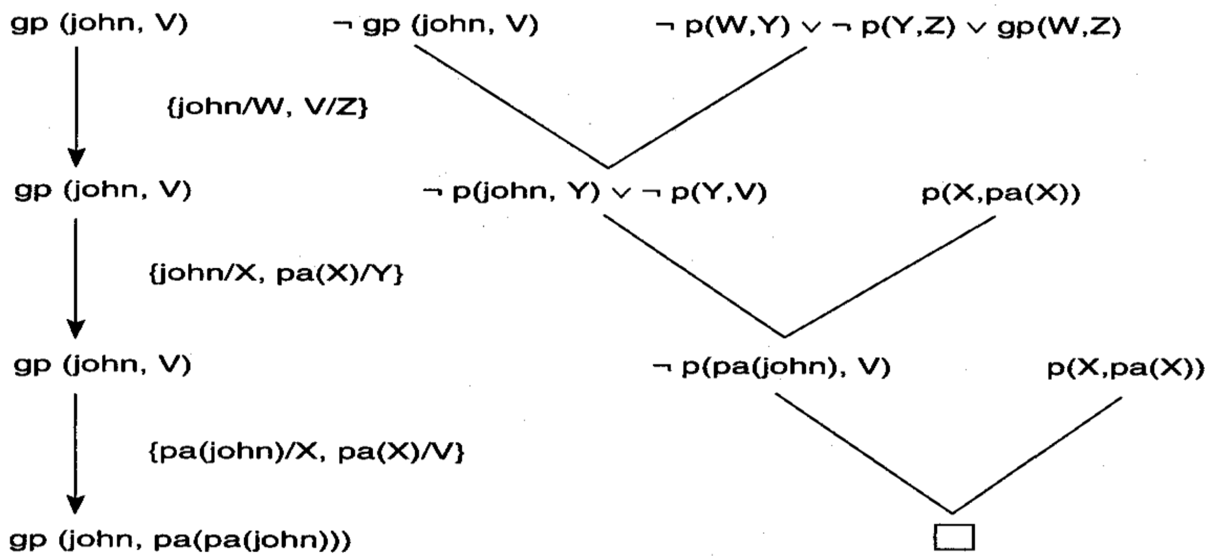


Figure 12.12 Skolemization as part of the answer extraction process.

In the process of putting the predicates above in clause form for the resolution refutation, the existential quantifier in the first predicate (everyone has a parent) requires a skolem function. This skolem function would be the obvious function: take the given X and find the parent of X. Let's call this the pa(X) for "find a parental ancestor for X ." For John this would be either his father or his mother. The clause form for the predicates of this problem is:

$p(X,pa(X))$

$\neg p(W,Y) \vee \neg p(Y,Z) \vee gp(W,Z)$

$\neg gp(john,V)$

The resolution refutation and answer extraction process for this problem are presented in Figure 12.12. Note that the unification substitutions in the answer are

$gp(john,pa(pa(john)))$

The answer to the question of whether John has a grandparent is to "find the parental ancestor of John's parental ancestor." The skolemized function allows us to compute this result.

The general process for answer extraction just described may be used in all resolution refutations, whether they be with the general unifications as in Figures 12.10 and 12.11 or from evaluating the skolem function as in Figure 12.12. The process will yield an answer. The method is really quite simple: the instances (unifications) under which the contradic-