3.1 RESOLUTION

Conjunctive normal form for first-order logic

As in the propositional calculus, first-order resolution requires that sentences be in **conjunctive normal form** (CNF), that is, a conjunction of clauses, where each clause is a disjunction of literals. Literals can contain variables, which are assumed to be universally quantified. For example, the sentence

$$\forall x\ American(x) \wedge Weapon(y) \wedge Sells(x, y, z) \wedge Hostile(z) \Rightarrow Criminal\ (x)$$

becomes, in CNF,

$$\neg \mathbf{American(x)} \vee \neg \mathbf{Weapon(y)} \vee \neg \mathbf{Sells(x, y, z)} \vee \neg \mathbf{Hostile(z)} \vee \mathbf{Criminal\ (x)}\ .$$

Every sentence of first-order logic can be converted into an inferentially equivalent CNF sentence.

The steps are as follows:

1. **Eliminate implications**:

$$\forall x\ [\neg \forall\ y\ \neg Animal(y) \vee Loves(x, y)] \vee [\exists\ y\ Loves(y, x)]\ .$$

2. **Move ¬ inwards**: We have

¬∀x p becomes ∃ x ¬p

¬∃x p becomes ∀ x ¬p . Our sentence goes through the following transformations:

$$\forall x\ [\exists\ y\ \neg(\neg Animal(y) \vee Loves(x, y))] \vee [\exists\ y\ Loves(y, x)]\ .$$

3. **Standardize variables**: For sentences like (∃xP(x))∨(∃xQ(x)) which use the same variable name twice, change the name of one of the variables. This avoids confusion laterwhen we drop the quantifiers. Thus, we have

$$\forall x\ [\exists\ y\ Animal\ (y) \wedge \neg Loves(x, y)] \vee [\exists\ z\ Loves(z, x)]\ .$$

4. **Skolemize**: **Skolemization** is the process of removing existential quantifiers by elimination. The Skolem entities depend on x and z: Here F and G are **Skolem functions**.

$$\forall x\ [Animal\ (F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(z), x)\ .$$

5. **Drop universal quantifiers**: All remaining variables must be universally quantified.

$$[Animal\ (F(x)) \wedge \neg Loves(x, F(x))] \vee Loves(G(z), x)$$

6. **Distribute ∨ over ∧**: This step may also require flattening out nested conjunctions and disjunctions.

$$[Animal(F(x)) \lor Loves(G(z), x)] \land [\neg Loves(x, F(x)) \lor Loves(G(z), x)] .$$

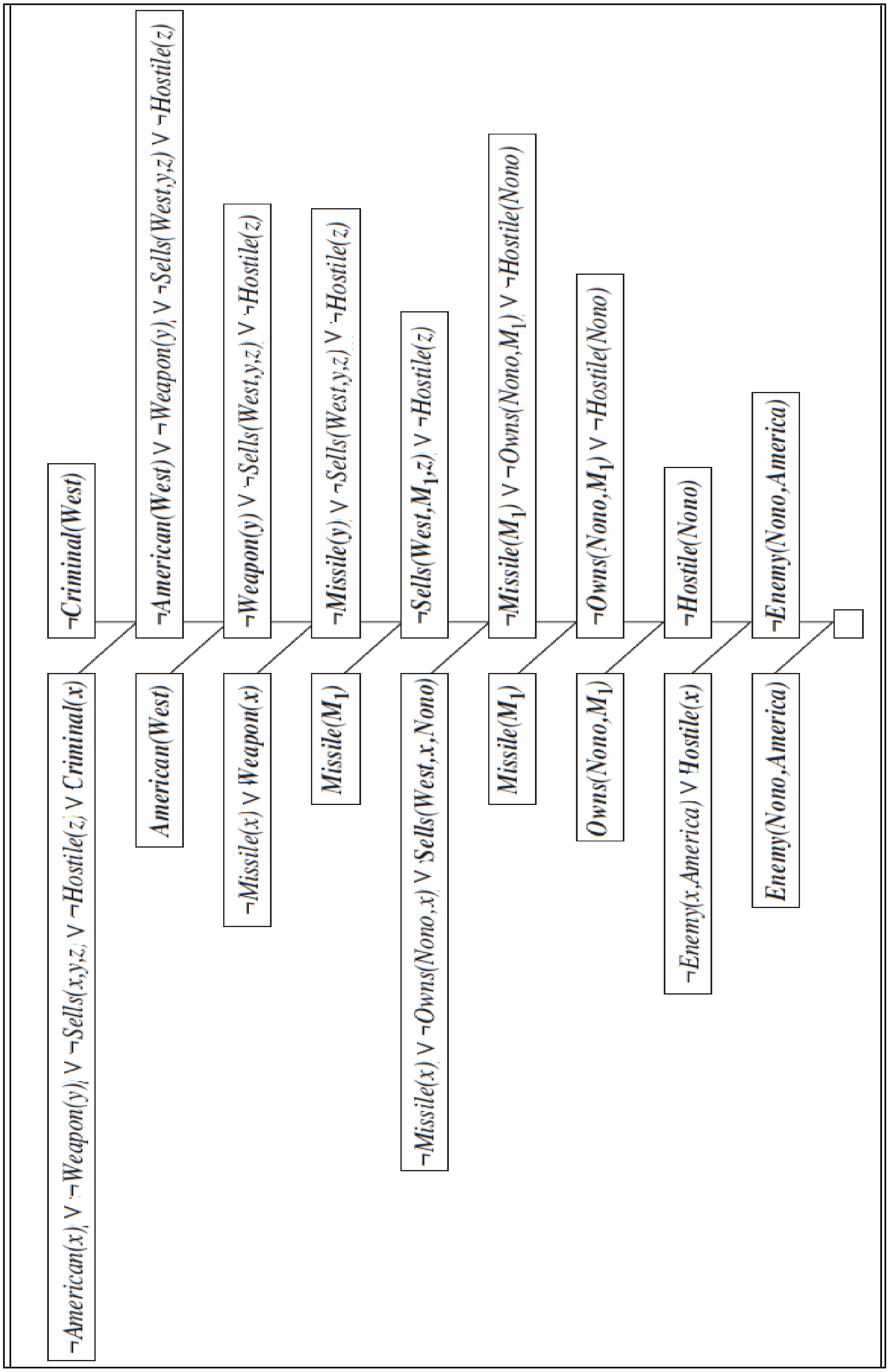

Figure 9.11 : Resolution Proof for Crime Example

The resolution inference rule

For example, we can resolve the two clauses

[Animal (F(x)) ∨ *Loves(G(x), x)] and [*¬*Loves(u, v)* ∨ ¬*Kills(u, v)]*

by eliminating the complementary literals Loves(G(x), x) and ¬Loves(u, v), with unifier θ={u/G(x), v/x}, to produce the **resolvent** clause

[Animal (F(x)) ∨ ¬*Kills(G(x), x)]* .

This rule is called the **binary resolution** rule because it resolves exactly two literals. An alternative approach is to extend **factoring,** the removal of redundant literals.

Example proofs

Resolution proves that KB |= α by proving KB ∧ ¬α unsatisfiable, that is, by deriving the empty clause. We give two example proofs. In the crime example sentences in CNF are

Crime Example :

The first is the crime example from Section 9.3. The sentences in CNF are

¬*American(x)* ∨ ¬*Weapon(y)* ∨ ¬*Sells(x, y, z)* ∨ ¬*Hostile(z)* ∨ *Criminal (x)*

¬*Missile(x)* ∨ ¬*Owns(Nono, x)* ∨ *Sells(West, x, Nono)*

¬*Enemy(x,America)* ∨ *Hostile(x)*

¬*Missile(x)* ∨*Weapon(x)*

Owns(Nono,M1) Missile(M1)

American(West) Enemy(Nono,America) .

We also include the negated goal ¬*Criminal (West).* The resolution proof is shown in Figure 9.11. Notice the structure: single ―spine‖ beginning with the goal clause, resolving against clauses from the knowledge base until the empty clause is generated.

Love Animal Example :

Our second example makes use of Skolemization and involves clauses that are notdefinite clauses. The result is a more complex proof structure. In English, the problem is:

Everyone who loves all animals is loved by someone.

Anyone who kills an animal is loved by no one.

Jack loves all animals.

Either Jack or Curiosity killed the cat, who is named Tuna.

Did Curiosity kill the cat?

First, we express original sentences, background knowledge, and negated goal G in FOL:

A. ∀ x [∀ y Animal (y) ⇒ Loves(x, y)] ⇒ [∃ y Loves(y, x)]

B. ∀ x [∃ z Animal (z) Λ Kills(x, z)] ⇒ [∀ y ¬Loves(y, x)]

C. ∀ x Animal(x) ⇒ Loves(Jack, x)

D. Kills(Jack, Tuna) ∨ Kills(Curiosity, Tuna)

E. Cat(Tuna)

F. ∀ x Cat(x) ⇒ Animal (x)

¬G. ¬Kills(Curiosity, Tuna)

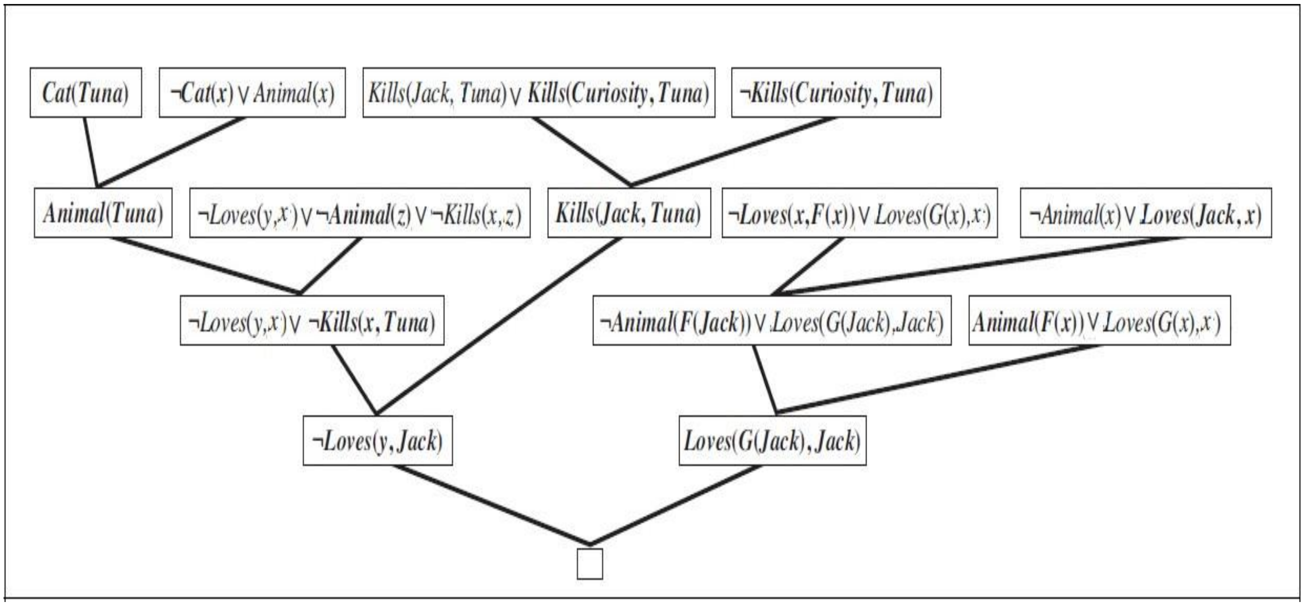

Figure 9.12 : Resolution Proof for Love Animal Example

Now we apply the conversion procedure to convert each sentence to CNF:

A1. Animal(F(x)) ∨ Loves(G(x), x)

A2. ¬Loves(x, F(x)) ∨ Loves(G(x), x)

B. ¬Loves(y, x) ∨ ¬Animal (z) ∨ ¬Kills(x, z)

C. ¬Animal(x) ∨ Loves(Jack, x)

D. Kills(Jack, Tuna) ∨ Kills(Curiosity, Tuna)

E. Cat(Tuna)

F. ¬Cat(x) ∨ Animal (x)

¬G. ¬Kills(Curiosity, Tuna)

The resolution proof that Curiosity killed the cat is given in Figure 9.12. In English, the proof could be paraphrased as follows:

Suppose Curiosity did not kill Tuna. We know that either Jack or Curiosity did; thus Jack must have. Now, Tuna is a cat and cats are animals, so Tuna is an animal. Because anyone who kills an animal is loved by no one, we know that no one loves Jack. On the other hand, Jack loves all animals, so someone loves him; so we have a contradiction. Therefore, Curiosity killed the cat

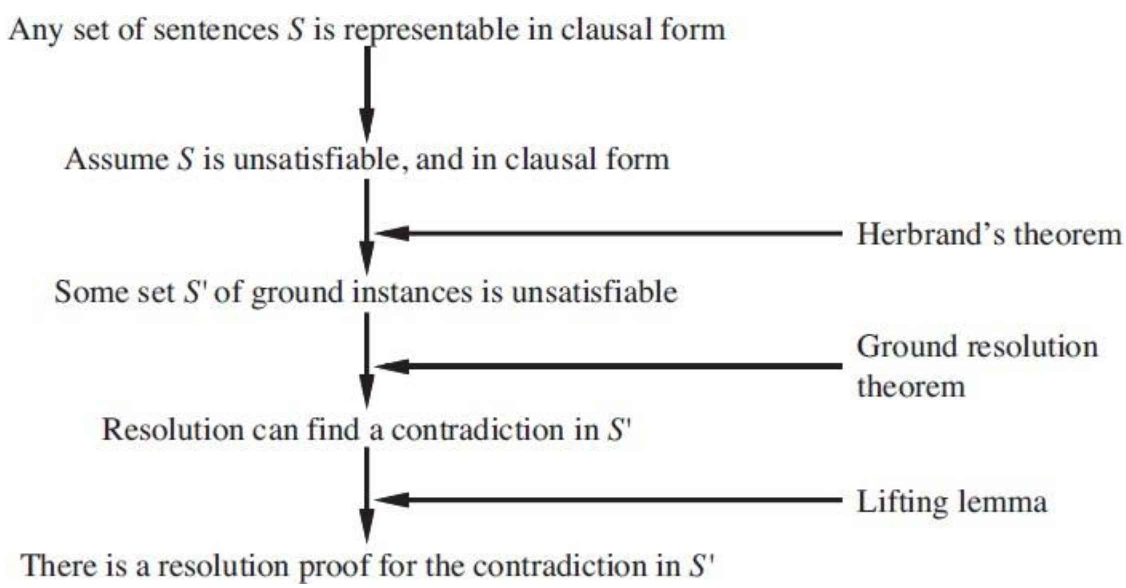

Figure 9.13 Structure of a completeness proof for resolution.

Completeness of resolution

Resolution is **refutation-complete**, which means that *if* a set of sentences is unsatisfiable, then resolution will always be able to derive a contradiction. Hence, it can be used to find all answers to a given question, Q(x), by proving that KB ∧ ¬Q(x) is unsatisfiable.

The basic structure of the proof (Figure 9.13) is as follows:

1. First, we observe that if S is unsatisfiable, then there exists a particular set of *ground instances* of the clauses of S such that this set is also unsatisfiable.

2. We then appeal to the **ground resolution theorem**, which states that propositional resolution is complete for ground sentences.

3. We then use a **lifting lemma** to show that, for any propositional resolution proof using the set of ground sentences, there is a corresponding first-order resolution proof using the first-order

sentences from which the ground sentences were obtained.

Resolution strategies

Unit preference: This strategy prefers to do resolutions where one of the sentences is a single Literal, also known as a **unit clause**.

Set of support: Every resolution step involve at least one element of a *set of support*. The resolvent is then added into the set of support.

Input resolution: Every resolution combines one input sentences with some other sentence.

Subsumption: Eliminates all sentences that are subsumed by an existing sentence in the KB.

3.2 KNOWLEDGE REPRESENTATION

The steps in Knowledge representation are:

1. Identify the task.
2. Assemble the relevant knowledge.
3. Decide on a vocabulary of predicates, functions, and constants.
4. Encode general knowledge about the domain.
5. Encode a description of the specific problem instance.
6. Pose queries to the inference procedure and get answers.
7. Debug the knowledge base.

3.7.1 Characteristics of Knowledge Representation

Representational Adequacy – Ability to represent all knowledge that are needed in the domain.

Inferential Adequacy – The ability to manipulate the structure in such a way to derive new structure corresponding to new knowledge inferred from old.

Inferential Efficiency – The ability to incorporate additional information to focus the mechanism of inference mechanism in the most promising directions.

Acquisitional Efficiency – The ability to acquire new information easily.

Multiple techniques for knowledge representation are

1. Simple Relation Knowledge – declarative facts in the form of table.
2. Inheritable Knowledge – elements of specific classes inherit attributes and values from more general classes in which they are included.
3. Inferential Knowledge – implements standard logical rules of inference with resolution.
4. Procedural Knowledge – specifies what to do and when.

Components of Good Representation

For analysis purposes it is useful to be able to break any knowledge representation downinto four fundamental components:

- Lexical Part – Determines symbols or words used
- Strutural or syntactic part – constraints on how the words or symbols used
- Semantic Part – association of real world meaning with representation.
- Procedural Part – Access procedures that generates and compute things with the representation.Knowledge can be represented in different forms, as mental images in one‟s thoughts, as spoken or written words in some language, as graphical or other pictures, and as character strings or collections of magnetic spots stored in a computer.

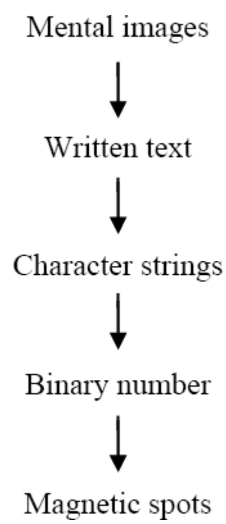

Figure: Different levels of knowledge Representation