جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم

قسم الامن السيبراني

Lecture: (4)

Knowledge Representation
(Resolution and Unification)

Subject: Artificial Intelligence Principles
Level: Third
Lecturer: Prof. Dr. Mehdi Ebady Manaa

Knowledge Representation (Resolution and Unification)

1. Introduction to Conjunctive Normal Form (CNF) and Resolution in Propositional Logic

In propositional logic, one of the fundamental objectives is to represent logical knowledge in a structured and standardized form that enables systematic reasoning, automated inference, and computational analysis. Among the most important standardized representations is the **Conjunctive Normal Form (CNF)**, which plays a central role in artificial intelligence, cybersecurity, automated theorem proving, and Satisfiability (SAT) solving.

Conjunctive Normal Form refers to a logical expression structured as a **conjunction (AND, denoted ∧)** of one or more clauses, where each clause is a disjunction (OR, denoted ∨) of literals. A literal is defined as either a propositional variable (such as ***p, q,*** or ***r)*** or its negation (***such as ¬p, ¬p, or ¬r***). The general structure of CNF can be expressed as:

(C1) ∧ (C2) ∧ (C3)∧ ⋯ ∧(Cn) تعتبر بنود

where each clause takes the form:

(L1 ∨ L2 ∨ L3 ∨…) بداخل كل بند فصل

This representation is not merely a syntactic convention but serves as the foundation for efficient logical inference mechanisms. In particular, CNF enables the application of the ***Resolution rule****, which is a sound and complete inference technique used to derive new logical conclusions from existing clauses.*

The **Resolution rule** operates by eliminating a complementary pair of literals—one positive and one negated—from two clauses. For example, given the clauses:

(p ∨ r)

(q ∨ ¬r)

we can infer the new clause:

(p ∨ q)

This inference is valid because the truth of the original clauses guarantees that at least one of the literals p or q must be true, regardless of the truth value of r. The derived clause is called the **resolvent**, and this process of elimination allows logical systems to progressively simplify and analyze complex logical expressions.

The importance of CNF and Resolution extends beyond theoretical logic. These concepts form the computational backbone of modern SAT solvers, ***which are widely used in software verification, network security analysis, vulnerability detection, and artificial intelligence planning systems.*** In cybersecurity, for example, logical reasoning based on CNF is applied in intrusion detection systems, policy verification, and automated vulnerability analysis. Similarly, in computer science, CNF enables efficient implementation of decision procedures and automated reasoning engines.

The **SAT problem** is the problem of determining whether there exists an assignment of truth values (True or False) to variables that makes a logical formula true.

Simple example 1

Formula:

(p ∨ q) ∧ (¬p ∨ r)

Question: Is this formula **satisfiable**?

The **SAT problem** is the problem of determining whether there exists an assignment of truth values ***(True or False)*** to variables that makes a logical formula true.

Can we assign True/False values to variables so that the whole logical expression becomes True?

- p=False
- q=True
- r=True

Check:

$$(p \vee q) = False \vee True = True$$

$$(\neg p \vee r) = True \vee True = True$$

Overall:

$$True \wedge True = True$$

Example 2:

$$(p) \wedge (\neg p)$$

This can never be True because:

- If p=True, then ¬p=False
- If p=False , then ¬p= True

So this is **UNSATISFIABLE (UNSAT)**.

Why SAT is important in Computer Science

SAT is used in:

- Artificial Intelligence
- Cybersecurity analysis

- Network verification
- Software verification
- Vulnerability detection
- Automated theorem proving
- Cryptography analysis

2. Resolution

It is an algorithm for proving facts true or false by virtue of contradiction. If we want to prove a theorem X is true, we have to show that the negation of X is not true.

Resolution theorem:

For any three clauses p, q and r,

$$p \vee r,\ q \vee \neg r \Rightarrow p \vee q$$

First: What is Modus Ponens?

Modus Ponens is one of the most fundamental and sound inference rules in propositional logic.

General form:

$$P \rightarrow Q$$

$$P$$

$$\therefore Q$$

Meaning in words:

- If **P implies Q**
- And **P is true**
- Then **Q must be true**

This rule allows us to derive a conclusion from a conditional statement and its premise.

Example 4:

Statement 1: If it rains, the ground becomes wet

$$Rain \rightarrow Wet$$

Statement 2: It rains

$$Rain$$

Conclusion (using Modus Ponens):

$$Wet$$

Example

Prove that

"Fido will die" from the statements:

"Fido is a dog",

"all dogs are animals"

"all animals will die", with applying **modus ponens**

1- All dogs are animals: $\forall X (dog(X) \rightarrow animal(X))$

2-Fido is a dog : dog (fido)

3-Modus Ponens and {fido / X} gives : animal(fido)

4- All animals will die : $\forall Y (animal(Y) \rightarrow die(Y))$

5- Modus Ponens and {fido/Y} gives : die (fido).

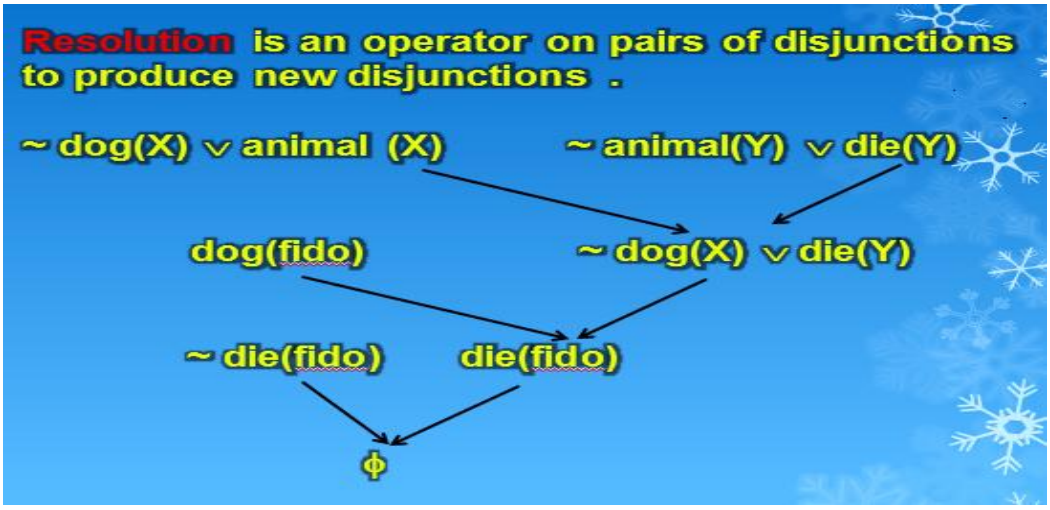
Clause form :

$\sim$ dog(X) $\vee$ animal (X)

dog(fido)

$\sim$ animal(Y) $\vee$ die(Y)

$\sim$ die(fido)



3. Unification

Unification is a technique for taking two sentences in predicate logic and finding a substitution that makes them look the same.

- A variable can be replaced by a constant.
- A variable can be replaced by another variable.
- A variable can be replaced with a predicate, as long as the predicate does not contain that variable.

Unification Conditions :

1- The two predicate names must be the same.

2-The two predicates must have same no. of arguments.

Example:

Given the following set of predicates,

1.hates(X , Y)

2.hates(John, Football).

3. hates(Adam, Spinach).

Unify 1 and 2: S={john/X, footbay/Y}

Unify 1 and 3: S={adam/X, spanich/Y}

If we introduce more complex unifications:

4. Like(X, season(Y))

5. Like(George, season(summer)).

6. Like(Z, summer)

Unify 4 and 6: S={Z/X, summer/Y}

Unify 4 and 5: S={george/X, summer/Y}

3-Frames

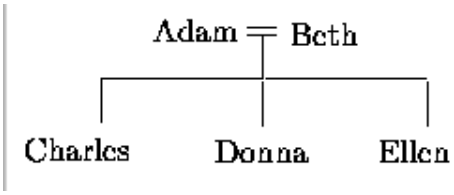
A frame is a collection of attributes which defines the state of an object and its relationship to other frames (objects).

Frames are called Slot-and-Filler data representations.

Slots are the data values,

Fillers are attached procedures.

Frames are often linked into a hierarchy to represent has-part and isa relationships.



Adam:

sex: Male

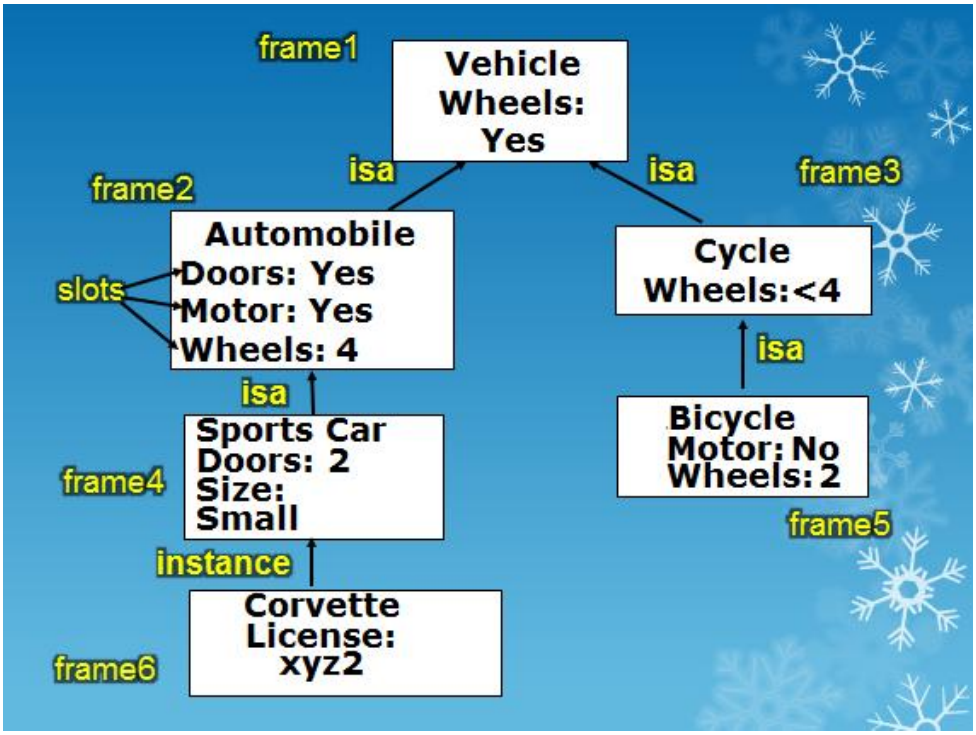
spouse: Beth

child: (Charles Donna Ellen)

A frame has two different types of name:

1-True name (tname), (frame-27)

2-Number of public names (pnames): Public names are stored as values in the name slot of the frame.



From the above figure:

- Each frame has a set of slots.
- The automobile frame has three slots.
- The automobile has doors, a motor, and four wheels, It is a subset of vehicles.
- A sports car is a subset or type of automobile. It has two doors and is small.
- Corvette is an instance of a sports car, and each instance has a unique license number.

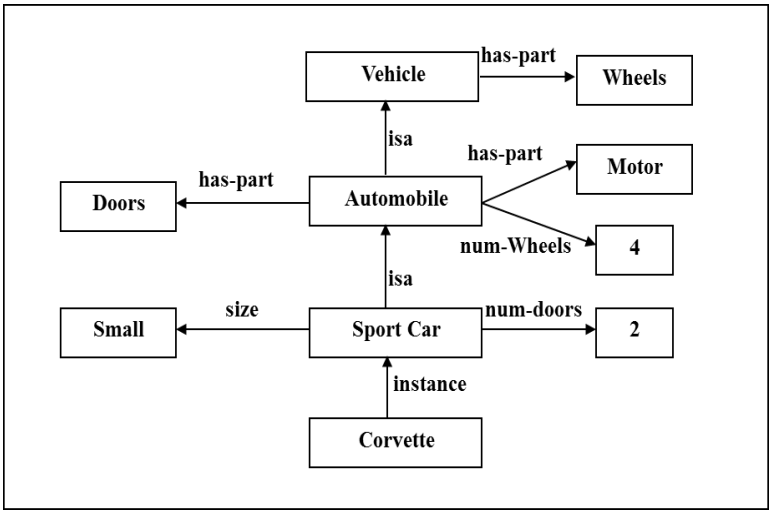
Frame problem

The problem is that if we copy the *complete frame* for each step in the sequence, then we may quickly use up the *computer memory*, as we duplicate the same unchanged knowledge over and over. and it is not always obvious which attributes in the frame should change.

Solution is to specify which parts must match for a condition to be true, and to only change those slots or attributes.

4-Semantic Nets

Semantic nets are used to define the meaning of a concept by its relationships to other concepts. A graph data structure is used, with nodes used to hold concepts, and links with natural language labels used to show the relationships. Frames and semantic nets are very closely related to predicate logic .



5-Probabilities

a- Unconditional probabilities:

Represent the chance that something will happen.

Example: we can look in a weather almanac and see that, on average, it has rained 10 days in March in London over the last hundred years. So the probability that it will rain on any given day in March is roughly 33 percent.

b- Conditional probability:

Expressed as P(H | E), that is read as the probability of hypothesis H given that we have observed evidence E.

Example: suppose we know that a big storm is blowing in from South Babil, and it will reach Baghdad tomorrow. Given that knowledge, we may say there is an 80 percent chance of rain.