Introduction to AI

Lecture 16
Unification and Resolution in First-Order Logic

Dr. Tamal Ghosh
Department of CSE
Adamas University

What is Unification?

- Unification is a process of making two different logical atomic expressions identical by finding a substitution. Unification depends on the substitution process.
- It takes two literals as input and makes them identical using substitution.
- Let Ψ_1 and Ψ_2 be two atomic sentences and σ be a unifier such that, $\Psi_1\sigma = \Psi_2\sigma$, then it can be expressed as **UNIFY(Ψ_1, Ψ_2)**.
- **Example: Find the Most General Unifier for Unify{fruit(x), fruit(mango)}**

Let **Ψ_1 = fruit(x), Ψ_2 = fruit(mango),**

Substitution θ = {mango/x} is a unifier for these atoms and by applying this substitution, both expressions will be identical.

- The UNIFY algorithm is used for unification, which takes two atomic sentences and returns a unifier for those sentences (If any exist).
- Unification is a key component of all first-order inference algorithms.
- It returns fail if the expressions do not match with each other.
- The substitution variables are called MGU.

Unification…

E.g. Let's say there are two different expressions, **P(x, y), and P(a, f(z))**.
In this example, we need to make both above statements identical to each other. For this, we will perform the substitution.

P(x, y)..... (i)
P(a, f(z))..... (ii)

Substitute x with a, and y with f(z) in the first expression, and it will be represented as a/x and f(z)/y.
With both substitutions, the first expression will be identical to the second expression and the substitution set will be: **[a/x, f(z)/y]**.

Conditions for Unification

The following are some basic conditions for unification:

- **Predicate symbols** must be the same, atoms or expressions with different predicate symbols can never be unified. {P(x, y); Q(a, b)} …. ×
- **The number of Arguments** in both expressions must be identical. {P(x, y, z); P(a, b)} …. ×
- Unification will **fail if there are two similar variables** present in the same expression. {P(x, x); P(a, b)} …. × but {P(f(x), f(x)); P(a, b)} …. ✓
- Unification will **fail if Ψ1 is a variable and occurs in Ψ2** (vice versa). {P(x, y); P(a, f(y))} …. ×

Unification Algorithm

Algorithm: Unify(Ψ_1, Ψ_2):

Step. 1: If Ψ1 or Ψ2 is a variable or constant, then:

a) If Ψ1 or Ψ2 are identical, then return **NIL**.

b) Else if Ψ1 is a variable,

a. then if Ψ1 occurs in Ψ2, then return **FAILURE**

b. Else return { (Ψ2/ Ψ1)}.

c) Else if Ψ2 is a variable,

a. If Ψ2 occurs in Ψ1 then return **FAILURE**

b. Else return {(Ψ1/ Ψ2)}.

d) Else return **FAILURE**.

Step. 2: If the initial **Predicate symbol** in Ψ1 and Ψ2 are not the same, then return **FAILURE**.

Step. 3: IF Ψ1 and Ψ2 have a different number of arguments, then return **FAILURE**.

FOL inference rules for quantifier

Step. 4: Set Substitution set(SUBST) to **NIL**.

Step. 5: For i=1 to the number of elements in Ψ1.

a) Call **Unify function** with the ith element of Ψ1 and ith element of Ψ2, and put the result into S.

b) If S = failure then returns **Failure**

c) If S ≠ NIL then do,

a. **Apply** S to the remainder of both L1 and L2.

b. SUBST= APPEND(S, SUBST).

Step. 6: Return **SUBST**.

Implementation of the Algorithm

Step.1: Initialize the substitution set to be empty.

Step.2: Recursively unify atomic sentences:

1.Check for Identical expression match.

2.If one expression is a variable v_i, and the other is a term t_i which does not contain variable v_i, then:

1. Substitute t_i / v_i in the existing substitutions
2. Add t_i /v_i to the substitution setlist.
3. If both expressions are functions, then the function name must be similar, and the number of arguments must be the same in both expressions.

Example

1. *Find the MGU of {p(f(a), g(Y)) and p(X, X)}*

Sol: S0 => {p(f(a), g(Y)); p(X, X)}, Ψ1 = p(f(a), g(Y)), and Ψ2 = p(X, X)
SUBST θ= {f(a) / X}
S1 => Ψ1 = p(f(a), g(Y)), and Ψ2 = p(f(a), f(a))
SUBST θ= {f(a) / g(y)}, **Unification failed.**

Unification is not possible for these expressions.

2. *Find the MGU of {p(b, X, f(g(Z))) and p(Z, f(Y), f(Y))}*

Here, Ψ1 = p(b, X, f(g(Z))) , and Ψ2 = p(Z, f(Y), f(Y))
S0 => { p(b, X, f(g(Z))); p(Z, f(Y), f(Y))}
SUBST θ={b/Z}

Example…

- S1 => { p(b, X, f(g(b))); p(b, f(Y), f(Y))}
- SUBST θ={f(Y) /X}
- S_2 => { p(b, f(Y), f(g(b))); p(b, f(Y), f(Y))}
  SUBST θ= {g(b) /Y}
  S_2 => { p(b, f(g(b)), f(g(b)); p(b, f(g(b)), f(g(b))} **Unified Successfully.**
  And Unifier = { b/Z, f(Y) /X , g(b) /Y}.

3. *Find the MGU of {p (X, X), and p (Z, f(Z))}*

Here, Ψ_1 = {p (X, X), and Ψ_2 = p (Z, f(Z))
S_0 => {p (X, X); p (Z, f(Z))}
SUBST θ= {X/Z}
S1 => {p (Z, Z), p (Z, f(Z))}
SUBST θ= {f(Z) / Z}, **Unification Failed**. **Hence, unification is not possible for these expressions.**

Example…

4. Find the MGU of UNIFY(prime (11), prime(y))

Here, Ψ1 = {prime(11) , and Ψ2 = prime(y)}

S0 => {prime(11) , prime(y)}

SUBST θ= {11/y}

S1 => {prime(11) , prime(11)} , **Successfully unified. Unifier: {11/y}.**

5. Find the MGU of Q(a, g(x, a), f(y)), Q(a, g(f(b), a), x)}

Here, Ψ_1 = Q(a, g(x, a), f(y)), and Ψ_2 = Q(a, g(f(b), a), x)

S_0 => {Q(a, g(x, a), f(y)); Q(a, g(f(b), a), x)}

SUBST θ= {f(b)/x}

S_1 => {Q(a, g(f(b), a), f(y)); Q(a, g(f(b), a), f(b))}

SUBST θ= {b/y}

S_1 => {Q(a, g(f(b), a), f(b)); Q(a, g(f(b), a), f(b))}, **Successfully Unified. Unifier: [a/a, f(b)/x, b/y].**

Example…

6. UNIFY(knows(Richard, x), knows(Richard, John))

Here, Ψ_1 = knows(Richard, x), and Ψ_2 = knows(Richard, John)

S_0 => { knows(Richard, x); knows(Richard, John)}

SUBST θ= {John/x}

S_1 => { knows(Richard, John); knows(Richard, John)}, **Successfully Unified.**

Unifier: {John/x}.

RESOLUTION

What is Resolution?

- Resolution is a theorem-proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions. It was invented by Mathematician John Alan Robinson in the year 1965.
- Resolution is used if there are various statements are given, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule that can efficiently operate on the **conjunctive normal form or clausal form**.
- **Clause**: Disjunction of literals (an atomic sentence) is called a **clause**. It is also known as a unit clause.
- **Conjunctive Normal Form**: A sentence represented as a conjunction of clauses is said to be **conjunctive normal form** or **CNF**.

The resolution inference rule

- The resolution rule for first-order logic is simply a lifted version of the propositional rule. Resolution can resolve two clauses if they contain complementary literals, which are assumed to be standardized apart so that they share no variables.

$$\frac{l_1 \vee \ldots\ldots \vee l_{k,} \qquad m_1 \vee \ldots\ldots \vee m_n}{\text{SUBST}(\theta, l_1 \vee \ldots\ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots\ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots\ldots \vee m_n)}$$

- Where $\mathbf{l_i}$ and $\mathbf{m_j}$ are complementary literals.
- This rule is also called the **binary resolution rule** because it only resolves exactly two literals.
- **Example:**
- We can resolve two clauses which are given below:
- **[Animal (g(x) V Loves (f(x), x)] and [¬ Loves(a, b) V ¬Kills(a, b)]**
- Where two complimentary literals are: **Loves (f(x), x) and ¬ Loves (a, b)**
- These literals can be unified with unifier **θ= [a/f(x), and b/x]** , and it will generate a resolvent clause: **[Animal (g(x) V ¬ Kills(f(x), x)].**

Steps for Resolution

1. Conversion of facts into first-order logic.
2. Convert FOL statements into CNF
3. Negate the statement that needs to be proved (proof by contradiction)
4. Draw resolution graph (unification).

- To better understand all the above steps, we will take an example in which we will apply resolution.

Example

1. John likes all kinds of food.
2. Apples and vegetables are food
3. Anything anyone eats and is not killed is food.
4. Anil eats peanuts and is still alive
5. Harry eats everything that Anil eats.

Prove by resolution that:

1. John likes peanuts.

Step-1: Conversion of Facts into FOL

- In the first step we will convert all the given statements into its first order logic.

a. $\forall x$: food(x) $\rightarrow$ likes(John, x)

b. food(Apple) $\wedge$ food(vegetables)

c. $\forall x \; \forall y$: eats(x, y) $\wedge \neg$ killed(x) $\rightarrow$ food(y)

d. eats (Anil, Peanuts) $\wedge$ alive(Anil).

e. $\forall x$: eats(Anil, x) $\rightarrow$ eats(Harry, x)

f. $\forall x$: $\neg$ killed(x) $\rightarrow$ alive(x) } **added predicates.**

g. $\forall x$: alive(x) $\rightarrow \neg$ killed(x) }

h. likes(John, Peanuts)

Step-2: Conversion of FOL into CNF

- In First order logic resolution, it is required to convert the FOL into CNF as CNF form makes easier for resolution proofs.
- **Eliminate all implication (→) and rewrite**
  - ∀x ¬ food(x) V likes(John, x)
  - food(Apple) Λ food(vegetables)
  - ∀x ∀y ¬ [eats(x, y) Λ ¬ killed(x)] V food(y)
  - eats (Anil, Peanuts) Λ alive(Anil)
  - ∀x ¬ eats(Anil, x) V eats(Harry, x)
  - ∀x¬ [¬ killed(x)] V alive(x)
  - ∀x ¬ alive(x) V ¬ killed(x)
  - likes(John, Peanuts).

Step-2…

- **Move negation (¬)inwards and rewrite**
  - ∀x ¬ food(x) V likes(John, x)
  - food(Apple) Λ food(vegetables)
  - ∀x ∀y ¬ eats(x, y) V killed(x) V food(y)
  - eats (Anil, Peanuts) Λ alive(Anil)
  - ∀x ¬ eats(Anil, x) V eats(Harry, x)
  - ∀x killed(x) V alive(x)
  - ∀x ¬ alive(x) V ¬ killed(x)
  - likes(John, Peanuts).

Step-2…

- **Rename variables or standardize variables**
  - ∀x ¬ food(x) V likes(John, x)
  - food(Apple) Λ food(vegetables)
  - ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)
  - eats (Anil, Peanuts) Λ alive(Anil)
  - ∀w¬ eats(Anil, w) V eats(Harry, w)
  - ∀g killed(g) V alive(g)
  - ∀k ¬ alive(k) V ¬ killed(k)
  - likes(John, Peanuts).

Step-2…

- **Eliminate existential instantiation quantifier by elimination.**
  In this step, we will eliminate existential quantifier ∃, and this process is known as **Skolemization**. But in this example problem since there is no existential quantifier so all the statements will remain the same in this step.
- **Drop Universal quantifiers.**
  In this step, we will drop all universal quantifiers since all the statements are not implicitly quantified so we don't need it.

Step-2…

- ¬ food(x) V likes(John, x)
- food(Apple)
- food(vegetables)
- ¬ eats(y, z) V killed(y) V food(z)
- eats (Anil, Peanuts)
- alive(Anil)
- ¬ eats(Anil, w) V eats(Harry, w)
- killed(g) V alive(g)
- ¬ alive(k) V ¬ killed(k)
- likes(John, Peanuts).

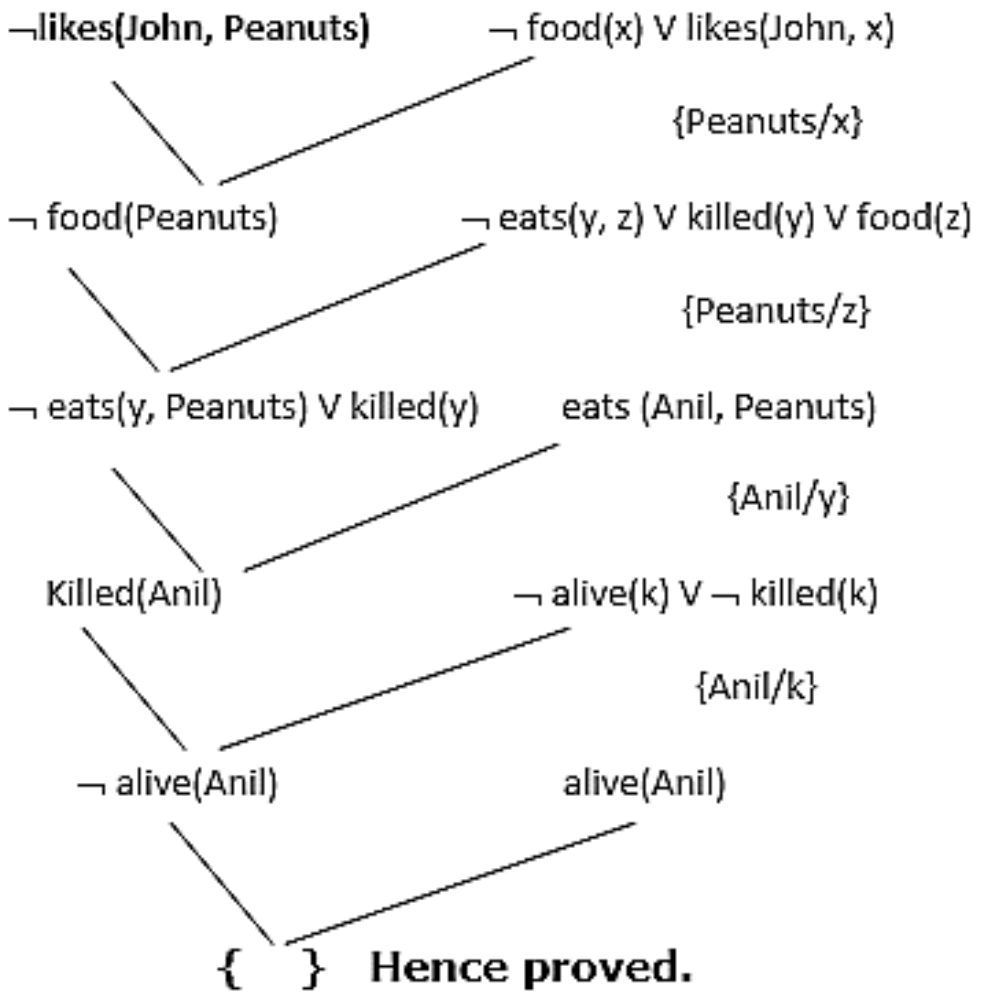
Step-2:

- Note: Statements "food(Apple) Λ food(vegetables)" and "eats (Anil, Peanuts) Λ alive(Anil)" can be written in two separate statements.
- **Distribute conjunction Λ over disjunction ¬.**
  This step will not make any change in this problem.
- **Step-3: Negate the statement to be proved**
- In this statement, we will apply negation to the conclusion statements, which will be written as ¬likes(John, Peanuts)

Step-4: Draw Resolution graph

- Now in this step, we will solve the problem by resolution tree using substitution. For the above problem, it will be given as follows:

- Hence the negation of the conclusion has been proved as a complete contradiction with the given set of statements.



Explanation of Resolution graph

- In the first step of resolution graph, **¬likes(John, Peanuts)** , and **likes(John, x)** get resolved(canceled) by substitution of **{Peanuts/x}**, and we are left with **¬ food(Peanuts)**
- In the second step of the resolution graph, **¬ food(Peanuts)** , and **food(z)** get resolved (canceled) by substitution of **{ Peanuts/z}**, and we are left with **¬ eats(y, Peanuts) V killed(y)** .
- In the third step of the resolution graph, **¬ eats(y, Peanuts)** and **eats (Anil, Peanuts)** get resolved by substitution **{Anil/y}**, and we are left with **Killed(Anil)** .
- In the fourth step of the resolution graph, **Killed(Anil)** and **¬ killed(k)** get resolve by substitution **{Anil/k}**, and we are left with **¬ alive(Anil)** .
- In the last step of the resolution graph **¬ alive(Anil)** and **alive(Anil)** get resolved.