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, $\boldsymbol{\Psi_1}\sigma = \boldsymbol{\Psi_2}\sigma$, then it can be expressed as **UNIFY(Ψ_1, Ψ_2)**.
- **Example: Find the MGU for Unify{King(x), King(John)}**

Let Ψ_1 = King(x), Ψ_2 = King(John),

Substitution θ = {John/x} is a unifier for these atoms and applying this substitution, and 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 Most General Unifier or MGU.

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 the 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:

Following are some basic conditions for unification:

- Predicate symbol must be same, atoms or expression with different predicate symbol can never be unified.
- Number of Arguments in both expressions must be identical.
- Unification will fail if there are two similar variables present in the same expression.

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 Ψ1is 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 same, then return FAILURE.
Step. 3: IF Ψ1 and Ψ2 have a different number of arguments, then return FAILURE.
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:

a. Check for Identical expression match.
b. If one expression is a variable v_i, and the other is a term t_i which does not contain variable v_i, then:
    a. Substitute t_i / v_i in the existing substitutions
    b. Add t_i /v_i to the substitution setlist.
    c. If both the expressions are functions, then function name must be similar, and the number of arguments must be the same in both the expression.

For each pair of the following atomic sentences find the most general unifier (If exist).

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

Sol: S_0 => Here, Ψ_1 = p(f(a), g(Y)), and Ψ_2 = p(X, X)

SUBST θ= {f(a) / X}

S1 => Ψ_1 = p(f(a), g(Y)), and Ψ_2 = 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, Ψ_1 = p(b, X, f(g(Z))) , and Ψ_2 = p(Z, f(Y), f(Y))

S_0 => { p(b, X, f(g(Z))); p(Z, f(Y), f(Y))}

SUBST θ={b/Z}

S_1 => { 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.

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

Here, Ψ_1 = {prime(11) , and Ψ_2 = prime(y)}

S_0 => {prime(11) , prime(y)}

SUBST θ= {11/y}

S_1 => {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].

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}.

← Prev

Next →

For Videos Join Our Youtube Channel: Join Now

Feedback

- Send your Feedback to feedback@javatpoint.com

Help Others, Please Share

Learn Latest Tutorials

Splunk	SPSS	Swagger
Transact-SQL	Tumblr	ReactJS
RegEx	React	R Programming

Regex

Reinforcement Learning

R Programming

RxJS

React Native tutorial

React Native

Python Design Patterns

Python Design Patterns

Python Pillow tutorial

Python Pillow

Python Turtle tutorial

Python Turtle

Keras tutorial

Keras

Preparation

Aptitude

Aptitude

Logical Reasoning

Reasoning

Verbal Ability

Verbal Ability

Interview Questions

Interview Questions

Company Interview Questions

Company Questions

Trending Technologies

Artificial Intelligence

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science Tutorial

Data Science

Angular 7 Tutorial

Angular 7

Blockchain Tutorial

Blockchain

Git Tutorial

Git

Machine Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

B.Tech / MCA

- DBMS tutorial
  DBMS
- Data Structures tutorial
  Data Structures
- DAA tutorial
  DAA
- Operating System
  Operating System
- Computer Network tutorial
  Computer Network
- Compiler Design tutorial
  Compiler Design
- Computer Organization and Architecture
  Computer Organization
- Discrete Mathematics Tutorial
  Discrete Mathematics
- Ethical Hacking
  Ethical Hacking
- Computer Graphics Tutorial
  Computer Graphics
- Software Engineering
  Software Engineering
- html tutorial
  Web Technology
- Cyber Security tutorial
  Cyber Security
- Automata Tutorial
  Automata
- C Language tutorial
  C Programming
- C++ tutorial
  C++
- Java tutorial
  Java
- .Net Framework tutorial
  .Net
- Python tutorial
  Python
- List of Programs
  Programs
- Control Systems tutorial
  Control System
- Data Mining Tutorial
  Data Mining
- Data Warehouse Tutorial
  Data Warehouse