Unification Algorithm in AI

Prof. Dr. A. K. M. Akhtar Hossain
Dept. of CSE, University of Rajshahi

Unification

- Any substitution that makes two or more expressions equal is called **a unifier** for the expressions.
- **In propositional logic**, it is easy to determine that two literals cannot both be true at the same time. **Simply look for L and ~L** .
- **In predicate logic,** this matching process is more complicated, since binding of variables must be considered.

Unification Algorithm

- **For example MAN (john) and ~MAN(john) is a contradiction, while MAN (john) and MAN(Himalayas) is not.**
- **Thus in order to determine contradictions, we need a matching procedure that compares two literals and discovers whether there exist a set of substitutions that makes them identical .**
- **There is a recursive procedure that does this matching . It is called Unification algorithm.**

Unification Algorithm

- **In Unification algorithm each literal is represented as a list, where first element is the name of a predicate and the remaining elements are arguments. The argument may be a single element (atom) or may be another list.**
- For example, **we can have literals as**
- **TRYASSASSINATE(Marcus, Caesar)**
- **TRYASSASSINATE(Marcus, Ruler of Rome)**

Unification Algorithm

- **To unify two literals , first check if their first elements are same. If so proceed. Otherwise they cannot be unified.**
- **For example the literals**
  - **TRYASSASSINATE (Marcus, Caesar)**
  - **HATE (Marcus, Caesar)**

Can not be Unified.

- **The unification algorithm recursively matches pairs of elements, one pair at a time.**

Unification Algorithm

- **The Unification algorithm is listed below as a procedure UNIFY (L1, L2). It returns a list representing the composition of the substitutions that were performed during the match.**
- **An empty list NIL indicates that a match was found without any substitutions. If the list contains a single value FAIL, it indicates that the unification procedure failed.**

Unification Algorithm

- **Algorithm UNIFY (L1, L2)**
- **1. if L1 or L2 are both variable or constants, then:**
  - **(a). if L1 or L2 are identical then return NIL.**
  - **(b). else if L1 is a variable then if L1 occurs in L2 then return {FAIL} else return (L2/L1).**
  - **(c). else if L2 is a variable then if L2 occurs in L1 then return {FAIL} else return (L1/L2).**
  - **(d). else return {FAIL}.**

Unification Algorithm

- 2. If the initial predicate symbols in L1 and L2 are not identical, then return {FAIL}.
- 3. If L1 and L2 have a different number of arguments, then return {FAIL}.
- 4. Set SUBST to NIL.

- 5. For i← 1 to number of arguments in L1:
  - (a). Call Unify with the ith argument of L1 and the ith argument of L2, putting result in S.
  - (b). If S contains FAIL then return {FAIL}.
  - (c). If S is not equal to NIL then:
    - (i). Apply S to the remainder of both L1 and L2.
    - (ii). SUBST:=APPEND(SUBST).
- 6. Return SUBST.

Example#1

- Let's say there are two different expressions,

  P(x, y), and P(a, f(z)).
- we need to make both above statements identical to each other.
- 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

- Predicate symbol must be same, atoms or expression with different predicate symbol can never be unified.

  tryassassinate (Marcus, Caesar)

  hate (Marcus, Caesar)

- Number of Arguments in both expressions must be identical.

  hate(Marcus)

  hate (Marcus, Caesar)

- Unification will fail if there are two similar variables present in the same expression.

- **END TODAY**