SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF INFORMATION TECHNOLOGY

19CSE303 - ARTIFICIAL INTELLIGENCE

III YEAR IV SEM

UNIT II – **LOGICAL REASONING**

TOPIC – Unification & Lift

Inference rules for quantifiers

The rule of **Universal Instantiation** (UI for short) says that we can infer any sentence obtained by substituting a ground term (a term without variables) for the variable. It can be applied multiple times to add new sentences.

Suppose our knowledge base contains the axiom stating that

All greedy kings are evil: $\forall x$ **King(x) ∧ Greedy(x) ⇒ Evil(x)**

- Then it seems quite permissible to infer any of the following sentences:

King(John) ∧ Greedy(John) ⇒ Evil(John)

King(Richard) ∧ Greedy(Richard) ⇒ Evil(Richard)

King(Father (John)) ∧ Greedy(Father (John)) ⇒ Evil(Father (John)) .

Let SUBST(θ,α) denote the result of applying the substitution θ to the sentence α. Then the rule is written

$$\frac{\forall v\ \alpha}{\text{SUBST}(\{v/g\}, \alpha)}$$

Inference rules for quantifiers

In the rule for **Existential Instantiation**, the variable is replaced by a **single** new constant symbol. It can be applied only once to replace the existential sentence.

The formal statement is as follows: for any sentence α, variable v, and constant symbol k that does not appear elsewhere in the knowledge base,

$$\frac{\exists v\ \alpha}{SUBST(\{v/k\}, \alpha)}$$

For example, from the sentence $\exists$ x Crown(x) $\wedge$ OnHead(x, John)

we can infer the sentence Crown(C1) $\wedge$ OnHead(C1, John) as long as C1 does not appear elsewhere in the knowledge base.

The existential sentence says there is some object satisfying a condition and applying the existential instantiation rule just gives a name to that object, this is called as **Skolem constant.**

Generalized Modus Ponens Rule

- $\forall x \; King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
- King(John)
- Greedy(John)
- {x/John} solves the query Evil(x).
- In this case, the substitution θ = {x/John}.
- Suppose that instead of knowing Greedy(John), we know that everyone is greedy

$\forall y \; Greedy(y)$

We have to find a substitution both for the variables in the implication sentence and for the variables in the sentences that are in the knowledge base. In this case, applying the substitution {x/John, y/John} to the implication premises **King(x)** and **Greedy(x)** and the knowledge-base sentences **King(John)** and **Greedy(y)** will make them identical.

Thus, we can infer the conclusion of the implication. This inference process can be captured as a single inference rule that we call **Generalized Modus Ponens.**

Generalized Modus Ponens Rule

For atomic sentences pi, pi' , and q, where there is a substitution θ

such that SUBST(θ, pi') = SUBST(θ, pi), for all i,

$$\frac{p1',\ p2',\ \ldots,\ pn',\ (p1 \wedge p2 \wedge \ldots \wedge pn \Rightarrow q)}{SUBST(\theta, q)}$$

There are n+ 1 premises to this rule: the n atomic sentences pi and the one implication.

The conclusion is the result of applying the substitution θ to the consequent q. For our example:

p1' is King(John) p1 is King(x)

p2' is Greedy(y) p2 is Greedy(x)

θ is {x/John, y/John} q is Evil(x)

SUBST(θ, q) is Evil(John) .

Generalized Modus Ponens is a **lifted version of Modus Ponens**—it raises Modus Ponens from ground (variable-free) propositional logic to first-order logic.

Unification

Lifted inference rules require finding substitutions that make different logical expressions look identical. This process is called **unification** and is a key component of all first-order inference algorithms.

The **UNIFY** algorithm takes two sentences and returns a unifier for them if one exists:

Substitution means replacing one variable with another term. It takes two literals as input and make them identical using substitution. It returns fail if the expressions do not match with each other.

UNIFY(p, q) = θ where SUBST(θ, p) = SUBST(θ, q)

Example: 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 represented as **a/x and f(z)/y.**

With both the substitution the first expression will be identical to the second expression and the substitution set will be **[a/x, f(z)/y]**

Unification

Given : Knows(John, x) is a predicate.

- whom does John know?
- The UNIFY algorithm will search all the related sentences in the knowledge base , which could unify with Knows(John, x)
- UNIFY(Knows(John, x), Knows(John, Jane)) = {x/Jane}

  UNIFY(Knows(John, x), Knows(y, Bill)) = {x/Bill, y/John}

  UNIFY(Knows(John, x), Knows(x,Elizabeth)) = fail.

  The last unification fails because x cannot take on the values John and Elizabeth at the same time.

Conditions for Unification

1. **Predicate symbol must be same.**

   Knows(John, Jane)

   Brother(John, Jane) fail

2. **Number of arguments in both expressions must be identical.**

   Hate(Marry)

   Hate(Marry, John) fail

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

   UNIFY(Knows(John, x), Knows(x, Elizabeth)) = fail.

THANK YOU