CS 461
Artificial Intelligence

Dr. Hashim Yasin

Inference Rules for Quantifiers

Universal Instantiation

- We can **infer any sentence** obtained by *substituting a ground term for the variable*.
- Ground term is the term that is without variables.

E.g., $\forall x \;\; King(x) \wedge Greedy(x) \;\Rightarrow\; Evil(x)$ yields

$$
\begin{aligned}
&King(John) \wedge Greedy(John) \;\Rightarrow\; Evil(John) \\
&King(Richard) \wedge Greedy(Richard) \;\Rightarrow\; Evil(Richard) \\
&King(Father(John)) \wedge Greedy(Father(John)) \;\Rightarrow\; Evil(Father(John)) \\
&\quad \vdots
\end{aligned}
$$

Inference Rules for Quantifiers

Universal Instantiation

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

for any variable v and ground term g

- $\text{SUBST}(\theta, \alpha)$ denotes the result of applying the substitution θ to the sentence α.

Inference Rules for Quantifiers

Existential Instantiation

- The variable is replaced by a single ***new constant symbol***.
- For any sentence $\boldsymbol{\alpha}$, variable $\boldsymbol{v}$, and constant symbol $\boldsymbol{K}$ that ***does not appear elsewhere in the knowledge base***,

$$\boxed{\frac{\exists v \quad \alpha}{\textsc{Subst}(\{v/k\}, \alpha)}\ .}$$

Inference Rules for Quantifiers

Existential Instantiation

E.g., $\exists x \; Crown(x) \wedge OnHead(x, John)$ yields

$$Crown(C_1) \wedge OnHead(C_1, John)$$

- C_1 is the constant which does not appear elsewhere in the knowledge base. Such a constant is called **Skolem constant** and the process is called **Skolemization**.

FOL to Propositional Inference

- Suppose the **KB** consisits of following sentances:

$$\forall x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)$$

- Instantiating the universal sentence in ***all*** possible ways, we have

$$King(John) \land Greedy(John) \;\Rightarrow\; Evil(John)$$
$$King(Richard) \land Greedy(Richard) \;\Rightarrow\; Evil(Richard)$$
$$King(John)$$
$$Greedy(John)$$
$$Brother(Richard, John)$$

This KB is propositionalized

A First-order Inference Rule

Inference Rule

- The inference process mentioned in previous example can be captured as a single inference rule that we call **Generalized Modus Ponens**

For atomic sentences p_i, p_i', and q, where there is a substitution θ such that $\text{SUBST}(\theta, p_i') = \text{SUBST}(\theta, p_i)$, for all i,

$$\frac{\begin{array}{l}(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q) \\ p_1',\ p_2',\ \ldots,\ p_n',\end{array}}{\text{SUBST}(\theta, q)}$$

$$\boxed{\frac{\begin{array}{c}\alpha \rightarrow \beta \\ \alpha\end{array}}{\beta}}$$

- The conclusion is the result of applying the substitution $\boldsymbol{\theta}$ to the consequent $\boldsymbol{q}$.

Inference Rule

Thus, from $p_1', \ldots, p_n'$ we can infer

$$\textsc{Subst}(\theta, p_1') \wedge \ldots \wedge \textsc{Subst}(\theta, p_n')$$

and from the implication $p_1 \wedge \ldots \wedge p_n \Rightarrow q$ we can infer

$$\textsc{Subst}(\theta, p_1) \wedge \ldots \wedge \textsc{Subst}(\theta, p_n) \;\Rightarrow\; \textsc{Subst}(\theta, q)\ .$$

- $\boldsymbol{\theta}$ in Generalized Modus Ponens is defined so that

$$\textsc{Subst}(\theta, p_i') = \textsc{Subst}(\theta, p_i) \quad \textit{for all } i$$

Inference Rule

$$\text{SUBST}(\theta, p_i{}') = \text{SUBST}(\theta, p_i)$$

- Therefore the *first of these two sentences matches the premise of the second exactly*. Hence,

$$\text{SUBST}(\theta, p_1) \wedge \ldots \wedge \text{SUBST}(\theta, p_n) \;\Rightarrow\; \text{SUBST}(\theta, q)$$

$$\text{SUBST}(\theta, p_1{}') \wedge \ldots \wedge \text{SUBST}(\theta, p_n{}')$$

$\text{SUBST}(\theta, q)$ follows by Modus Ponens.

$$\frac{\alpha \to \beta \qquad \alpha}{\beta}$$

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

Inference Rule … Example 1

$$\frac{\begin{array}{c}\alpha \rightarrow \beta \\ \alpha\end{array}}{\beta}$$

$$Faster(x,y) \wedge Faster(y,z) \Rightarrow Faster(x,z)$$
$$Faster(Bob,Pat)$$
$$Faster(Pat,Steve)$$

$\boldsymbol{\alpha \rightarrow \beta}$
with unification
we can prove α

$$\frac{\begin{array}{c}(p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q) \\ p_1', \; p_2', \; \ldots, \; p_n',\end{array}}{q\theta}$$

where $p_i'\theta = p_i\theta$ for all i

$$\begin{aligned}
\text{E.g. } p_1' &= \text{Faster(Bob,Pat)} \\
p_2' &= \text{Faster(Pat,Steve)} \\
p_1 \wedge p_2 \Rightarrow q &= Faster(x,y) \wedge Faster(y,z) \Rightarrow Faster(x,z) \\
\theta &= \{x/Bob, y/Pat, z/Steve\} \\
q\theta &= Faster(Bob, Steve)
\end{aligned}$$

Inference Rule … Example 2

$$\begin{aligned}
&\forall x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x) \\
&King(John) \\
&Greedy(John)
\end{aligned}$$

- Suppose that instead of knowing $Greedy(John)$, we know that *everyone* is greedy:

$$\forall y \;\; Greedy(y)$$

- The knowledge base becomes,

$$\begin{aligned}
&\forall x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x) \\
&King(John) \\
&\forall y \;\; Greedy(y)
\end{aligned}$$

Inference Rule … Example 2

$$\frac{(p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \quad p_1{}', \; p_2{}', \; \ldots, \; p_n{}',}{\textsc{Subst}(\theta, q)}$$

$$\forall x \;\; King(x) \land Greedy(x) \;\Rightarrow\; Evil(x)$$
$$King(John)$$
$$\forall y \;\; Greedy(y)$$

- John is a king (given) and John is greedy (because everyone is greedy).

$p_1{}'$ is $King(John)$	p_1 is $King(x)$
$p_2{}'$ is $Greedy(y)$	p_2 is $Greedy(x)$
θ is $\{x/John, y/John\}$	q is $Evil(x)$
$\textsc{Subst}(\theta, q)$ is $Evil(John)$.	

Inference Rule

- **Generalized Modus Ponens** is a ***sound*** inference rule.
- First, we observe that, for any sentence $\boldsymbol{p}$ (whose variables are assumed to be universally quantified) and for any **substitution $\boldsymbol{\theta}$**,

$$p \models \textsc{Subst}(\theta, p)$$

holds by **Universal Instantiation**.

- It holds in particular for a $\boldsymbol{\theta}$ that satisfies the conditions of the Generalized Modus Ponens rule.

Unification

Unification

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

$$\text{UNIFY}(p, q) = \theta \text{ where } \text{SUBST}(\theta, p) = \text{SUBST}(\theta, q) .$$

- Suppose we have a query **AskVars(Knows(John, x)): whom does John know?**
- To answer this question, we have to find all sentences in the knowledge base that unify with **Knows(John, x).**

Unification

- Here are the results of the unification

$$\text{UNIFY}(Knows(John, x),\ Knows(John, Jane)) = \{x/Jane\}$$
$$\text{UNIFY}(Knows(John, x),\ Knows(y, Bill)) = \{x/Bill, y/John\}$$
$$\text{UNIFY}(Knows(John, x),\ Knows(y, Mother(y))) = \{y/John, x/Mother(John)\}$$
$$\text{UNIFY}(Knows(John, x),\ Knows(x, Elizabeth)) = fail\ .$$

- The last unification fails because **x** cannot take on the values **John** and **Elizabeth** at the same time
- **Knows(x, Elizabeth)** means **"Everyone knows Elizabeth,"** and we can infer that **John knows Elizabeth**

Unification --- Standardizing Apart

- This problem arises because two sentences happen to use the **same variable** name, **x**
- The problem can be avoided by **standardizing apart** which means **renaming its variables** to avoid name clashes.
- **Standardizing apart** eliminates overlap of variables.
- For example, we can rename **x** in **Knows(x, Elizabeth)** to $\mathbf{x_{17}}$ (a new variable name) without changing its meaning.

$$\text{UNIFY}(Knows(John, x),\ Knows(x_{17}, Elizabeth)) = \{x/Elizabeth, x_{17}/John\}$$

Most General Unifier (MGU)

- **Example:**

$$\text{UNIFY}(Knows(John, x), Knows(y, z))$$

could return,

$$\{y/John, x/z\}$$

$$\{y/John, x/John, z/John\}.$$

- The first unifier gives $Knows(John, z)$ as the result of unification, whereas the second gives $Knows(John, John)$.
- The **first unifier is more general** than the second, because it ***places fewer restrictions on the values of the variables***.

Unification Example

- $Q(y, G(A, B))$ and $Q(G(x, x), y)$.

- $O(F(y), y)$ and $O(F(x), J)$.

Unification Example

- $Q\big(y, G(A, B)\big)$ and $Q(G(x, x), y)$.

Progressive unification:

Q(<u>y</u>, G(A, B)), Q(<u>G(x, x)</u>, y) : {<u>y/G(x, x)</u>},

Q(G(x, x), <u>G(A, B)</u>), Q(G(x, x), <u>G(x, x)</u>) : {y/G(x, x)} *needs recursion*

Q(G(x, x), G(<u>A</u>, B)), Q(G(x, x), G(<u>x</u>, x)) : {y/G(x, x), <u>x/A</u>}

Q(G(A, A), G(A, <u>B</u>)), Q(G(A, A), G(A, <u>A</u>)) : {y/G(x, x), x/A}

Unification Example

- $Q\big(y, G(A, B)\big)$ and $Q(G(x, x), y)$.

Progressive unification:

Q(y, G(A, B)), Q(G(x, x), y) : {y/G(x, x)},

Q(G(x, x), G(A, B)), Q(G(x, x), G(x, x)) : {y/G(x, x)} *needs recursion*

Q(G(x, x), G(A, B)), Q(G(x, x), G(x, x)) : {y/G(x, x), x/A}

Q(G(A, A), G(A, B)), Q(G(A, A), G(A, A)) : {y/G(x, x), x/A}

Cannot unify *constant A* with *constant B*.

Unification Example

- $O(F(y), y)$ and $O(F(x), J)$.

Progressive unification:

O (<u>F (y)</u>, y), O (<u>F(x)</u>, J) : {} *needs recursion*
O (F (<u>y</u>), y), O (F(<u>x</u>), J) : {y/x}
O (F (x), <u>x</u>), O (F (x), <u>J</u>) : {y/x, x/J} = {y/J, x/J}
O (F (J), J), O (F (J), J) : {y/x, x/J} = {y/J, x/J}

FOL Examples

Kinship Domain: For example, one's mother is one's female parent:

$$\forall m, c \;\; Mother(c) = m \;\Leftrightarrow\; Female(m) \land Parent(m, c) \,.$$

One's husband is one's male spouse:

$$\forall w, h \;\; Husband(h, w) \;\Leftrightarrow\; Male(h) \land Spouse(h, w) \,.$$

Male and female are disjoint categories:

$$\forall x \;\; Male(x) \;\Leftrightarrow\; \neg Female(x) \,.$$

Parent and child are inverse relations:

$$\forall p, c \;\; Parent(p, c) \;\Leftrightarrow\; Child(c, p) \,.$$

A grandparent is a parent of one's parent:

$$\forall g, c \;\; Grandparent(g, c) \;\Leftrightarrow\; \exists p \;\; Parent(g, p) \land Parent(p, c) \,.$$

A sibling is another child of one's parents:

$$\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; x \neq y \land \exists p \;\; Parent(p, x) \land Parent(p, y)$$

Reading Material

- **Artificial Intelligence,** A Modern Approach

  Stuart J. Russell and Peter Norvig
  - **Chapter 8 & 9.**