AI 2002
Artificial Intelligence

Dr. Hashim Yasin

First Order Logic

Connection between ∀ and ∃

- Asserting that **"Everyone dislikes parsnips"** is the same as asserting there does not exist someone who likes them, and vice versa:

$$\forall x \;\; \neg Likes(x, Parsnips) \quad \text{is equivalent to} \quad \neg \exists x \;\; Likes(x, Parsnips)$$

- We can go one step further: **"Everyone likes ice cream**" means that there is no one who does not like ice cream:

$$\forall x \;\; Likes(x, IceCream) \quad \text{is equivalent to} \quad \neg \exists x \;\; \neg Likes(x, IceCream)$$

Connection between ∀ and ∃

- **∀ is really conjunction** over the universe of objects while **∃ is a disjunction**.
- **Quantifiers obey De Morgan's rules**. The De Morgan rules for quantified and unquantified sentences are as follows:

$$\forall x \ \neg Likes(x, Parsnips) \quad \text{is equivalent to} \quad \neg\exists x \ Likes(x, Parsnips)$$

$$\forall x \ Likes(x, IceCream) \quad \text{is equivalent to} \quad \neg\exists x \ \neg Likes(x, IceCream)$$

$$\boxed{\begin{array}{llll} \neg\exists x \ P \equiv \forall x \ \neg P & \qquad & \neg(P \lor Q) & \equiv \neg P \land \neg Q \\ \neg\forall x \ P \equiv \exists x \ \neg P & & \neg(P \land Q) & \equiv \neg P \lor \neg Q \\ \neg\exists x \ \neg P \equiv \forall x \ P & & P \land Q & \equiv \neg(\neg P \lor \neg Q) \\ \neg\forall x \ \neg P \equiv \exists x \ P & & P \lor Q & \equiv \neg(\neg P \land \neg Q) \end{array}}$$

Equality

- We can use the **equality** symbol to signify ***that two terms refer to the same object***. For example

$$Father(John) = Henry$$

- The equality symbol can be used to state facts about a given function.
- To say that ***Richard has at least two brothers***,

$$\exists x, y \;\; Brother(x, Richard) \land Brother(y, Richard)$$

- The above sentence does not have the intended meaning. The correct version is:

$$\exists x, y \;\; Brother(x, Richard) \land Brother(y, Richard) \land \neg(x = y)$$

FOL - Syntax

$$
\begin{aligned}
Sentence &\rightarrow AtomicSentence \mid ComplexSentence \\
AtomicSentence &\rightarrow Predicate \mid Predicate(Term, \ldots) \mid Term = Term \\
ComplexSentence &\rightarrow (\ Sentence\ ) \mid [\ Sentence\ ] \\
&\mid \neg\ Sentence \\
&\mid Sentence \land Sentence \\
&\mid Sentence \lor Sentence \\
&\mid Sentence \ \Rightarrow \ Sentence \\
&\mid Sentence \ \Leftrightarrow \ Sentence \\
&\mid Quantifier\ Variable, \ldots\ Sentence
\end{aligned}
$$

FOL - Syntax

$$
\begin{array}{rcl}
\mathit{Term} & \rightarrow & \mathit{Function}(\mathit{Term}, \ldots) \\
& | & \mathit{Constant} \\
& | & \mathit{Variable} \\
\\
\mathit{Quantifier} & \rightarrow & \forall \mid \exists \\
\mathit{Constant} & \rightarrow & A \mid X_1 \mid \mathit{John} \mid \cdots \\
\mathit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\mathit{Predicate} & \rightarrow & \mathit{True} \mid \mathit{False} \mid \mathit{After} \mid \mathit{Loves} \mid \mathit{Raining} \mid \cdots \\
\mathit{Function} & \rightarrow & \mathit{Mother} \mid \mathit{LeftLeg} \mid \cdots \\
\text{OPERATOR PRECEDENCE} & : & \neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{array}
$$

Assertions and Queries in FOL

- Sentences are added to a knowledge base using **TELL**, exactly as in propositional logic. Such sentences are called **assertions.**

$$\text{TELL}(KB,\ King(John))\ .$$
$$\text{TELL}(KB,\ Person(Richard))\ .$$
$$\text{TELL}(KB,\ \forall x\ \ King(x)\ \Rightarrow\ Person(x))$$

- We can ask questions of the knowledge base using **ASK**. For example,

$$\left.\begin{array}{l}\text{ASK}(KB,\ King(John))\\ \text{ASK}(KB,\ Person(John))\\ \text{ASK}(KB,\ \exists x\ \ Person(x))\end{array}\right\}\ \textbf{Return \underline{true}}$$

Assertions and Queries in FOL

- If we want to know **what value of x** makes the sentence true, we will need a different function, **ASKVARS**, which we call with

$$\textsc{AskVars}(KB, Person(x))$$

- which yields a stream of answers. In this case (given example) there will be two answers: **{x/John}** and **{x/Richard}.**
- Such an answer is called a **substitution** or **binding list**.

Inference in First Order Logic

Inference in First Order Logic

<u>Two ways</u> of inference in First Order Logic

- The *first-order* inference can be done by converting the knowledge base to *propositional* logic
  - some simple inference rules that can be applied to sentences with quantifiers to obtain sentences without quantifiers.

- The inference methods that manipulate first-order sentences directly.

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., $\forall x \;\; King(x) \land Greedy(x) \Rightarrow Evil(x)$ yields

$$King(John) \land Greedy(John) \Rightarrow Evil(John)$$
$$King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)$$
$$King(Father(John)) \land Greedy(Father(John)) \Rightarrow Evil(Father(John))$$
$$\vdots$$

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

Universal Instantiation

Example

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

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

- The three sentences are obtained with substitutions **{x/John}, {x/Richard }**, and **{x/Father (John)}**.

Inference Rules for Quantifiers

Existential Instantiation

- The variable is replaced by a single ***new constant symbol***.
- For any sentence α, variable v, and constant symbol k that ***does not appear elsewhere in the knowledge base***,

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

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**.

Inference Rules for Quantifiers

- **Universal Instantiation** can be applied **several** times to add new sentences; the new KB is **logically equivalent** to the old one.
- **Existential Instantiation** can be applied **once** to replace the existential sentence; the new KB is **NOT equivalent** to the old,
- But it is **inferentially equivalent** in a sense that it is *satisfiable iff the old KB was satisfiable*.

$$\exists x \; Kill(x, Victim)$$

$$Kill(Murderer, Victim).$$

FOL to Propositional Inference

FOL to Propositional Inference

- In order **to reduce FOL to propositional inference**, we must have rules for inferring non-quantified sentences from quantified sentences.

For ∃:

- An **existentially quantified sentence** can be replaced by one instantiation.

For ∀:

- A **universally quantified sentence** can be replaced by the set of *all possible* instantiations.

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

Problems in Propositionalization

- Propositionalization seems to generate the lots of **irrelevant sentences**. ***For Example***
- Given the query **Evil(x)** for the following KB,

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

- It may generate sentences such as

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

The inference is that John is evil

Problems in Propositionalization

- Propositionalization seems to generate lots of **irrelevant sentences**.

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

- The propositionalization produces lots of facts such as $Greedy(Richard)$ that are irrelevant

With p k-ary predicates and n constants, there are $p \cdot n^k$ instantiations

- With *function symbols*, it gets **much much worse!**

Problems in Propositionalization

Solution:

- **The inference is that John is evil—{x/John}** solves the query **Evil(x)**—The **substitution θ ={x/John}** achieves that goal.
- If there is some substitution $\boldsymbol{\theta}$
  - that makes **the premise of the implication** identical to sentences *already in the knowledge base*,
  - then we can assert the conclusion of the implication, after applying θ.
- In this case, the substitution $\theta = \{x/John\}$ achieves that aim.

Problems in Propositionalization

- For Example,

$$
\begin{aligned}
&King(John) \land Greedy(John) \Rightarrow Evil(John) \\
&King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard) \\
&King(John) \\
&Greedy(John) \\
&Brother(Richard, John)
\end{aligned}
$$

- makes **the premise of the implication** identical to sentences *already in the knowledge base*

Problems in Propositionalization

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

$$\forall x \;\; King(x) \wedge Greedy(x) \;\Rightarrow\; Evil(x)$$
$$King(John)$$
$$\cancel{Greedy(John)} \quad \forall y \; Greedy(y)$$
$$Brother(Richard, John)$$

- Then we would still able to conclude that **Evil(John)**, because we know that
  - **John is a king (given)**
  - **John is greedy (because everyone is greedy).**

Problems in Propositionalization

x/y

- Apply the **substitution {x/John, y/John}** to
  - the implication premises **King(x)** and **Greedy(x)**
  - the knowledge-base sentences **King(John)** and **Greedy(y)** will make them identical.
- In this way, we can infer the **conclusion of the implication**.

Reading Material

- **Artificial Intelligence,** A Modern Approach

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