Artificial Intelligence
AI 2002
Lecture 15

Mahzaib Younas

Lecturer Department of Computer Science

FAST NUCES CFD

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:

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

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

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

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***,

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

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

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

FOL - Syntax

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

FOL - Syntax

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

Assertions

- Sentences are added to a knowledge base using TELL are called assertions
- We want to TELL things to the KB

TELL(KB, King(John))

TELL(KB, $\forall$ x king(x) => Person(x))

John is a king and that king is a person.

Queries

- Questions are asked to the knowledge base using ASK called as queries or goals.
- We want to ASK things to the KB

TELL(KB, King(John))

TELL(KB, ∀ x king(x) => Person(x))

John is a king and that king is a person.

$$\text{Ask}(KB,\ King(John))$$

$$\text{Ask}(KB,\ Person(John))$$

Assertions and Queries in FOL

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

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

$$\textsc{Ask}(KB,\ King(John))$$

$$\textsc{Ask}(KB,\ Person(John))$$
$$\textsc{Ask}(KB,\ \exists x\ \ Person(x))$$

Inferences in First Order Logics

Inference in First Order Logic

Two ways 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., $\forall x \;\; King(x) \wedge Greedy(x) \;\Rightarrow\; Evil(x)$ yields

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

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

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

- 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 $\boldsymbol{\alpha}$, variable $\boldsymbol{v}$, and constant symbol $\boldsymbol{k}$ that ***does not appear elsewhere in the knowledge base***,

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

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 Interference

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 *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,

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

- 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***:

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

- 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**.

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

- The condition for unification are

1. Predicate symbol must be same in both sentences.
2. Number of arguments in both expression must be identical
3. Unification will fail if the two similar variable present in the same expression.

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.

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

Unification Example

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

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

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}

Unification Example

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

<u>Progressive unification:</u>

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}

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

FOL Examples

Kinship Domain:

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.**