First-Order Logic

LESSON 8

Reading

Chaper 8

Outline

Why FOL?

Syntax and semantics of FOL

Using FOL

Wumpus world in FOL

Knowledge engineering in FOL

Pros and cons of propositional logic

- ☺ Propositional logic is declarative
- ☺ Propositional logic allows partial/disjunctive/negated information
  - (unlike most data structures and databases)
- ☺Propositional logic is compositional:
- ☺
  - meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
- ☺ Meaning in propositional logic is context-independent
  - (unlike natural language, where meaning depends on context)
- ☹ Propositional logic has very limited expressive power
  - (unlike natural language)
  - E.g., cannot say "pits cause breezes in adjacent squares“
    - except by writing one sentence for each square

First-order logic

Whereas propositional logic assumes the world contains facts,

first-order logic (like natural language) assumes the world contains

- Objects: people, houses, numbers, colors, baseball games, wars, …
-
- Relations: red, round, prime, brother of, bigger than, part of, comes between, …
- Functions: father of, best friend, one more than, plus, …
-

Syntax of FOL: Basic elements

Constants	KingJohn, 2, NUS,...
Predicates	Brother, >,...
Functions	Sqrt, LeftLegOf,...
Variables	x, y, a, b,...
Connectives	$\neg$, $\Rightarrow$, $\wedge$, $\vee$, $\Leftrightarrow$
Equality	=
Quantifiers	$\forall$, $\exists$

Atomic sentences

Atomic sentence = *predicate* (*$term_1$*,...,*$term_n$*) or *$term_1$* = *$term_2$*

Term = *function* (*$term_1$*,...,*$term_n$*) or *constant* or *variable*

E.g., *Brother(KingJohn,RichardTheLionheart) > (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))*

Complex sentences

Complex sentences are made from atomic sentences using connectives

$$\neg S,\ S_1 \wedge S_2,\ S_1 \vee S_2,\ S_1 \Rightarrow S_2,\ S_1 \Leftrightarrow S_2,$$

E.g. *Sibling(KingJohn,Richard)* $\Rightarrow$ *Sibling(Richard,KingJohn)*

$>(1,2) \vee \leq (1,2)$

$>(1,2) \wedge \neg >(1,2)$

Truth in first-order logic

Sentences are true with respect to a model and an interpretation

Model contains objects (domain elements) and relations among them

Interpretation specifies referents for

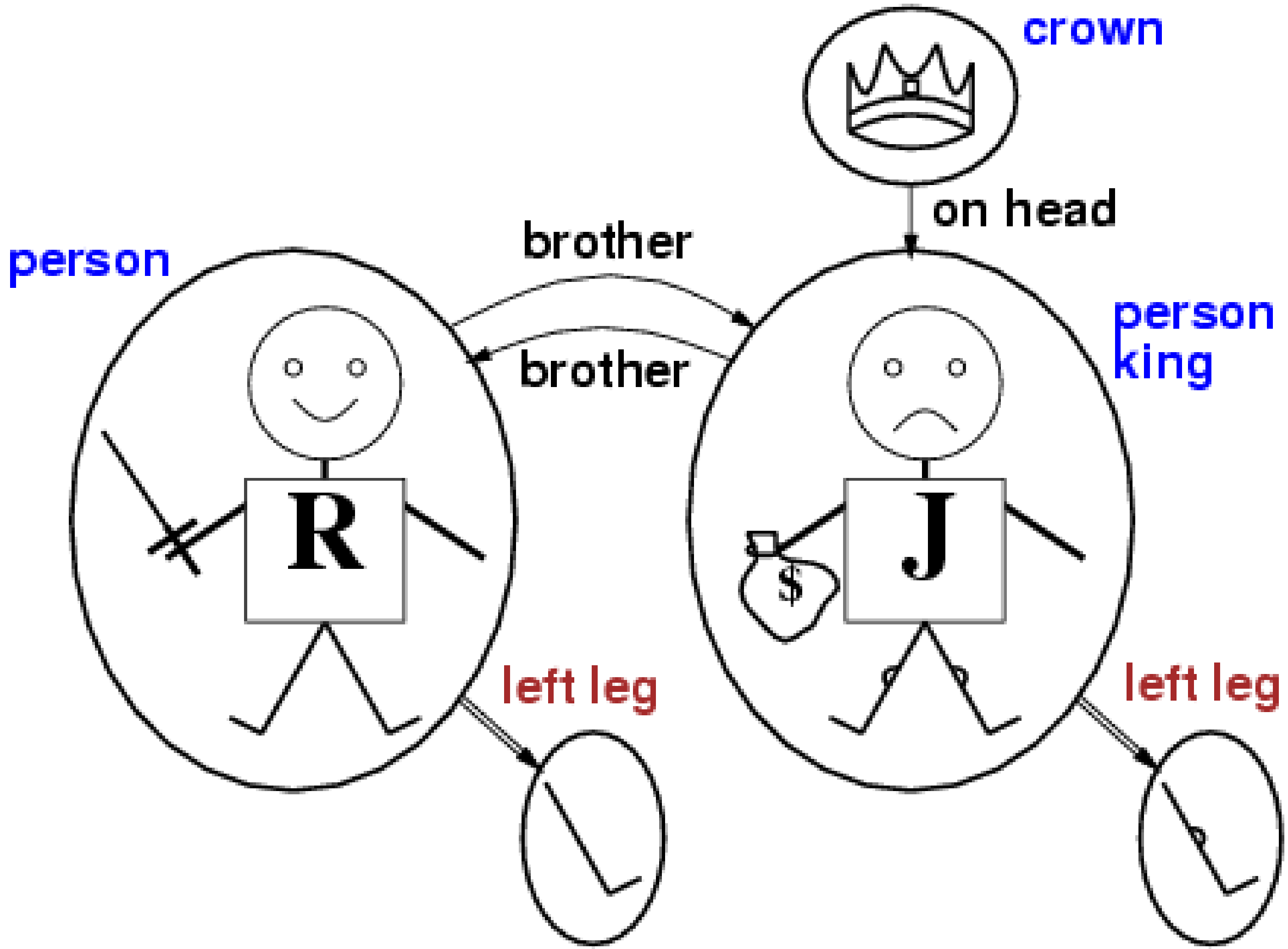
constant symbols	→	objects
predicate symbols	→	relations
function symbols	→	functional relations

An atomic sentence *predicate*($term_1,\ldots,term_n$) is true

iff the objects referred to by $term_1,\ldots,term_n$

are in the relation referred to by *predicate*

Models for FOL: Example



Universal quantification

$\forall$*<variables> <sentence>*

Everyone at NUS is smart:

$\forall x\ At(x,NUS) \Rightarrow Smart(x)$

$\forall x$ *P* is true in a model *m* iff *P* is true with *x* being each possible object in the model

Roughly speaking, equivalent to the conjunction of instantiations of *P*

$$
\begin{aligned}
& At(KingJohn,NUS) \Rightarrow Smart(KingJohn) \\
\wedge \quad & At(Richard,NUS) \Rightarrow Smart(Richard) \\
\wedge \quad & At(NUS,NUS) \Rightarrow Smart(NUS) \\
\wedge \quad & \ldots
\end{aligned}
$$

A common mistake to avoid

Typically, $\Rightarrow$ is the main connective with $\forall$

Common mistake: using $\wedge$ as the main connective with $\forall$:

$\forall x\ At(x,NUS) \wedge Smart(x)$

means "Everyone is at NUS and everyone is smart"

Existential quantification

$\exists$<*variables*> <*sentence*>

Someone at NUS is smart:

$\exists x$ At(x,NUS) $\wedge$ Smart(x)\$

$\exists x$ *P* is true in a model *m* iff *P* is true with *x* being some possible object in the model

Roughly speaking, equivalent to the disjunction of instantiations of *P*

At(KingJohn,NUS) $\wedge$ Smart(KingJohn)
$\vee$ At(Richard,NUS) $\wedge$ Smart(Richard)
$\vee$ At(NUS,NUS) $\wedge$ Smart(NUS)
$\vee$ ...

Another common mistake to avoid

Typically, $\wedge$ is the main connective with $\exists$

Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists x \; At(x,NUS) \Rightarrow Smart(x)$$

is true if there is anyone who is not at NUS!

Properties of quantifiers

∀x ∀y is the same as ∀y ∀x

∃x ∃y is the same as ∃y ∃x

∃x ∀y is not the same as ∀y ∃x

∃x ∀y Loves(x,y)

- "There is a person who loves everyone in the world"
-

∀y ∃x Loves(x,y)

- "Everyone in the world is loved by at least one person"
-

Quantifier duality: each can be expressed using the other

∀x Likes(x,IceCream) ¬∃x ¬Likes(x,IceCream)

∃x Likes(x,Broccoli) ¬∀x ¬Likes(x,Broccoli)

Equality

$term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object

E.g., definition of *Sibling* in terms of *Parent*:

$\forall x,y\ Sibling(x,y) \Leftrightarrow [\neg(x = y) \land \exists m,f\ \neg (m = f) \land$
$Parent(m,x) \land Parent(f,x) \land Parent(m,y) \land Parent(f,y)]$

Using FOL

The kinship domain:

Brothers are siblings

$\forall$x,y *Brother(x,y)* $\Leftrightarrow$ *Sibling(x,y)*

One's mother is one's female parent

$\forall$m,c *Mother(c)* = m $\Leftrightarrow$ *(Female(m)* $\wedge$ *Parent(m,c))*

"Sibling" is symmetric

$\forall$x,y *Sibling(x,y)* $\Leftrightarrow$ *Sibling(y,x)*

Using FOL

The set domain:

$\forall s\ Set(s) \Leftrightarrow (s = \{\}) \lor (\exists x,s_2\ Set(s_2) \land s = \{x|s_2\})$

$\neg\exists x,s\ \{x|s\} = \{\}$

$\forall x,s\ x \in s \Leftrightarrow s = \{x|s\}$

$\forall x,s\ x \in s \Leftrightarrow [\ \exists y,s_2\}\ (s = \{y|s_2\} \land (x = y \lor x \in s_2))]$

$\forall s_1,s_2\ s_1 \subseteq s_2 \Leftrightarrow (\forall x\ x \in s_1 \Rightarrow x \in s_2)$

$\forall s_1,s_2\ (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \land s_2 \subseteq s_1)$

$\forall x,s_1,s_2\ x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \land x \in s_2)$

$\forall x,s_1,s_2\ x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \lor x \in s_2)$

Interacting with FOL KBs

Suppose a wumpus-world agent is using an FOL KB and perceives a smell and a breeze (but no glitter) at *t=5*:

`Tell`(KB,Percept([Smell,Breeze,None],5))

`Ask`(KB,∃a BestAction(a,5))

I.e., does the KB entail some best action at *t=5*?

Answer: *Yes*, {*a/Shoot*} ← substitution (binding list)

Given a sentence *S* and a substitution σ,

*S*σ denotes the result of plugging σ into *S*; e.g.,

S = Smarter(x,y)

σ = {x/Hillary,y/Bill}

*S*σ = Smarter(Hillary,Bill)

`Ask`(KB,S) returns some/all σ such that KB ╞ σ

Knowledge base for the wumpus world

Perception

- $\forall t,s,b$ Percept([s,b,Glitter],t) $\Rightarrow$ Glitter(t)
-

Reflex

- $\forall t$ Glitter(t) $\Rightarrow$ BestAction(Grab,t)

Deducing hidden properties

$\forall x,y,a,b$ *Adjacent*([x,y],[a,b]) $\Leftrightarrow$

[a,b] $\in$ {[x+1,y], [x-1,y],[x,y+1],[x,y-1]}

Properties of squares:

$\forall s,t$ *At*(Agent,s,t) $\wedge$ Breeze(t) $\Rightarrow$ Breezy(s)

Squares are breezy near a pit:

- Diagnostic rule---infer cause from effect
  $\forall$s Breezy(s) $\Rightarrow$ \Exi{r} Adjacent(r,s) $\wedge$ Pit(r)$
- Causal rule---infer effect from cause
  $\forall$r Pit(r) $\Rightarrow$ [$\forall$s Adjacent(r,s) $\Rightarrow$ Breezy(s)$]

Knowledge engineering in FOL

1. Identify the task

2.

2. Assemble the relevant knowledge

3.

3. Decide on a vocabulary of predicates, functions, and constants

4.

4. Encode general knowledge about the domain

5.

5. Encode a description of the specific problem instance

6.

6. Pose queries to the inference procedure and get answers

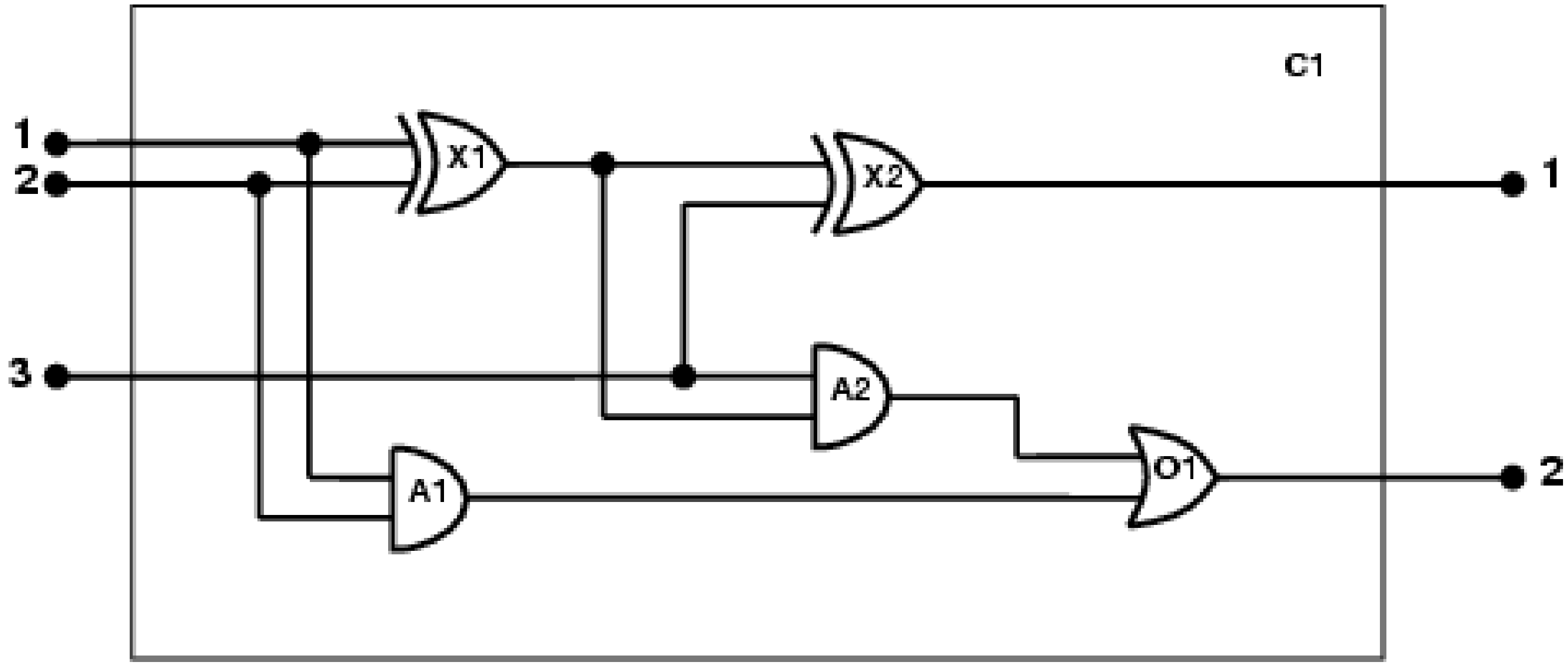
7.

7. Debug the knowledge base

8.

The electronic circuits domain

One-bit full adder



The electronic circuits domain

1. Identify the task
2.
   - Does the circuit actually add properly? (circuit verification)
   -
2. Assemble the relevant knowledge
3.
   - Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
   -
   - Irrelevant: size, shape, color, cost of gates
   -
3. Decide on a vocabulary
4.
   - Alternatives:
   -

     $Type(X_1) = XOR$

     $Type(X_1, XOR)$
     $XOR(X_1)$

The electronic circuits domain

4. Encode general knowledge of the domain
5.
   - $\forall t_1,t_2\ Connected(t_1, t_2) \Rightarrow Signal(t_1) = Signal(t_2)$
   - $\forall t\ Signal(t) = 1 \lor Signal(t) = 0$
   -
   - $1 \neq 0$
   -
   - $\forall t_1,t_2\ Connected(t_1, t_2) \Rightarrow Connected(t_2, t_1)$
   -
   - $\forall g\ Type(g) = OR \Rightarrow Signal(Out(1,g)) = 1 \Leftrightarrow \exists n\ Signal(In(n,g)) = 1$
   -
   - $\forall g\ Type(g) = AND \Rightarrow Signal(Out(1,g)) = 0 \Leftrightarrow \exists n\ Signal(In(n,g)) = 0$
   -
   - $\forall g\ Type(g) = XOR \Rightarrow Signal(Out(1,g)) = 1 \Leftrightarrow Signal(In(1,g)) \neq Signal(In(2,g))$
   -
   - $\forall g\ Type(g) = NOT \Rightarrow Signal(Out(1,g)) \neq Signal(In(1,g))$
   -

The electronic circuits domain

5. Encode the specific problem instance
6.

$Type(X_1) = XOR$ $Type(X_2) = XOR$
$Type(A_1) = AND$ $Type(A_2) = AND$
$Type(O_1) = OR$

$Connected(Out(1,X_1),In(1,X_2))$ $Connected(In(1,C_1),In(1,X_1))$
$Connected(Out(1,X_1),In(2,A_2))$ $Connected(In(1,C_1),In(1,A_1))$
$Connected(Out(1,A_2),In(1,O_1))$ $Connected(In(2,C_1),In(2,X_1))$
$Connected(Out(1,A_1),In(2,O_1))$ $Connected(In(2,C_1),In(2,A_1))$
$Connected(Out(1,X_2),Out(1,C_1))$ $Connected(In(3,C_1),In(2,X_2))$
$Connected(Out(1,O_1),Out(2,C_1))$ $Connected(In(3,C_1),In(1,A_2))$

The electronic circuits domain

6. Pose queries to the inference procedure
7. What are the possible sets of values of all the terminals for the adder circuit?

$$\exists i_1,i_2,i_3,o_1,o_2\ Signal(In(1,C\_1)) = i_1 \wedge Signal(In(2,C_1)) = i_2 \wedge Signal(In(3,C_1)) = i_3 \wedge Signal(Out(1,C_1)) = o_1 \wedge Signal(Out(2,C_1)) = o_2$$

7. Debug the knowledge base
8. May have omitted assertions like $1 \neq 0$

Summary

First-order logic:

- objects and relations are semantic primitives
- syntax: constants, functions, predicates, equality, quantifiers
-

Increased expressive power: sufficient to define wumpus world