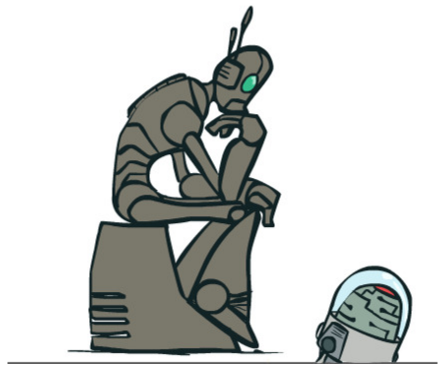
Artificial Intelligence

Chapter 8: 1st Order (Predicate) Logic



Updates and Additions: Dr. Siamak Sarmady
By: Tom Lenaerts
Université Libre de Bruxelles

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 only 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, …

Logics in General

- Ontological Commitment: What exists in the world — TRUTH
- Epistemological Commitment: What an agent believes about facts — BELIEF

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0, 1]$
Fuzzy logic	degree of truth $\in [0, 1]$	known interval value

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*

- Examples:

  Brother(KingJohn,RichardTheLionheart)

  > (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))

Complex sentences

- Complex sentences are made from atomic sentences using connectives

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

Examples:

Sibling(KingJohn,Richard) $\Rightarrow$ *Sibling(Richard,KingJohn)*
$>(1,2) \vee \leq (1,2)$
$>(1,2) \wedge \neg >(1,2)$

Truth in first-order logic

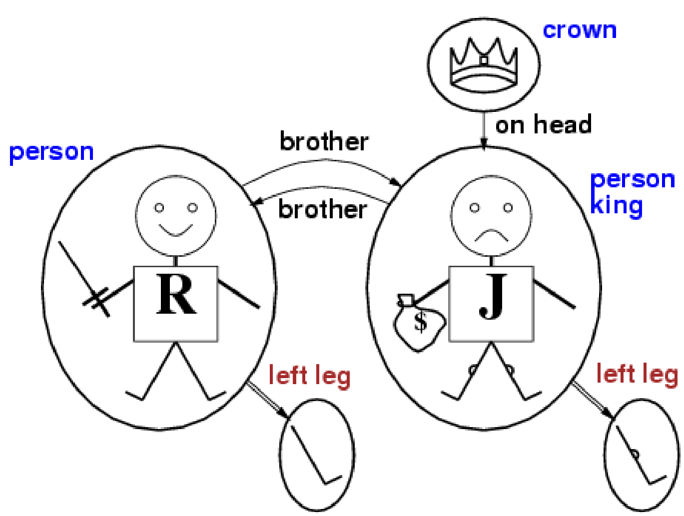
- Sentences are true with respect to a model and an interpretation
- Model contains: objects (domain elements) and relations among them
- Interpretation specifies referents for

  constant symbols → objects

  predicate symbols → relations

  function symbols → functional relations
- An atomic sentence *predicate*($term_1$,...,$term_n$) is true iff the objects referred to by $term_1$,...,$term_n$ are in the relation referred to by *predicate*

Models for FOL: Example



Models for FOL

- We can enumerate the models for a given KB vocabulary:

For each number of domain elements n from 1 to ∞
  For each k-ary predicate P_k in the vocabulary
    For each possible k-ary relation on n objects
      For each constant symbol C in the vocabulary
        For each choice of referent for C from n objects . . .

- Computing entailment by enumerating the models will not be easy !!

Quantifiers

- Allows us to express properties of collections of objects instead of enumerating objects by name
- Universal: "for all" $\forall$
- Existential: "there exists" $\exists$

Universal quantification

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

Everyone at UU (Urmia University) is smart:

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

$\forall x$ *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*

$$
\begin{aligned}
& At(KingJohn,UU) \Rightarrow Smart(KingJohn) \\
\wedge\ & At(Ali,UU) \Rightarrow Smart(Ali) \\
\wedge\ & At(UU,UU) \Rightarrow Smart(UU) \\
\wedge\ & \ldots
\end{aligned}
$$

A common mistake to avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$
  - A universally quantifier is also equivalent to a set of implications over all objects
- Common mistake: using $\wedge$ as the main connective with $\forall$:

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

  means “Everyone is at Urmia Univesity and everyone is smart”

Existential quantification

∃<*variables*> <*sentence*>

Someone at UU is smart:
$\exists x$ At(x, UU) ∧ 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,UU) ∧ Smart(KingJohn)
  ∨ At(Richard,UU) ∧ Smart(Richard)
  ∨ At(UU, UU) ∧ Smart(UU)
  ∨ ...

Another common mistake to avoid

- Typically, ∧ is the main connective with ∃
- Common mistake: using ⇒ as the main connective with ∃:

$\exists x$ At(x, UU) ⇒ Smart(x)
is true if there is anyone who is not at UU!

Properties of quantifiers

$\forall x\ \forall y$ is the same as $\forall y\ \forall x$
$\exists x\ \exists y$ is the same as $\exists y\ \exists x$

$\exists x\ \forall y$ is not the same as $\forall y\ \exists x$
$\exists x\ \forall y$ Loves(x,y)

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

$\forall y\ \exists x$ Loves(x,y)

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

- Quantifier duality: each can be expressed using the other
  $\forall x$ Likes(x,IceCream) $\neg\exists x\ \neg$Likes(x,IceCream)
  $\exists x$ Likes(x,Broccoli) $\neg\forall x\ \neg$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

- Example:

definition of *Sibling* in terms of *Parent*:

m: mother

F: father

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

Predicate vs. Function

Predicate tells that a property exists for that object:

President(Obama,America)=true.

This tells you that property of Obama being President of America is true. But the following tells Putin being Americas president is false:

President(Putin,America)=false.

Functions return the value associated with a specific property of an object like America's President , Ann's mother etc. You give them a value and they will return a value. Let Pres be a function that returns the president of country passed as arguments

Pres(America)=Obama.
Pres(Russia)=Putin.

Simply put, predicate is a function that returns either a true or false!

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

- Given a sentence *S* and a substitution α,
- *S*α denotes the result of plugging α into *S*; e.g.,
  S = Smarter(x,y)
  α = {x/Hillary,y/Bill}
  *S*α = Smarter(Hillary,Bill)

- `Ask`(KB,S) returns some/all α such that KB |= Sα.

- **Domain:** part of the world of about which we will store information.
- **Axiom:** Facts about the domain that other facts will be based on them.
- **Definition:** Any axiom in the form of $\forall x,y\ P(x,y) \Leftrightarrow$ is a definition of P.

21

Using FOL

The family 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)*
- A first cousin is a child of a parent’s sibling
  $\forall$x,y *FirstCousin(x,y)* $\Leftrightarrow$ $\exists$p,ps Parent(p,x) $\wedge$ Sibling(ps,p) $\wedge$ Parent(ps,y)

Using FOL

The set domain:

- $\forall s\ Set(s) \Leftrightarrow (s = \{\}) \vee (\exists x,s_2\ Set(s_2) \wedge s = \{x|s_2\})$
- $\neg\exists x,s\ \{x|s\} = \{\}$
- $\forall x,s\ x \in s \Leftrightarrow s = \{x|s\}$
- $\forall x,s\ x \in s \Leftrightarrow [\exists y,s_2\} (s = \{y|s_2\} \wedge (x = y \vee x \in s_2))]$
- $\forall s_1,s_2\ s_1 \subseteq s_2 \Leftrightarrow (\forall x\ x \in s_1 \Rightarrow x \in s_2)$
- $\forall s_1,s_2\ (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$
- $\forall x,s_1,s_2\ x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$
- $\forall x,s_1,s_2\ x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$

FOL Version of Wumpus World

- Typical percept sentence:
  Percept([Stench,Breeze,Glitter,None,None],5)
- Actions:
  Turn(Right), Turn(Left), Forward, Shoot, Grab, Release, Climb
- To determine best action, construct query:
  $\forall$ a BestAction(a,5)
- ASK solves this and returns {a/Grab}

Knowledge base for the wumpus world

- Perception
  - $\forall$b,g,t Percept([Smell,b,g],t) $\Rightarrow$ Smelt(t)
  - $\forall$s,b,t Percept([s,b,Glitter],t) $\Rightarrow$ Glitter(t)
- Reflex
  - $\forall$t Glitter(t) $\Rightarrow$ BestAction(Grab,t)
- Reflex with internal state
  - $\forall$t Glitter(t) $\wedge\neg$Holding(Gold,t) $\Rightarrow$ BestAction(Grab,t)

Holding(Gold,t) can not be observed: keep track of change.

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 locaton:
$\forall$s,t *At*(Agent,s,t) $\wedge$ Smelt(t) $\Rightarrow$ Smelly(s)
$\forall$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) $\Rightarrow$ $\exists$ r Adjacent(r,s) $\wedge$ Pit(r)
- Causal rule---infer effect from cause (model based reasoning)
  $\forall$r Pit(r) $\Rightarrow$ [$\forall$s Adjacent(r,s) $\Rightarrow$ Breezy(s)]

Knowledge engineering in FOL

1. Identify the task (what will the KB be used for)
2. Assemble the relevant knowledge

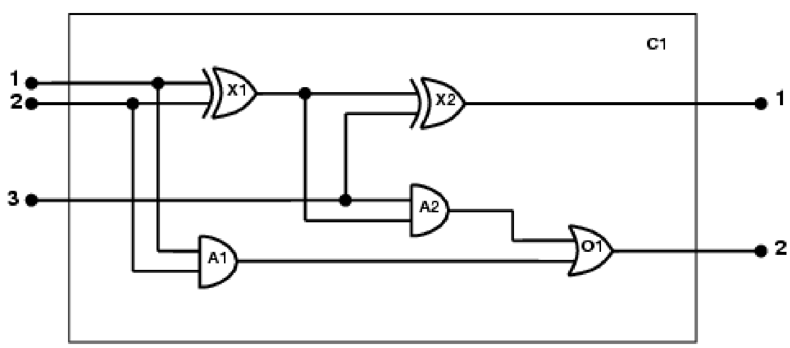
   Knowledge acquisition.
3. Decide on a vocabulary of predicates, functions, and constants

   Translate domain-level knowledge into logic-level names.
4. Encode general knowledge about the domain

   define axioms
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

The electronic circuits domain

One-bit full adder



The electronic circuits domain

- Identify the task
  - Does the circuit actually add properly? (circuit verification)
- Assemble the relevant knowledge
  - Composed of wires and gates;
  - Types of gates (AND, OR, XOR, NOT)
  - Connections between terminals
  - Irrelevant: size, shape, color, cost of gates
- Decide on a vocabulary
  - Alternatives:
    $Type(X_1) = XOR$
    $Type(X_1, XOR)$
    $XOR(X_1)$

The electronic circuits domain

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

The electronic circuits domain

5. Encode the specific problem instance

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

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

The electronic circuits domain

6. Pose queries to the inference procedure

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

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