First-Order Logic

CS4881 Artificial Intelligence

Jay Urbain, Ph.D.

Q: How many Artificial Intelligence (AI) scientists does it take to change a light bulb?

The logical formalism group (16):

- One to figure out how to describe changing in first order logic.
- One to figure out how to describe light bulb changing in second order logic.
- One to show the adequacy of FOL.
- One to show the inadequacy of FOL.
- One to show that light bulb logic is non-monotonic.
- One to show that it isn't non-monotonic.
- One to show how non-monotonic logic is incorporated in FOL.
- One to determine the bindings for the variables.
- One to show the completeness of the solution.
- One to show the consistency of the solution.
- One to show that the two just above are incoherent.
- One to hack a theorem proover for light bulb resolution.
- One to suggest a parallel theory of light bulb logic theorem proving.
- One to show that the parallel theory isn't complete.
- One to indicate how it is a description of human light bulb changing behavior.
- One to call the electrician.

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 (First-order Predicate Calculus)

- Whereas propositional logic assumes the world contains facts, first-order logic (FOL) like natural language assumes the world contains:
  - **Objects (entities)**: people, houses, numbers, colors, baseball games, wars, …
  - **Relations (predicates)**: 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 King John, 2, Jay,...
  - "Grounded"
- Predicates Brother, >,...
  - An operator (relation) in logic that returns True or False.
- Functions Sqrt, LeftLegOf,...
  - Returns elements
- 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 logic *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.

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

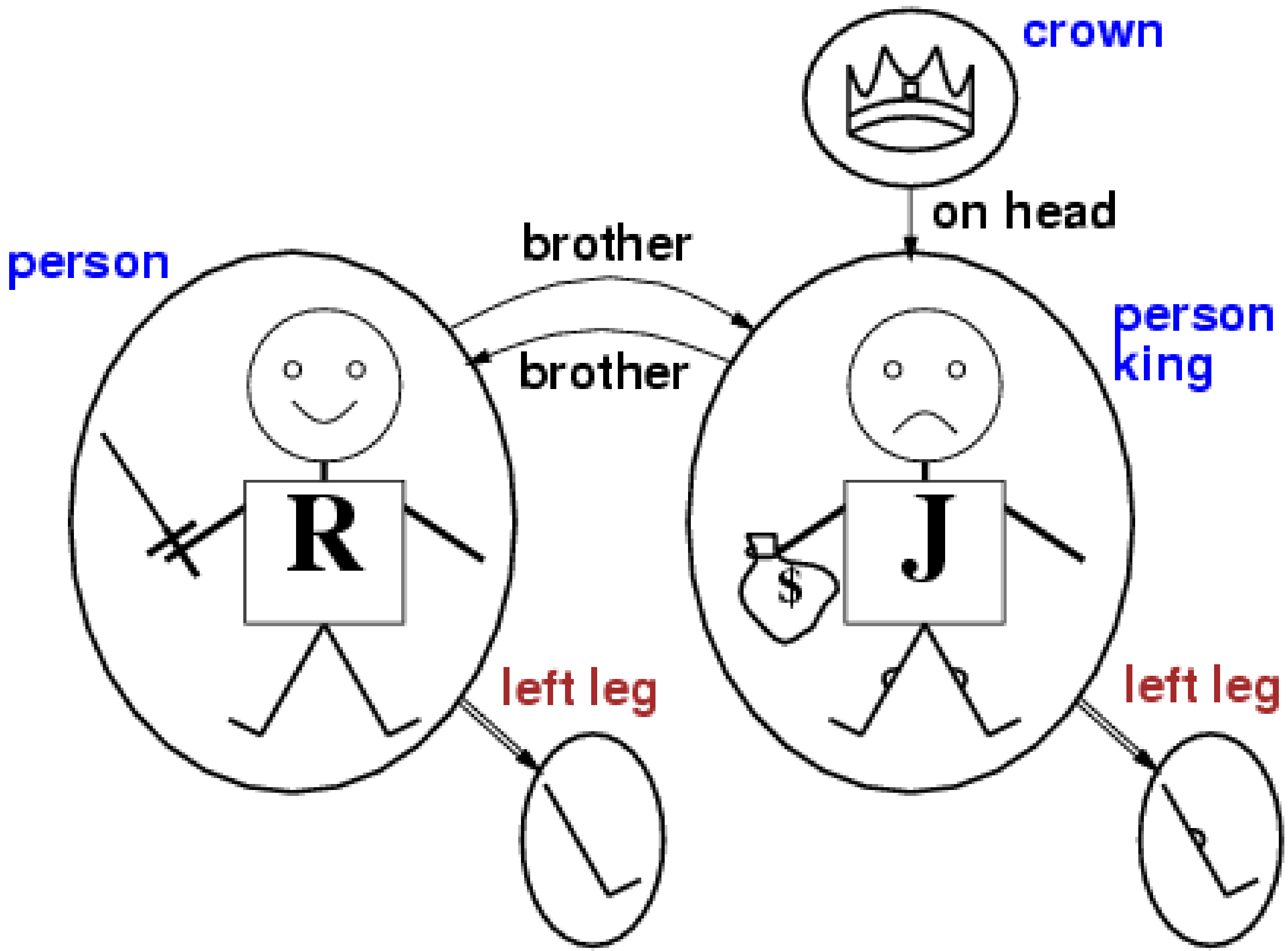
Truth in first-order logic

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



Quantified Variables

- FOL distinguished from Propositional Logic by its use of *quantified variables*.

- Notes:
  - *"First-order" distinguishes FOL from higher order logics in which there are predicates having predicates or functions as arguments.*
  - *In First-order theories, predicates are often associated with sets. The Relational Model is based on FOL.*
  - *In higher order theories, predicates may be interpreted as sets of sets.*

Universal quantification

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

Everyone at MSOE is smart:
$\forall x\ At(x, MSOE) \Rightarrow Smart(x)$ *Is this true?*

- $\forall x$ *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*

$$
\begin{aligned}
& At(KingJohn,MSOE) \Rightarrow Smart(KingJohn) \\
\wedge\ & At(Richard,MSOE) \Rightarrow Smart(Richard) \\
\wedge\ & At(SE/CE,MSOE) \Rightarrow Smart(SE/CE) \\
\wedge\ & \ldots
\end{aligned}
$$

A common mistake to avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$
- Common mistake: using $\wedge$ as the main connective with $\forall$:

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

  means "Everyone is at MSOE *and* everyone is smart"

Existential quantification

- $\exists$<*variables*> <*sentence*>
- Someone at MSOE is smart:
  $\exists x$ At(x, MSOE) $\wedge$ 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,MSOE) $\wedge$ Smart(KingJohn)
  $\vee$ At(Richard, MSOE) $\wedge$ Smart(Richard)
  $\vee$ At(SE/CE, MSOE) $\wedge$ Smart(SE/CE)
  $\vee$ ...

Another common mistake to avoid

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

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

  $$\exists x\ At(x,MSOE) \Rightarrow Smart(x)$$

  is true if there is anyone who is *not* at MSOE or they are smart!

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 (at least one) 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) $\quad\quad \neg\exists x \; \neg$Likes(x,IceCream)
- $\exists x$ Likes(x,Broccoli) $\quad\quad \neg\forall x \; \neg$Likes(x,Broccoli)

Equality

- $term_1 = term_2$ is true under a given interpretation *iff* $term_1$ and $term_2$ refer to the same object

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

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

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 – (Advanced)

Defining the **set** domain:

- The only sets are the empty set and those made by adjoining something to a set:
- $\forall s\ Set(s) \Leftrightarrow (s = \{\}\ ) \vee (\exists x, s_2\ Set(s_2) \wedge s = \{x|s_2\})$
- The empty set has no elements adjoined into it.
- $\neg\exists x,s\ \{x|s\} = \{\}$
- Adjoining an element already inns are the element the set has no effect.
- $\forall x,s\ x \in s \Leftrightarrow s = \{x|s\}$
- The only members of a set are the elements already adjoined into it.
- $\forall x,s\ x \in s \Leftrightarrow [\ \exists y,s_2\}\ (s = \{y|s_2\} \wedge (x = y \vee x \in s_2))]'$
- A set is a subset of another set iff all of the first set is included in the second set.
- $\forall s_1,s_2\ s_1 \subseteq s_2 \Leftrightarrow (\forall x\ x \in s_1 \Rightarrow x \in s_2)$
- Subsets are equal if the are subsets of each other
- $\forall s_1,s_2\ (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$
- An object is in the intersection of two sets if it is a member of both
- $\forall x,s_1,s_2\ x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$
- An object is in the union of two sets if it is a member of eit:her set
- $\forall x,s_1,s_2\ x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$

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

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

Deducing hidden properties

- $\forall x,y,a,b\ \mathit{Adjacent}([x,y],[a,b]) \Leftrightarrow$
  $[a,b] \in \{[x+1,y], [x-1,y],[x,y+1],[x,y-1]\}$

Properties of squares (*s == location*):

- $\forall s,t\ \mathit{At}(\text{Agent},s,t) \wedge \text{Breeze}(t) \Rightarrow \text{Breezy}(s)$

Squares are breezy near a pit:

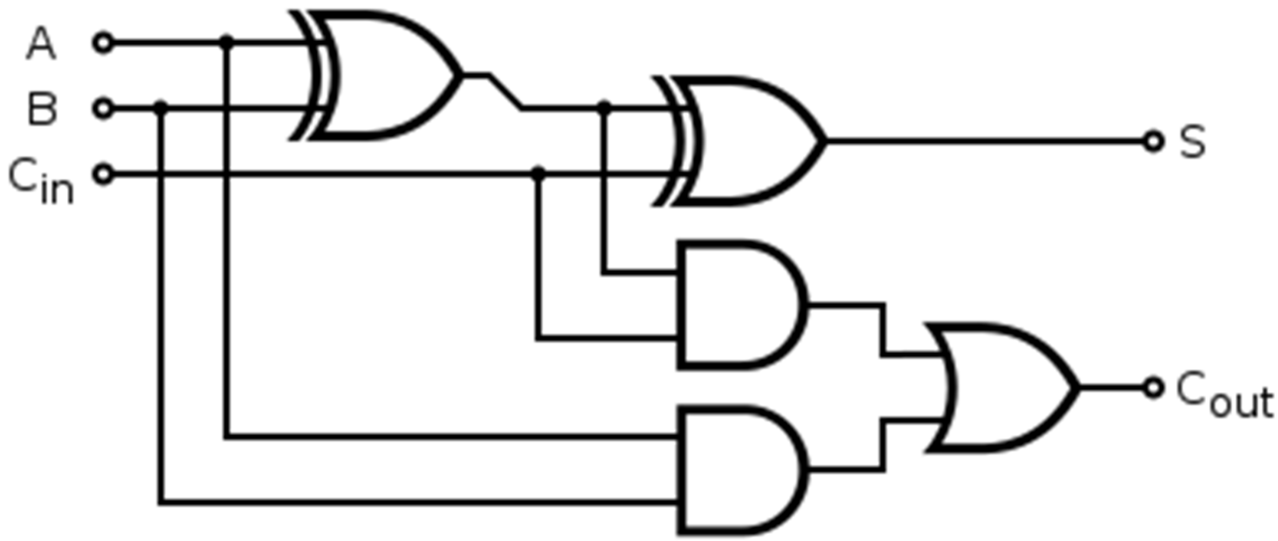
- Diagnostic rule---infer cause from effect
  $\forall s\ \text{Breeze}(s) \Rightarrow \text{Adjacent}(r,s) \wedge \text{Pit}(r)$
- Causal rule---infer effect from cause
  $\forall r\ \text{Pit}(r) \Rightarrow [\forall s\ \text{Adjacent}(r,s) \Rightarrow \text{Breezy}(s)\ ]$

Knowledge engineering in FOL

1. Identify the *task*
2. Assemble the relevant *knowledge*
3. Decide on a *vocabulary* of predicates, functions, and constants
4. *Encode* general knowledge about the domain
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:



$$S = (A \oplus B) \oplus C_{in}$$

$$C_{out} = (A \cdot B) + (C_{in} \cdot (A \oplus B))$$

The electronic circuits domain

1. Identify the task
   – Does the circuit actually add correctly? (circuit verification)

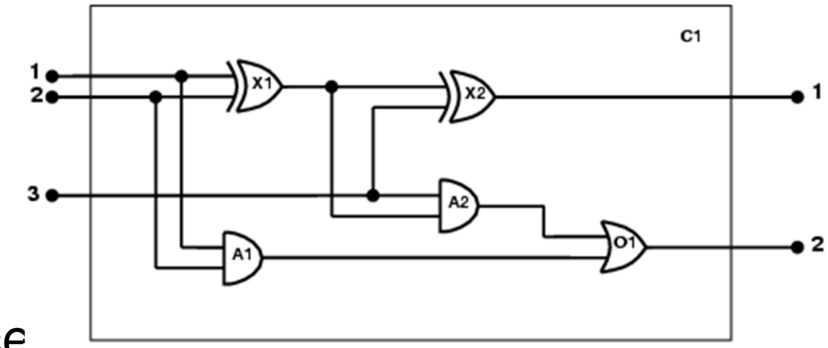
2. Assemble the relevant knowledge
   – Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
   – Irrelevant: size, shape, color, cost of gates

3. 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)
   - $\forall g$ Gate(g) ^Type(g) = OR $\Rightarrow$ Signal(Out(1,g)) = 1 $\Leftrightarrow$ $\exists n$ Signal(In(n,g)) = 1
   - $\forall g$ Gate(g) ^ Type(g) = AND $\Rightarrow$ Signal(Out(1,g)) = 0 $\Leftrightarrow$ $\exists n$ Signal(In(n,g)) = 0
   - $\forall g$ Gate(g) ^ Type(g) = XOR $\Rightarrow$ Signal(Out(1,g)) = 1 $\Leftrightarrow$ Signal(In(1,g)) $\neq$ Signal(In(2,g))
   - $\forall g$ Gate(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(Out(1,X_1),In(2,A_2))
Connected(Out(1,A_2),In(1,O_1))
Connected(Out(1,A_1),In(2,O_1))
Connected(Out(1,X_2),Out(1,C_1))
Connected(Out(1,O_1),Out(2,C_1))

Connected(In(1,C_1),In(1,X_1))
Connected(In(1,C_1),In(1,A_1))
Connected(In(2,C_1),In(2,X_1))
Connected(In(2,C_1),In(2,A_1))
Connected(In(3,C_1),In(2,X_2))
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