FIRST-ORDER LOGIC

CHAPTER 8
Outline

◊ Why FOL?
◊ Syntax and semantics of FOL
◊ Natural Language to Logic
◊ Wumpus world in FOL
Pros and cons of propositional logic

- Propositional logic is **declarative**: pieces of syntax correspond to facts and you can derive new facts from other facts.
- Propositional logic allows partial/disjunctive/negated information (unlike most data structures and databases)
- Propositional logic is **compositional**: meaning of $B_{1,1} \land P_{1,2}$ 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
Formal Languages

Ontological commitment: What exists in the world

Epistemological commitment: What an agent believes about facts

<table>
<thead>
<tr>
<th>Language</th>
<th>Ontological Commitment</th>
<th>Epistemological Commitment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propositional logic</td>
<td>facts</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>First-order logic</td>
<td>facts, objects, relations</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Temporal logic</td>
<td>facts, objects, relations, times</td>
<td>true/false/unknown</td>
</tr>
<tr>
<td>Probability theory</td>
<td>facts</td>
<td>degree of belief 0...1</td>
</tr>
<tr>
<td>Fuzzy logic</td>
<td>degree of truth</td>
<td>degree of belief 0...1</td>
</tr>
</tbody>
</table>
First-order logic

Whereas propositional logic assumes world contains facts, first-order logic (like natural language) assumes the world contains:

- **Objects**: people, houses, numbers, colors, baseball games, wars, centuries . . .
- **Relations**: red, round, prime. . ., brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, . . .
- **Functions**: father of, best friend, one more than, end of . . .
## Syntax of FOL: Basic elements

<table>
<thead>
<tr>
<th>Category</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constants</td>
<td>KingJohn, 2, Sabanci, ...</td>
</tr>
<tr>
<td>Variables</td>
<td>x, y, a, b, ...</td>
</tr>
<tr>
<td>Predicates</td>
<td>Brother, &gt;, ...</td>
</tr>
<tr>
<td>Functions</td>
<td>Sqrt, FatherOf, ...</td>
</tr>
<tr>
<td>Connectives</td>
<td>∧ ∨ ¬ ⇒ ⇔</td>
</tr>
<tr>
<td>Equality</td>
<td>=</td>
</tr>
<tr>
<td>Quantifiers</td>
<td>∀ ∃</td>
</tr>
</tbody>
</table>
Atomic sentences

Atomic sentence  =  \textit{predicate}(term_1, \ldots, term_n)

or \hspace{0.2cm} term_1 = term_2

Term  =  \textit{function}(term_1, \ldots, term_n)

or constant or variable

E.g.,\hspace{0.2cm} Married(Ali, Ayse)
Father(Ahmet) = Ali
Married(Father(Ahmet), Mother(Ahmet))
Complex sentences

Complex sentences are made from atomic sentences using connectives

\[ \neg S, \quad S_1 \land S_2, \quad S_1 \lor S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2 \]

E.g. \( Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn) \)
\[ >(1,2) \lor \leq(1,2) \]

Convention:

\( P(x,y) \) means "\( x \) is a \( P \) of \( y \)" or "\( x \) \( P \) \( y \)". In other words we say Father(fathersname, childsname) rather than the other way around.
Truth in first-order logic

Sentences are true with respect to a model and an interpretation.

Model contains objects and relations among them.

Interpretation specifies referents for:
- constant symbols → objects
- predicate symbols → relations
- function symbols → functional relations

An atomic sentence $\text{predicate}(term_1, \ldots, term_n)$ is true iff the objects referred to by $term_1, \ldots, term_n$ are in the relation referred to by $\text{predicate}$.
Model contains 5 objects, 2 binary, 3 unary relations and 1 unary function.
Consider the interpretation in which

\textit{Richard} → Richard the Lionheart
\textit{John} → the evil King John
\textit{Brother} → the brotherhood relation

Under this interpretation, \textit{Brother}(Richard, John) is true.
Semantics Summary

♦ Under an interpretation, each term refers to an object.

♦ An atomic sentence containing the equality symbol (=) is true iff the two equated terms refer to the same object.

♦ An atomic sentence is true iff the relation referred to by the predicate symbol holds among the objects referred to by the arguments.

\[1 = 2\] may be true in a certain model and interpretation.
Universal and Existential Quantification

Once we have objects, FOL lets us express properties of entire collections of objects.
Universal quantification

∀(variables) (sentence)

Everyone at Sabanci is smart:
∀ x At(x, Sabanci) ⇒ Smart(x)

∀ x P is true in a model m iff P is true with x being each possible object in the model

∀ x P is equivalent to the conjunction of instantiations of P

(At(Ahmet, Sabanci) ⇒ Smart(Ahmet))
∧ (At(Mehmet, Sabanci) ⇒ Smart(Mehmet))
∧ (At(Ayse, Sabanci) ⇒ Smart(Ayse))
∧ ... 

Typically, ⇒ is the main connective with ∀.
A common mistake to avoid

Typically, \( \Rightarrow \) is the main connective with \( \forall \)

Common mistake: using \( \land \) as the main connective with \( \forall \):

\[
\forall x \; At(x, Sabanci) \land Smart(x)
\]

means “Everyone is at Sabanci and everyone is smart”
Existential quantification

\[ \exists \langle \text{variables} \rangle \langle \text{sentence} \rangle \]

Someone at Koc is smart:
\[ \exists x \ At(x, Koc) \land \text{Smart}(x) \]

\[ \exists x \ P \] is true in a model \( m \) iff \( P \) is true with \( x \) being some possible object in the model.

\[ \exists x \ P \] is equivalent to the disjunction of instantiations of \( P \)

\[
(At(Kemal, Koc) \land \text{Smart}(Kemal)) \\
\lor (At(Esra, Koc) \land \text{Smart}(Esra)) \\
\lor (At(Funda, Koc) \land \text{Smart}(Funda)) \\
\lor \ldots
\]
Another common mistake to avoid

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

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

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

is true if there is anyone who is not at Koc!
Properties of quantifiers

\( \forall x \ \forall y \) is the same as \( \forall y \ \forall x \) (why??)

\( \exists x \ \exists y \) is the same as \( \exists y \ \exists x \) (why??)
Properties of quantifiers

\( \forall x \ \forall y \) is the same as \( \forall y \ \forall x \) (why??)

\( \exists x \ \exists y \) is the same as \( \exists y \ \exists x \) (why??)

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

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

... 

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

...
Properties of quantifiers

∀x  ∀y is the same as ∀y  ∀x (why??)

∃x  ∃y is the same as ∃y  ∃x (why??)

∃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” “For everyone, there is someone who loves them”
Properties of quantifiers

\[ \forall x \ \forall y \ \text{is the same as} \ \forall y \ \forall x \ \text{(why??)} \]

\[ \exists x \ \exists y \ \text{is the same as} \ \exists y \ \exists x \ \text{(why??)} \]

\[ \exists x \ \forall y \ \text{is not the same as} \ \forall y \ \exists x \]

\[ \exists x \ \forall y \ Loves(x, y) \]

“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

\[ \forall x \ Likes(x, \text{IceCream}) \ \text{and} \ \neg \exists x \ \neg Likes(x, \text{IceCream}) \]

\[ \exists x \ Likes(x, \text{Broccoli}) \ \text{and} \ \neg \forall x \ \neg Likes(x, \text{Broccoli}) \]
Brothers are siblings
Brothers are siblings

\[ \forall x, y \ \text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y). \]

“Sibling” is symmetric
Brothers are siblings

\[ \forall x, y \; \text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y). \]

“Sibling” is symmetric

\[ \forall x, y \; \text{Sibling}(x, y) \iff \text{Sibling}(y, x). \]

One’s mother is one’s female parent
Brothers are siblings

\[ \forall x, y \; \text{Brother}(x, y) \Rightarrow \text{Sibling}(x, y) . \]

“Sibling” is symmetric

\[ \forall x, y \; \text{Sibling}(x, y) \iff \text{Sibling}(y, x) . \]

One’s mother is one’s female parent

\[ \forall x, y \; \text{Mother}(x, y) \iff (\text{Female}(x) \land \text{Parent}(x, y)) . \]

A first cousin is a child of a parent’s sibling
Brothers are siblings

\[ \forall x, y \; \text{Brother}(x, y) \implies \text{Sibling}(x, y). \]

“Sibling” is symmetric

\[ \forall x, y \; \text{Sibling}(x, y) \iff \text{Sibling}(y, x). \]

One’s mother is one’s female parent

\[ \forall x, y \; \text{Mother}(x, y) \iff (\text{Female}(x) \land \text{Parent}(x, y)). \]

A first cousin is a child of a parent’s sibling

\[ \forall x, y \; \text{FirstCousin}(x, y) \iff \exists p, ps \; \text{Parent}(p, x) \land \text{Sibling}(ps, p) \land \text{Parent}(ps, y) \]
Equality

$\text{term}_1 = \text{term}_2$ is true under a given interpretation if and only if $\text{term}_1$ and $\text{term}_2$ refer to the same object

E.g., $1 = 2$ and $\forall x \times (\text{Sqrt}(x), \text{Sqrt}(x)) = x$ are satisfiable

$2 = 2$ is valid

E.g., definition of (full) Sibling in terms of Parent:

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

\[ \text{Brother}(\text{John}, \text{Richard}) \land \text{Brother}(\text{Geoffrey}, \text{Richard}). \]

- **Unique names assumption**: Every constant symbol refers to a distinct object.
- **Closed world assumption**: Atomic sentences not known to be true are false.
- **Domain closure**: Each model contains no more domain elements than those named by a constant symbol.

Under what is called **database semantics** (above assumptions), the above sentence conveys our knowledge that "Richard has two brothers". This is different than the standard semantics of FOL.
Higher-Order Logic

FOL: one can quantify over objects (first order entities that actually exist in the world)

Higher Order Logic: quantify over relations and functions

e.g. $\forall x, y \ (x = y) \iff (\forall P \ P(x) \iff P(y))$
Using FOL

\[ \text{Tell}(KB, \text{King}(John)) \]
\[ \text{Tell}(KB, \text{Person}(Richard)) \]
\[ \text{Tell}(\forall x \; \text{King}(x) \iff \text{Person}(x)) \].

Now we can ask questions to the knowledgebase using ASK:

\[ \text{Ask}(KB, \text{King}(John)) \] returns True.

\[ \text{Ask}(KB, \text{Person}(John)) \] should also return True.

\[ \text{Ask}(KB, \exists x \; \text{Person}(x)) \] is True, but not very useful if we don’t know who that person is.

\[ \text{AskVars}(KB, \text{Person}(x)) \] returns a substitution in the form \( \{x/John\}, \{x/Richard\} \).
Substitution in FOL

*AskVars* is useful in KB that can be written in Horn form in which every way of making a query true will bind the variables to specific values.

Consider a sentence in FOL which is *not* a Horn close:

\[ \text{King}(John) \lor \text{King}(Richard). \]

Then \( \text{Ask}(KB, \exists x \ \text{King}(x)) \) should return True, but there is no binding to \( x \).
Uniqueness quantifier (there exists only one), sets, \( \lambda \) operator, notational variations (Prolog, Lisp,...)
Exercise with the Kinship Domain

...
What to put in a KB?

**Axioms**: basic facts about a domain (basic predicates)

\[ \forall x, y \ Sibling(x, y) \iff \lnot(x = y) \land \exists p \ (Parent(p, x) \land Parent(p, y)). \]

**Theorems**: facts entailed by axioms (e.g. the theorem below is entailed by definition of the Sibling predicate given above)

\[ \forall x, y \ Sibling(x, y) \iff Sibling(y, x). \]

Theorems are useful to encode important conclusions, to prevent the need to make deductions from scratch.
Not all axioms are definitions.

∀ x \textit{Person}(x) \Leftrightarrow \ldots \text{(difficult!)}

Instead:

∀ x \textit{Person}(x) \Rightarrow \ldots

∀ x \ldots \Rightarrow \textit{Person}(x)
Natural Numbers

From only a few axioms (Peano axioms), we can build up natural numbers (non-negative integers):

\[ \text{NatNum}(0). \]

\[ \forall n \; \text{NatNum}(n) \Rightarrow \text{NatNum}(S(n)), \text{ where } S \text{ is the successor function}. \]

Now we can define addition:

\[ \forall n \; \text{NatNum}(n) \Rightarrow + (0, n) = n. \]

\[ \forall m, n \; \text{NatNum}(m) \land \text{NatNum}(n) \Rightarrow + (S(m), n) = S(+ (m, n)). \]

Using syntactic sugar (here the **infix** notation) for clarity:

\[ \forall m, n \; \text{NatNum}(m) \land \text{NatNum}(n) \Rightarrow (m + 1) + n = (m + n) + 1. \]

The last axiom reduces addition to repeated application of the successor function.
Sets

Sets: Unordered collection of elements, where $\emptyset$ indicate the empty list.

The only sets are the empty set and those made by adjoining/adding something to a set:

$$\forall s \ Set(x) \iff (s = \emptyset) \lor (\exists x, s_2 Set(s_2) \land s = \{x \mid s_2\}).$$

Go over the other axioms.
Lists

Lists: Lists are similar to sets but they are ordered.

We can define Nil (const. list with no elements) and other necessary functions using the Lisp names:

\[ \text{Nil} = [] \].

\[ \forall x, t \ \text{Member}(x, [x|t]). \]

\[ \forall x, y, t \ \text{Member}(x, t) \Rightarrow \text{Member}(x, [y|t]). \]

ex. 8.17 asks you to complete the other axioms for lists.
Knowledge base for the wumpus world - agent and percepts

Percept:

\[
\text{Percept([Smell, Breeze, Glitter, none, none], 5)}
\]

where the percept is represented in a list.

**Actions:** Turn(Right), Turn(Left), Forward,...

To ask for best actions:

\[
\text{ASKVARS(∃a } \text{ BestAction(a, 5))}
\]

which would return a binding list such as \{a/Grab\}. 
Raw percept data implies certain facts about the current state:

\[ \forall b, g, t \ Percept([\text{Smell}, b, g], t) \Rightarrow \text{Smelt}(t) \]

\[ \forall s, b, t \ Percept([s, b, \text{Glitter}], t) \Rightarrow \text{AtGold}(t) \]

Simple reflex behavior can be implemented as:

\[ \forall t \ \text{Glitter}(t) \Rightarrow \text{BestAction}(\text{Grab}, t) \]
Agent’s location changes over time, so we will use:

\[ At(Agent, s, t) \]

Properties of locations (notice time or location dependence):

\[ \forall s, t \ At(Agent, s, t) \land Smelt(t) \Rightarrow Smelly(s) \]
\[ \forall s, t \ At(Agent, s, t) \land Breeze(t) \Rightarrow Breezy(s) \]
Now with the power of FOL, we can represent adjacent squares as:

\[ \forall x, y, a, b \]
\[ \text{Adjacent}([x, y], [a, b]) \iff [a, b] \in \{[x+1, y], [x-1, y], [x, y+1], [x, y-1]\} \]

Then, we could have a rule such as:

\[ \forall x, y \text{  Breeze}([x, y]) \iff \exists a, b \text{  Pit}([a, b]) \land \text{Adjacent}([x, y], [a, b]) \]
Knowledge base for the wumpus world - environment

Diagnostic rule—infer cause from effect
\[ \forall y \ Breezy(y) \Rightarrow \exists x \ Pit(x) \land \text{Adjacent}(x, y) \]

Causal rule—infer effect from cause
\[ \forall x, y \ Pit(x) \land \text{Adjacent}(x, y) \Rightarrow Breezy(y) \]

Neither of these is complete—e.g., the causal rule doesn’t say whether squares far away from pits can be breezy

Definition for the Breezy predicate:
\[ \forall y \ Breezy(y) \Leftrightarrow [\exists x \ Pit(x) \land \text{Adjacent}(x, y)] \]
Suppose a wumpus-world agent is using an FOL KB and perceives a glitter and a breeze at $t = 5$:

$\text{Tell}(KB, \text{Percept}([\text{Glitter, Breeze, None}], 5))$
$\text{Ask}(KB, \exists a \ \text{Action}(a, 5))$

I.e., does $KB$ entail any particular actions at $t = 5$?

Answer: $Yes, \ \{a/\text{Grab}\} \leftarrow \text{substitution} \ (\text{binding list})$

Given a sentence $S$ and a substitution $\sigma$, $S\sigma$ denotes the result of plugging $\sigma$ into $S$; e.g.,

$S = \text{Smarter}(x, y)$
$\sigma = \{x/\text{Hillary}, y/\text{Bill}\}$
$S\sigma = \text{Smarter}(\text{Hillary, Bill})$

$\text{Ask}(KB, S)$ returns some/all $\sigma$ such that $KB \models S\sigma$
Knowledge base for the wumpus world

Simple reflex behaviour can be implemented by quantified implication sentences: Reflex: \( \forall t \ AtGold(t) \Rightarrow Action(Grab, t) \)

Reflex with internal state: do we have the gold already?
\( \forall t \ AtGold(t) \land \neg Holding(Gold, t) \Rightarrow Action(Grab, t) \)

\( Holding(Gold, t) \) cannot be observed
\( \Rightarrow \) keeping track of change is essential
Keeping track of change

Facts hold in situations, rather than eternally
E.g., \(\text{Holding}(\text{Gold, Now})\) rather than just \(\text{Holding}(\text{Gold})\)

Situation calculus is one way to represent change in FOL:
 Adds a situation argument to each non-eternal predicate
E.g., \(\text{Now}\) in \(\text{Holding}(\text{Gold, Now})\) denotes a situation

Situations are connected by the \text{Result} function
\(\text{Result}(a, s)\) is the situation that results from doing \(a\) in \(s\)
Describing actions I

“Effect” axiom—describe changes due to action
∀s AtGold(s) ⇒ Holding(Gold, Result(Grab, s))

“Frame” axiom—describe non-changes due to action
∀s HaveArrow(s) ⇒ HaveArrow(Result(Grab, s))

Frame problem: find an elegant way to handle non-change
(a) representation—avoid frame axioms
(b) inference—avoid repeated “copy-overs” to keep track of state

Qualification problem: true descriptions of real actions require endless caveats—what if gold is slippery or nailed down or . . .

Ramification problem: real actions have many secondary consequences—what about the dust on the gold, wear and tear on gloves, . . .
Describing actions II

Successor-state axioms solve the representational frame problem

Each axiom is “about” a predicate (not an action per se):

\[
P \text{ true afterwards } \iff \begin{cases} 
\text{an action made } P \text{ true} \\
\lor \quad P \text{ true already and no action made } P \text{ false} 
\end{cases}
\]

For holding the gold:

\[
\forall a, s \quad \text{Holding}(\text{Gold}, \text{Result}(a, s)) \iff \\
\quad \begin{cases} 
(a = \text{Grab} \land \text{AtGold}(s)) \\
\lor \quad \text{Holding}(\text{Gold}, s) \land a \neq \text{Release} 
\end{cases}
\]