FIRST-ORDER LOGIC

CHAPTER 7

Outline

◊ Syntax and semantics of FOL
◊ Fun with sentences
◊ Wumpus world in FOL
Syntax of FOL: Basic elements

Constants \( \text{KingJohn, 2, UCB, \ldots} \)
Predicates \( \text{Brother, >, \ldots} \)
Functions \( \text{Sqrt, LeftLegOf, \ldots} \)
Variables \( x, y, a, b, \ldots \)
Connectives \( \land, \lor, \neg, \Rightarrow, \Leftrightarrow \)
Equality \( = \)
Quantifiers \( \forall, \exists \)

Atomic sentences

Atomic sentence \( = \) \( \text{predicate}(\text{term}_1, \ldots, \text{term}_n) \)

or \( \text{term}_1 = \text{term}_2 \)

Term \( = \) \( \text{function}(\text{term}_1, \ldots, \text{term}_n) \)

or constant or variable

E.g., \( \text{Brother}(\text{KingJohn, RichardTheLionheart}) \)
\( > (\text{Length}(\text{LeftLegOf}(\text{Richard})), \text{Length}(\text{LeftLegOf}(\text{KingJohn}))) \)
Complex sentences

Complex sentences are made from atomic sentences using connectives

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

E.g.  \( \text{Sibling}(\text{KingJohn}, \text{Richard}) \Rightarrow \text{Sibling}(\text{Richard}, \text{KingJohn}) \)
\( > (1,2) \lor \leq (1,2) \)
\( > (1,2) \land \neg > (1,2) \)

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 \( \text{predicate}(\text{term}_1, \ldots, \text{term}_n) \) is true iff the objects referred to by \( \text{term}_1, \ldots, \text{term}_n \) are in the relation referred to by \( \text{predicate} \)
Models for FOL: Example

Objects

Relations: sets of tuples of objects

\{ \langle \text{KingJohn}, \text{Berkeley} \rangle, \langle \text{Smart}, \text{Richard} \rangle, \ldots \} 

Functional relations: all tuples of objects + "value" object

\{ \langle \text{KingJohn}, \text{Berkeley} \rangle, \langle \text{Smart}, \text{Richard} \rangle, \ldots \} 

Universal quantification

\( \forall (\text{variables}) \ (\text{sentence}) \)

Everyone at Berkeley is smart:
\( \forall x \ \text{At}(x, \text{Berkeley}) \Rightarrow \text{Smart}(x) \)

\( \forall x \ P \) is equivalent to the conjunction of instantiations of \( P \)

\( \text{At}(\text{KingJohn}, \text{Berkeley}) \Rightarrow \text{Smart}(\text{KingJohn}) \)

\( \land \ \text{At}(\text{Richard}, \text{Berkeley}) \Rightarrow \text{Smart}(\text{Richard}) \)

\( \land \ \text{At}(\text{Berkeley}, \text{Berkeley}) \Rightarrow \text{Smart}(\text{Berkeley}) \ \land \ldots \)

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

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

\( \forall x \ \text{At}(x, \text{Berkeley}) \land \text{Smart}(x) \)

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

\[ \exists (\text{variables}) \ (\text{sentence}) \]
Someone at Stanford is smart:
\[ \exists x \ At(x, Stanford) \land Smart(x) \]
\[ \exists x \ P \] is equivalent to the disjunction of instantiations of \( P \)
\[ At(KingJohn, Stanford) \land Smart(KingJohn) \]
\[ \lor \ At(Richard, Stanford) \land Smart(Richard) \]
\[ \lor \ At(Stanford, Stanford) \land Smart(Stanford) \lor \ldots \]
Typically, \( \land \) is the main connective with \( \exists \).
Common mistake: using \( \Rightarrow \) as the main connective with \( \exists \):
\[ \exists x \ At(x, Stanford) \Rightarrow Smart(x) \]
is true if there is anyone who is not at Stanford!

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 \( \forall y \ \exists x \)
\[ \exists x \ \forall y \ Loves(x, y) \]
"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
\[ \forall x \ Likes(x, IceCream) \quad \neg \exists x \ \neg Likes(x, IceCream) \]
\[ \exists x \ Likes(x, Broccoli) \quad \neg \forall x \ \neg Likes(x, Broccoli) \]
Fun with sentences

Brothers are siblings

"Sibling" is reflexive

One’s mother is one’s female parent

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

∀x, y  Brother(x, y) ⇔ Sibling(x, y).

∀x, y  Sibling(x, y) ⇔ Sibling(y, x)

∀x, y  Mother(x, y) ⇔ (Female(x) and Parent(x, y))

∀x, y  FirstCousin(x, y) ⇔ ∃p, ps  Parent(p, x) ∧ Sibling(ps, p) ∧ 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 \( \forall x \times (\sqrt{x}, \sqrt{x}) = x \) are satisfiable.

\( 2 = 2 \) is valid.

E.g., definition of (full) \textit{Sibling} in terms of \textit{Parent}:

\[
\forall x, y \quad \text{Sibling}(x, y) \iff \neg(x = y) \land \exists m, f \neg(m = f) \land \\
\quad \text{Parent}(m, x) \land \text{Parent}(f, x) \land \text{Parent}(m, y) \land \text{Parent}(f, y)
\]

**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 \( t = 5 \):

\[
\text{TELL}(KB, \text{Percept}([\text{Smell}, \text{Breeze}, \text{None}], 5))
\]

\[
\text{ASK}(KB, \exists a \text{ Action}(a, 5))
\]

I.e., does the KB entail any particular actions at \( t = 5 \)?

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

Given a sentence \( S \) and a substitution \( \sigma \),

\( S\sigma \) denotes the result of plugging \( \sigma \) into \( S \); e.g.,

\[
S = \text{Smarter}(x, y)
\]

\[
\sigma = \{x/\text{Hillary}, y/\text{Bill}\}
\]

\[
S\sigma = \text{Smarter}(\text{Hillary}, \text{Bill})
\]

\[
\text{ASK}(KB, S) \text{ returns some/all } \sigma \text{ such that } KB \models S\sigma
\]
**Knowledge base for the wumpus world**

**“Perception”**
\[ \forall b, g, t \ Percept([\text{Smell}, b, g], t) \Rightarrow \text{Smelt}(t) \]
\[ \forall s, b, t \ Percept([s, b, \text{Glitter}], t) \Rightarrow \text{AtGold}(t) \]

**Reflex:** \( \forall t \ \text{AtGold}(t) \Rightarrow \text{Action(Grab, t)} \)

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

\( \text{Holding(Gold, t)} \) cannot be observed
\[ \Rightarrow \text{keeping track of change is essential} \]

---

**Deducing hidden properties**

Properties of locations:
\[ \forall l, t \ \text{At(Agent, l, t) \land Smelt(t) \Rightarrow Smelly(l)} \]
\[ \forall l, t \ \text{At(Agent, l, t) \land Breeze(t) \Rightarrow Breezy(l)} \]

Squares are breezy near a pit:

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

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

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:**
\[ \forall y \ \text{Breezy}(y) \Leftrightarrow [\exists x \ \text{Pit}(x) \land \text{Adjacent}(x, y)] \]
Keeping track of change

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

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

Describing actions I

“Effect” axiom—describe changes due to action
\( \forall s \quad \text{AtGold}(s) \Rightarrow \text{Holding}(\text{Gold}, \text{Result}(\text{Grab}, s)) \)

“Frame” axiom—describe non-changes due to action
\( \forall s \quad \text{HaveArrow}(s) \Rightarrow \text{HaveArrow}(\text{Result}(\text{Grab}, s)) \)

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):

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

For holding the gold:

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

Making plans

Initial condition in KB:
\[ At(Agent, [1, 1], S_0) \]
\[ At(Gold, [1, 2], S_0) \]

Query: \[ \text{Ask}(KB, \exists s \quad Holding(Gold, s)) \]
\[ \text{i.e., in what situation will I be holding the gold?} \]

Answer: \{ \text{s/Result(Grab, Result(Forward, S_0))}\}
\[ \text{i.e., go forward and then grab the gold} \]

This assumes that the agent is interested in plans starting at \( S_0 \) and that \( S_0 \) is the only situation described in the KB
Making plans: A better way

Represent plans as action sequences \([a_1, a_2, \ldots, a_n]\)

\[ \text{PlanResult}(p, s) \text{ is the result of executing } p \text{ in } s \]

Then the query \( \text{Ask}(KB, \exists p \ \text{Holding}(Gold, \text{PlanResult}(p, S_0))) \)

has the solution \( \{p/\text{[Forward, Grab]}\} \)

Definition of \( \text{PlanResult} \) in terms of \( \text{Result} \):

\[
\forall s \ \text{PlanResult}([], s) = s \\
\forall a, p, s \ \text{PlanResult}([a|p], s) = \text{PlanResult}(p, \text{Result}(a, s))
\]

Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner

---

Summary

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

Increased expressive power: sufficient to define wumpus world

Situation calculus:
- conventions for describing actions and change in FOL
- can formulate planning as inference on a situation calculus KB