First-Order Logic

Chapter 8

Outline

• Why FOL?
• Syntax and semantics of 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 $B_{1,1} \land P_{1,2}$ 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

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, …
FOL Syntax

• Add variables and quantifiers to
  propositional logic

Syntax of FOL: Basic elements

• Constants KingJohn, 2, Pitt,...
• Predicates Brother, >,...
• Functions Sqrt, LeftLegOf,...
• Variables x, y, a, b,...
• Connectives ¬, ⇒, ∧, ∨, ⇔
• Equality =
• Quantifiers ∀, ∃
Atomic sentences

Atomic sentence = \textit{predicate} (term_1,\ldots,term_n)
\text{ or } \text{term}_1 = \text{term}_2

Term = \textit{function} (term_1,\ldots,term_n)
\text{ or } \text{constant or variable}

\begin{itemize}
  \item E.g., \textit{Brother} (KingJohn, RichardTheLionheart)
  \item \texttt{ > (Length(LeftLegOf(Richard)),Length(LeftLegOf(KingJohn)))}
\end{itemize}

Complex sentences

\begin{itemize}
  \item Complex sentences are made from atomic sentences using connectives
    \begin{align*}
      \neg S, S_1 \land S_2, S_1 \lor S_2, S_1 \Rightarrow S_2, S_1 \Leftrightarrow S_2
    \end{align*}
  \end{itemize}

\begin{itemize}
  \item E.g. \textit{Sibling} (KingJohn, Richard) \Rightarrow \textit{Sibling} (Richard, KingJohn)
  \item \texttt{ > (1,2) \lor \leq (1,2)}
  \item \texttt{ > (1,2) \land \neg > (1,2)}
\end{itemize}
Knowledge engineering involves deciding what types of things Should be constants, predicates, and functions for your problem

**Propositional Logic vs FOL**

\[
\text{B23 } \rightarrow \ (P32 \lor P\ 23 \lor P34 \lor P\ 43) \ ...
\]

“Internal squares adjacent to pits are breezy”:

\[
\text{All } X \ Y \ (B(X,Y) \land (X > 1) \land (Y > 1) \land (Y < 4) \land (X < 4)) \iff
\]

\[
(P(X-1,Y) \lor P(X,Y-1) \lor P(X+1,Y) \lor (X,Y+1))
\]
FOL (FOPC) Worlds

- Rather than just T,F, now worlds contain:
- **Objects:** the gold, the wumpus, …
  "the domain"
- **Predicates:** holding, breezy
- **Functions:** sonOf

*Ontological commitment*

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 $\rightarrow$ objects
  - predicate symbols $\rightarrow$ relations
  - function symbols $\rightarrow$ functional relation

**Interpretation:** assignment of elements from the world to elements of the language

- An atomic sentence $\text{predicate(term}_1,\ldots,\text{term}_n)$ is true iff the objects referred to by $\text{term}_1,\ldots,\text{term}_n$ are in the relation referred to by $\text{predicate}$
Quantifiers

• All X p(X) means that p holds for all elements in the domain
•Exists X p(X) means that p holds for at least one element of the domain

Universal quantification

• ∀<variables> <sentence>

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

• ∀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

   At(KingJohn,Pitt) ⇒ Smart(KingJohn)
   ∧ At(Richard,Pitt) ⇒ Smart(Richard)
   ∧ At(Pitt,Pitt) ⇒ Smart(Pitt)
   ∧ ...
A common mistake to avoid

- Typically, ⇒ is the main connective with ∀
- Common mistake: using ∧ as the main connective with ∀:
  \[ \forall x \text{ At}(x, \text{Pitt}) \land \text{Smart}(x) \]
  means “Everyone is at Pitt and everyone is smart”

Existential quantification

- \[ \exists<\text{variables}> <\text{sentence}> \]
- Someone at Pitt is smart:
  \[ \exists x \text{ At}(x, \text{Pitt}) \land \text{Smart}(x) \]
- \[ \exists x \ P \text{ is true in a model } m \text{ iff } P \text{ is true with } x \text{ being some possible object in the model} \]
- Roughly speaking, equivalent to the disjunction of instantiations of \( P \):
  \[ \text{At}(\text{KingJohn}, \text{Pitt}) \land \text{Smart}(\text{KingJohn}) \lor \text{At}(\text{Richard}, \text{Pitt}) \land \text{Smart}(\text{Richard}) \lor \text{At}(\text{Pitt}, \text{Pitt}) \land \text{Smart}(\text{Pitt}) \lor \ldots \]
Another common mistake to avoid

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

• Common mistake: using $\Rightarrow$ as the main connective with $\exists$:
  $\exists x \text{ At}(x,\text{Pitt}) \Rightarrow \text{Smart}(x)$
  is true if there is anyone who is not at Pitt!

Examples

• Everyone likes chocolate
• Someone likes chocolate
• Everyone likes chocolate unless they are allergic to it
Examples

• Everyone likes chocolate
  – $\forall X \text{ person}(X) \rightarrow \text{likes}(X, \text{chocolate})$

• Someone likes chocolate
  – $\exists X \text{ person}(X) \land \text{likes}(X, \text{chocolate})$

• Everyone likes chocolate unless they are allergic to it
  – $\forall X (\text{person}(X) \land \neg \text{allergic}(X, \text{chocolate})) \rightarrow \text{likes}(X, \text{chocolate})$

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 $\forall y \exists x$

• $\exists x \forall y \text{ Loves}(x,y)$
  – “There is a person who loves everyone in the world”

• $\forall y \exists x \text{ Loves}(x,y)$
  – “Everyone in the world is loved by at least one person”
Nesting of Variables

Put quantifiers in front of \( \text{likes}(P,F) \)
Assume the domain of discourse of \( P \) is the set of people
Assume the domain of discourse of \( F \) is the set of foods

1. Everyone likes some kind of food
   \( \forall P \exists F \text{likes}(P,F) \)
2. There is a kind of food that everyone likes
   \( \exists F \forall P \text{likes}(P,F) \)
3. Someone likes all kinds of food
   \( \exists P \forall F \text{likes}(P,F) \)
4. Every food has someone who likes it
   \( \forall F \exists P \text{likes}(P,F) \)

Answers

(DOD of \( P \) is people and \( F \) is food)

Everyone likes some kind of food
   \( \forall P \exists F \text{likes}(P,F) \)
There is a kind of food that everyone likes
   \( \exists F \forall P \text{likes}(P,F) \)
Someone likes all kinds of food
   \( \exists P \forall F \text{likes}(P,F) \)
Every food has someone who likes it
   \( \forall F \exists P \text{likes}(P,F) \)
Answers, without Domain of Discourse Assumptions

Everyone likes some kind of food
\[ \text{All } P \ \text{person}(P) \rightarrow \text{Exists } F \ \text{food}(F) \text{ and } \text{likes}(P,F) \]

There is a kind of food that everyone likes
\[ \text{Exists } F \ \text{food}(F) \text{ and } (\text{All } P \ \text{person}(P) \rightarrow \text{likes}(P,F)) \]

Someone likes all kinds of food
\[ \text{Exists } P \ \text{person}(P) \text{ and } (\text{All } F \ \text{food}(F) \rightarrow \text{likes}(P,F)) \]

Every food has someone who likes it
\[ \text{All } F \ \text{food}(F) \rightarrow \text{Exists } P \ \text{person}(P) \text{ and } \text{likes}(P,F) \]

Quantification and Negation

- \( \neg(\forall x \ p(x)) \equiv \exists x \ \neg p(x) \)
- \( \neg(\exists x \ p(x)) \equiv \forall x \ \neg p(x) \)

- **Quantifier duality**: each can be expressed using the other
  - \( \forall x \ \text{Likes}(x,\text{IceCream}) \rightarrow \neg \exists x \ \neg \text{Likes}(x,\text{IceCream}) \)
  - \( \exists x \ \text{Likes}(x,\text{Broccoli}) \rightarrow \neg \forall x \ \neg \text{Likes}(x,\text{Broccoli}) \)
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 $Parent$:
  \[
  \forall x, y \ Sibling(x, y) \iff \neg(x = y) \land \exists m, f \neg(m = f) \land \\
  \quad Parent(m, x) \land Parent(f, x) \land Parent(m, y) \land Parent(f, y) \]

Representational Schemes

• What are the objects, predicates, and functions? Keep in mind that you need to encode knowledge of specific problem instances and general knowledge.

• In practice, consider interpretations just to understand what the choices are. The world and interpretation are defined, or at least constrained, through the logical sentences we write.
Example Choice: Predicates versus Constants

• Rep-Scheme 1: Let’s consider the world: \( D = \{a,b,c,d,e\} \). \( \text{green}: \{a,b,c\} \). \( \text{blue}: \{d,e\} \). Some sentences that are satisfied by the intended interpretation:

\[
\text{green}(a). \quad \text{green}(b). \quad \text{blue}(d). \\
\neg(\text{All } x \text{ green}(x)). \quad \text{All } x \text{ green}(x) \lor \text{blue}(x).
\]

But what if we want to say that blue is pretty?

Choice: Predicates versus Constants

• Rep-Scheme 2: The world: \( D = \{a,b,c,d,e,\text{green,blue}\} \)
  \( \text{colorof}: \{<a,\text{green}>,<b,\text{green}>,<c,\text{green}>,<d,\text{blue}>,<e,\text{blue}>\} \)
  \( \text{pretty}: \{\text{blue}\} \quad \text{notprimary}: \{\text{green}\} \)

• Some sentences that are satisfied by the intended interpretation:

\[
\text{colorOf}(a,\text{green}). \quad \text{colorOf}(b,\text{green}). \quad \text{colorOf}(d,\text{blue}). \\
\neg(\text{All } X \text{ colorOf}(X,\text{green})). \\
\text{All } X \text{ colorOf}(X,\text{green}) \lor \text{colorOf}(X,\text{blue}). \\
***\text{pretty}(\text{blue}). \quad \text{notprimary}(\text{green}).***
\]

We have reified predicates \text{blue} and \text{green}: made them into objects.
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

Summary

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

• Increased expressive power: e.g., better to define wumpus world