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 $B_{1,1} \wedge P_{1,2}$ 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 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	$\neg$, $\Rightarrow$, $\wedge$, $\vee$, $\Leftrightarrow$
• 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*

- E.g., *Brother(KingJohn,RichardTheLionheart)*
- *> (Length(LeftLegOf(Richard)),Length(LeftLegOf(KingJohn)))*

Complex sentences

- Complex sentences are made from atomic sentences using connectives

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

E.g. *Sibling(KingJohn,Richard)* $\Rightarrow$ *Sibling(Richard,KingJohn)*

$>(1,2) \vee \leq (1,2)$

$>(1,2) \wedge \neg >(1,2)$

Sentence → AtomicSentence |
(Sentence Connective Sentence) |
Quantifier Variable, .. Sentence |
~Sentence
AtomicSentence → Predicate(Term,…) | Term = Term
Term → Function(Term,…) |
Constant |
Variable
Connective → → | ^ | v | ←→
Quantifier → all, exists
Constant → john, 1, …
Variable → A, B, C, X
Predicate → breezy, sunny, red
Function → fatherOf, plus

Knowledge engineering involves deciding what types of things
Should be constants, predicates, and functions for your problem

Propositional Logic vs FOL

B23 → (P32 v P 23 v P34 v P 43) …

"Internal squares adjacent to pits are breezy":

All X Y (B(X,Y) ^ (X > 1) ^ (Y > 1) ^ (Y < 4) ^ (X < 4)) ←→

(P(X-1,Y) v P(X,Y-1) v P(X+1,Y) v (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	→	objects
predicate symbols	→	relations
function symbols	→	functional relation

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

- 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*

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

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

Everyone at Pitt is smart:
$\forall$x At(x,Pitt) $\Rightarrow$ 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*

At(KingJohn,Pitt) $\Rightarrow$ Smart(KingJohn)
$\wedge$ At(Richard,Pitt) $\Rightarrow$ Smart(Richard)
$\wedge$ *At(Pitt,Pitt) $\Rightarrow$ Smart(Pitt)*
$\wedge$ ...

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,Pitt) $\wedge$ Smart(x)

  means "Everyone is at Pitt and everyone is smart"

Existential quantification

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

Another common mistake to avoid

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

  $$\exists x \text{ At}(x,\text{Pitt}) \Rightarrow \text{Smart}(x)$$

  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
  - $\forall X$ person(X) → likes(X, chocolate)
- Someone likes chocolate
  - $\exists X$ person(X) ^ likes(X, chocolate)
- Everyone likes chocolate unless they are allergic to it
  - $\forall X$ (person(X) ^ ¬allergic (X, chocolate)) → likes(X, 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 $\forall y\ \exists x$
- $\exists x\ \forall y$ 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"

Nesting of Variables

Put quantifiers in front of 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
2. There is a kind of food that everyone likes
3. Someone likes all kinds of food
4. Every food has someone who likes it

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

Everyone likes some kind of food
  All P Exists F likes(P,F)
There is a kind of food that everyone likes
  Exists F All P likes(P,F)
Someone likes all kinds of food
  Exists P All F likes(P,F)
Every food has someone who likes it
  All F Exists P likes(P,F)

Answers, without Domain of Discourse Assumptions

Everyone likes some kind of food

All P person(P) → Exists F food(F) and likes(P,F)

There is a kind of food that everyone likes

Exists F food(F) and (All P person(P) → likes(P,F))

Someone likes all kinds of food

Exists P person(P) and (All F food(F) → likes(P,F))

Every food has someone who likes it

All F food (F) → Exists P person(P) and likes(P,F)

Quantification and Negation

- ~($\forall$x p(x)) equiv $\exists$x ~p(x)
- ~($\exists$x p(x)) equiv $\forall$x ~p(x)

- 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

- E.g., definition of *Sibling* in terms of *Parent*:
  $\forall x,y\ Sibling(x,y) \Leftrightarrow [\neg(x = y) \wedge \exists m,f\ \neg (m = f) \wedge$
  $Parent(m,x) \wedge Parent(f,x) \wedge Parent(m,y) \wedge Parent(f,y)]$

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}. green: {a,b,c}. blue: {d,e}. Some sentences that are satisfied by the intended interpretation:

green(a). green(b). blue(d).

~(All x green(x)). All x green(x) v 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,green,blue}
  colorof:
  {<a,green>,<b,green>,<c,green>,<d,blue>,<e,blue>}
  pretty: {blue} notprimary: {green}

- Some sentences that are satisfied by the intended interpretation:

colorOf(a,green). colorOf(b,green). colorOf(d,blue).

~(All X colorOf(X,green)).

All X colorOf(X,green) v colorOf(X,blue).

pretty(blue). notprimary(green).

We have reified predicates blue and 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