First-Order Logic

Dr. Melanie Martin
CS 4480

Based on slides from
http://aima.eecs.berkeley.edu/2nd-ed/slides-ppt/

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

- 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, … (squares, pits, wumpuses)
  – Relations: red, round, prime, brother of, bigger than, part of, comes between, … (is breezy, is adjacent to, shoots)
  – Functions: father of, best friend, one more than, plus, …

Propositional Logic vs FOL

$B_{23} \rightarrow (P_{32} \vee P_{23} \vee P_{34} \vee P_{43})$ …

"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))

Syntax of FOL: Basic elements

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

BNF Grammar on p 293

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

Detour: Some British History

- Richard the Lionheart
  - **Richard I** (8 September 1157 - 6 April 1199) was King of England from 6 July 1189 until his death in 1199.
  - Rebelled unsuccessfully against father Henry II
  - Spoke very little English and mostly lived in Aquitaine
  - Was a central Christian commander during the Third Crusade
  - In his absence, brother John tries to seize throne, Richard forgives him

Atomic sentences

Atomic sentence = *predicate* ($term_1$,...,$term_n$)
| $term_1$ = $term_2$

Term = *function* ($term_1$,...,$term_n$)
| *constant* | *variable*

- *Brother(John, Richard)*
- *Married(Father(Richard), Mother(John))*

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)* ⇒ *Sibling(Richard,KingJohn)*
$>(1,2) \vee \leq (1,2)$
$>(1,2) \wedge \neg >(1,2)$

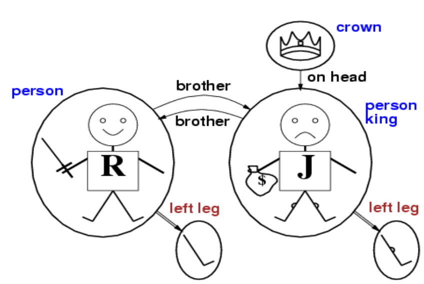
Truth in first-order logic

- 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*

Models for FOL: Example



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 CSU is smart:
$\forall x\ At(x,CSU) \Rightarrow Smart(x)$

- $\forall x\ P$ 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*

  $At(KingJohn,CSU) \Rightarrow Smart(KingJohn)$
  $\wedge\ At(Richard,CSU) \Rightarrow Smart(Richard)$
  $\wedge\ At(CSU,CSU) \Rightarrow Smart(CSU)$
  $\wedge\ \ldots$

A common mistake to avoid

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

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

  means "Everyone is at CSU and everyone is smart"

Existential quantification

- $\exists$<*variables*> <*sentence*>
- Someone at CSU is smart:
- $\exists x\ At(x,CSU) \wedge Smart(x)$
- $\exists x\ P$ 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*

  $At(KingJohn,CSU) \wedge Smart(KingJohn)$
  $\vee\ At(Richard,CSU) \wedge Smart(Richard)$
  $\vee\ At(CSU,CSU) \wedge Smart(CSU)$
  $\vee\ \ldots$

Another common mistake to avoid

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

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

  is true if there is anyone who is not at CSU!

  Transform to:
  $\exists x\ \sim(At(x,CSU))\ \vee\ Smart(x)$

Examples

- Everyone likes chocolate
- Someone likes chocolate
- Everyone likes chocolate unless they are allergic to it

Examples

- Everyone likes chocolate
  - ∀X person(X) → likes(X, chocolate)
- Someone likes chocolate
  - ∃X person(X) ^ likes(X, chocolate)
- Everyone likes chocolate unless they are allergic to it
  - ∀X (person(X) ^ ¬allergic (X, chocolate)) → likes(X, chocolate)

Properties of quantifiers

- ∀x ∀y is the same as ∀y ∀x
- ∃x ∃y is the same as ∃y ∃x
- ∃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"

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
∀ P, ∃ F likes(P,F)
There is a kind of food that everyone likes
∃ F, ∀ P likes(P,F)
Someone likes all kinds of food
∃ P, ∀ F likes(P,F)
Every food has someone who likes it
∀ F, ∃ P likes(P,F)

Answers, without Domain of Discourse Assumptions

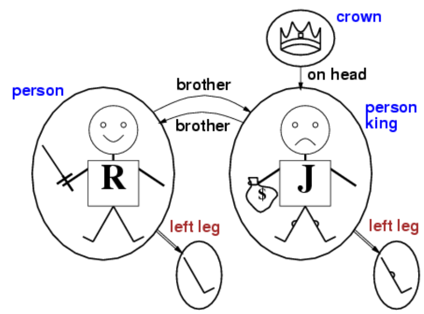
Everyone likes some kind of food
∀ P person(P) → ∃ F food(F) and likes(P,F)
There is a kind of food that everyone likes
∃ F food(F) and (∀ P person(P) → likes(P,F))
Someone likes all kinds of food
∃ P person(P) and (∀ F food(F) → likes(P,F))
Every food has someone who likes it
∀ F food (F) → ∃ P person(P) and likes(P,F)

Quantification and Negation

- ¬∃x p(x) equiv ∀x ¬p(x)
  - ¬∃x likes(x, parsnips)
  - ∀x ¬likes(x, parsnips)
- ¬∀x p(x) equiv ∃x ¬p(x)
  - ¬∀x likes(x, parsnips)
  - ∃x ¬likes(x, parsnips)
- Quantifier duality: each can be expressed using the other
- ∀x Likes(x,IceCream) ¬∃x ¬Likes(x,IceCream)
- ∃x Likes(x,Broccoli) ¬∀x ¬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*:
  $\forall x,y\ Sibling(x,y) \Leftrightarrow [\neg(x = y) \wedge \exists m,f\ \neg\ (m = f) \wedge Parent(m,x) \wedge Parent(f,x) \wedge Parent(m,y) \wedge Parent(f,y)]$



- Predicate of brotherhood:
  - {<R,J>,<J,R>}
- Predicate of being on: {<C,J>}
- Predicate of being a person:
  - {J,R}
- Predicate of being the king: {J}
- Predicate of being a crown: {C}
- Function for left legs: <{J,JLL},{R,RLL}>

Interpretation

- Specifies which objects, functions, and predicates are referred to by which constant symbols, function symbols, and predicate symbols.
- Under the intended interpretation:
- "richardI" refers to R; "johnII" refers to J; "crown" refers to the crown.
- "onHead","brother","person","king", "crown", "leftLeg", "strong"

Lots of other possible interpretations

- 5 objects, so just for constants "richard" and "john" there are 25 possibilities
- Note that the legs don't have their own names!
- "johnII" and "johnLackland" may be assigned the same object, J
- Also possible: "crown" and "johnII" refer to C (just not the intended interpretation)

Why isn't the "intended interpretation" enough?

- Vague notion. What is intended may be ambiguous (and often is, for non-toy domains)
- Logically possible: square(x) ^ round(x). Your KB has to include knowledge that rules this out.

Determining truth values of FOPC sentences

- Assign meanings to terms:
  - "johnII" ← J; "leftLeg(johnII)" ← JLL
- Assign truth values to atomic sentences
  - "brother(johnII,richardI)"
  - "brother(johnlackland,richardI)"
  - Both True, because <J,R> is in the set assigned "brother"
  - "strong(leftleg(johnlackland))"
  - True, because JLL is in the set assigned "strong"

Examples given the Sample Interpretation

- ∀ X,Y brother(X,Y) FALSE
- ∀ X,Y (person(X) ^ person(Y)) → brother(X,Y) FALSE
- ∀ X,Y (person(X) ^ person(Y) ^ ~(X=Y)) → brother(X,Y) TRUE
- ∃ X crown(X) TRUE
- ∃ X ∃ Y sister(X,Y) FALSE

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).

~(∀ x green(x)). ∀ 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).

~(∀ X colorOf(X,green)).

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

pretty(blue). notprimary(green).

We have reified predicates blue and green: made them into objects

Using FOL

The kinship domain:

- Brothers are siblings
  ∀x,y *Brother(x,y)* ⇔ *Sibling(x,y)*
- One's mother is one's female parent
  ∀m,c *Mother(c)* = m ⇔ *(Female(m)* ∧ *Parent(m,c))*
- "Sibling" is symmetric
  ∀x,y *Sibling(x,y)* ⇔ *Sibling(y,x)*

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 ⊢ Sσ

Knowledge base for the wumpus world

- Perception
  – ∀t,s,b Percept([s,b,Glitter],t) ⇒ Glitter(t)
- Reflex
  – ∀t Glitter(t) ⇒ BestAction(Grab,t)

Deducing hidden properties

- ∀x,y,a,b *Adjacent*([x,y],[a,b]) ⇔
  [a,b] ∈ {[x+1,y], [x-1,y],[x,y+1],[x,y-1]}

Properties of squares:

- ∀s,t *At*(Agent,s,t) ∧ Breeze(t) ⇒ Breezy(s)

Squares are breezy near a pit:

- Diagnostic rule---infer cause from effect
  ∀s Breezy(s) ⇒ ∃ r, Adjacent(r,s) ∧ Pit(r)
- Causal rule---infer effect from cause
  ∀r Pit(r) ⇒ [∀s Adjacent(r,s) ⇒ 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

Summary

- First-order logic:
  – objects and relations are semantic primitives
  – syntax: constants, functions, predicates, equality, quantifiers
- Increased expressive power: better to define wumpus world