First-Order Logic

Outline

- Why FOL?
- Syntax and semantics of FOL
- Using FOL
- Wumpus world in FOL
- Knowledge engineering in FOL

2

These slides are based on the slides at AIMA book webpage

Pros and cons of propositional logic

- ☺ Propositional logic is declarative
  - Contrast with procedural approach (Programming languages C++, Java)
  - knowledge and inference are separate, inference is entirely domain-independent
- ☺ 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

3

These slides are based on the slides at AIMA book webpage

First-order logic

- Whereas propositional logic assumes the world contains facts,
  - PL : facts hold or do not hold.
- 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 : objects with relations between them that hold or do not hold

4

These slides are based on the slides at AIMA book webpage

Syntax of FOL: Basic elements

- Constants John, 2, ISU,...
- Predicates Brother, Teacher, >,...
- Functions Sqrt, LeftLegOf,...
- Variables x, y, a, b,...
- Connectives $\neg$, $\Rightarrow$, $\wedge$, $\vee$, $\Leftrightarrow$
- Equality =
- Quantifiers $\forall$, $\exists$

5

These slides are based on the slides at AIMA book webpage

Atomic sentences

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

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

(A term refers to an object)

- E.g., *Brother(KingJohn,RichardTheLionheart)*

Length(LeftLegOf(Richard))=Length(LeftLegOf(KingJohn))

6

These slides are based on the slides at AIMA book webpage

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

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

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

7

These slides are based on the slides at AIMA book webpage

Complex sentences

- Universal quantification

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

  Everyone at ISU is smart:

  $\forall$x At(x,ISU) $\Rightarrow$ Smart(x)

- Existential quantification

  $\exists$<*variables*> <*sentence*>

  Someone at ISU is smart:

  $\exists x$ At(x, ISU) $\land$ Smart(x)

8

These slides are based on the slides at AIMA book webpage

Truth in first-order logic

- Sentences are true with respect to a model
- Model contains objects (called domain elements) and an interpretation of symbols
- Interpretation specifies referents for
  constant symbol → object in domain D
  predicate symbol → relation
  function symbol → functional relation
- Each predicate symbol of arity k is mapped to a relation {<object1, ..., objectk>}, a set of k-tuples over D^k which are true or, equivalently, a function from D^k to {*true*,*false*}
- Each function symbol of arity k is mapped to a function from D^k to D+1 (domain + an invisible object)

9

These slides are based on the slides at AIMA book webpage

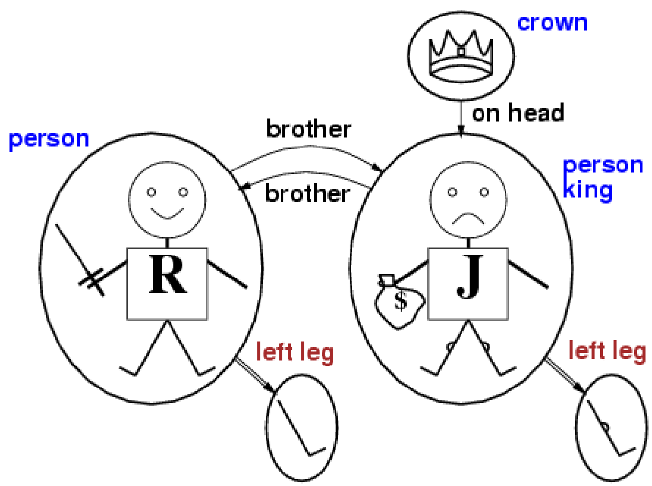
Truth in first-order logic

- An atomic sentence *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 *predicate*.
- $term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object
- The semantics of sentences formed with logical connectives is identical to that in PL

10

These slides are based on the slides at AIMA book webpage

Models for FOL: Example



11

These slides are based on the slides at AIMA book webpage

Models for FOL: Example

- The intended interpretation
  - Constant Richard → Richard the Lionheart
  - Constant John → King John
  - Predicate Brother → the brotherhood relation
    {< Richard the Lionheart, King John>, <King John, Richard the Lionheart>}
  - Predicates OnHead, Person, King, Crown …
  - Function Leftleg ->
    <Richard the Lionheart> → Richard's left leg
    < King John> → John's left leg

- Another interpretation
  - Constant Richard → the crown
  - Constant John → King John's left leg
  - Predicate Brother →
    {< Richard the Lionheart, the crown>}
  - …

12

These slides are based on the slides at AIMA book webpage

Models for FOL

- Entailment can be defined
- We can enumerate the models for a given KB vocabulary:

For each number of domain elements n from 1 to ∞
  For each k-ary predicate P_k in the vocabulary
    For each possible k-ary relation on n objects
      For each constant symbol C in the vocabulary
        For each choice of referent for C from n objects . . .

- Computing entailment by enumerating the models will not be easy !!
  - the number of possible models may be unbounded or very large (10^{25} for the example)

13

These slides are based on the slides at AIMA book webpage

Quantifiers

- Allows us to express properties of collections of objects instead of enumerating objects by name

- Universal: “for all” $\forall$
- Existential: “there exists” $\exists$

14

These slides are based on the slides at AIMA book webpage

Universal quantification

∀<*variables*> <*sentence*>

Everyone at ISU is smart:

$\forall x\ At(x,ISU) \Rightarrow Smart(x)$

$\forall x\ P$ 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*

$At(KingJohn,ISU) \Rightarrow Smart(KingJohn)$
$\land\ At(Richard, ISU) \Rightarrow Smart(Richard)$
$\land\ At(ISU,ISU) \Rightarrow Smart(ISU)$
$\land\ \ldots$

A common mistake to avoid

- Typically, $\Rightarrow$ is the main connective with $\forall$

- Common mistake: using $\land$ as the main connective with $\forall$:

  $\forall x\ At(x, ISU) \land Smart(x)$

  means “Everyone is at ISU and everyone is smart”

These slides are based on the slides at AIMA book webpage

Existential quantification

∃<*variables*> <*sentence*>

Someone at ISU is smart:
$\exists x$ At(x, ISU) $\wedge$ 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,ISU) $\wedge$ Smart(KingJohn)
  $\vee$ At(Richard,ISU) $\wedge$ Smart(Richard)
  $\vee$ At(ISU, ISU) $\wedge$ Smart(ISU)
  $\vee$ ...

Another common mistake to avoid

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

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

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

These slides are based on the slides at AIMA book webpage

Properties of quantifiers

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

- 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)

19

These slides are based on the slides at AIMA book webpage

Using FOL

The kinship domain:
Predicates *Parent, Brother, Sibling, Child, Son, Wife*, *Cousin*, ...

- 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))*
  (use Mother(m,c)?)
- "Sibling" is symmetric

20

These slides are based on the slides at AIMA book webpage

Using FOL

The kinship domain:

- Brothers are siblings
  $\forall x,y\ Brother(x,y) \Rightarrow Sibling(x,y)$
- One's mother is one's female parent
  $\forall m,c\ Mother(c) = m \Leftrightarrow (Female(m) \wedge Parent(m,c))$
- "Sibling" is symmetric
  $\forall x,y\ Sibling(x,y) \Leftrightarrow Sibling(y,x)$
- A first cousin is a child of a parent's sibling

Using FOL

The kinship domain:

- Brothers are siblings
  $\forall x,y\ Brother(x,y) \Rightarrow Sibling(x,y)$
- One's mother is one's female parent
  $\forall m,c\ Mother(c) = m \Leftrightarrow (Female(m) \wedge Parent(m,c))$
- "Sibling" is symmetric
  $\forall x,y\ Sibling(x,y) \Leftrightarrow Sibling(y,x)$
- A first cousin is a child of a parent's sibling
  $\forall x,y\ FirstCousin(x,y) \Leftrightarrow \exists p,ps\ Parent(p,x) \wedge Sibling(ps,p) \wedge Parent(ps,y)$
- Definition of full *Sibling* in terms of *Parent*:

Using FOL

The kinship domain:

- Brothers are siblings
  $\forall x,y\ Brother(x,y) \Rightarrow Sibling(x,y)$
- One's mother is one's female parent
  $\forall m,c\ Mother(c) = m \Leftrightarrow (Female(m) \wedge Parent(m,c))$
- “Sibling” is symmetric
  $\forall x,y\ Sibling(x,y) \Leftrightarrow Sibling(y,x)$
- A first cousin is a child of a parent’s sibling
  $\forall x,y\ FirstCousin(x,y) \Leftrightarrow \exists p,ps\ Parent(p,x) \wedge Sibling(ps,p) \wedge Parent(ps,y)$

- Definition of full *Sibling* in terms of *Parent*:
  $\forall x,y\ FullSibling(x,y) \Leftrightarrow [\neg(x = y) \wedge \exists m,f\ \neg(m = f) \wedge Parent(m,x) \wedge Parent(f,x) \wedge Parent(m,y) \wedge Parent(f,y)]$

Using FOL - The set domain

The set theory can be built up from a tiny kernel of axioms

Constant { }, unary predicate *Set*
Syntactic sugar: an extension to the standard syntax to make sentences easier to read

- Binary functions: $\{x|s\}$, $s_1 \cap s_2$, $s_1 \cup s_2$
- Binary predicates: $x \in s$, $s_1 \subseteq s_2$

- $\forall s\ Set(s) \Leftrightarrow (s = \{\}) \vee (\exists x,s_2\ Set(s_2) \wedge [s = \{x|s_2\}])$
- $\neg\exists x,s\quad \{x|s\} = \{\}$
- $\forall x,s\quad x \in s \Leftrightarrow s = \{x|s\}$
- $\forall x,s\quad x \in s \Leftrightarrow [\exists y,s_2\ (s = \{y|s_2\} \wedge (x = y \vee x \in s_2))]$
- $\forall s_1,s_2\quad s_1 \subseteq s_2 \Leftrightarrow (\forall x\ \ x \in s_1 \Rightarrow x \in s_2)$
- $\forall s_1,s_2\quad (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$
- $\forall x,s_1,s_2\quad x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$
- $\forall x,s_1,s_2\quad x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$

Using FOL

The list domain (in Lisp vocabulary):

Constants: Empy list *Nil*,
Predicates: *List, ...*
Functions: *Cons, Append, First, Rest*
Syntactic sugar: []=Nil, [x|s] = *Cons*(x,s), [x], [A,B,C]

- List([])
- ∀x,l List(l) ⇔ List([x|l])
- ∀x,l First([x|l]) = x
- ∀x,l Rest([x|l]) = l
- ∀l Append([],l) = l
- ...

Example Knowledge for wumpus world

- Use list term to represent the square objects (or Square(x,y)) instead of naming each square
  ∀x,y,a,b *Adjacent*([x,y],[a,b]) ⇔
  (x=a ∧ (y=b-1 ∨ y=b+1)) ∨(y=b ∧ (x=a-1 ∨ x=a+1))
- constant *Wumpus*
- function *Home(Wumpus)* to name the one square with wumpus
- Unary Predicate *Pit(), Breezy(), Smelly()*
- Squares are breezy near a pit:

Example Knowledge for wumpus world

- Unary Predicate *Pit(), Breezy(), Smelly()*
- Squares are breezy near a pit:
  - Diagnostic rule---infer cause from effect
    $\forall s\ Breezy(s) \Leftrightarrow \exists\ r\ Adjacent(r,s) \wedge Pit(r)$
  - Causal rule---infer effect from cause
    $\forall r\ Pit(r) \Rightarrow [\forall s\ Adjacent(r,s) \Rightarrow Breezy(s)]$
    (not complete!)
    $\forall s\ [\forall r\ Adjacent(r,s) \Rightarrow \neg Pit(r)] \Rightarrow \neg\ Breezy(s)$

27

These slides are based on the slides at AIMA book webpage

Example Knowledge for wumpus world

- Typical percept sentence:
  Percept([Stench,Breeze,Glitter,None,None],5)
- Actions:
  Turn(Right), Turn(Left), Forward, Shoot, Grab
- Perception
  - $\forall b,g,t,m,c\ Percept([Stench,b,g,m,c],t) \Rightarrow Smell(t)$
  - $\forall s,b,t,m,c\ Percept([s,b,Glitter,m,c],t) \Rightarrow Glitter(t)$
- Properties of locations:
  $\forall s,t\ \mathit{At}(Agent,s,t) \wedge Smell(t) \Rightarrow Smelly(s)$
  $\forall s,t\ \mathit{At}(Agent,s,t) \wedge Breeze(t) \Rightarrow Breezy(s)$
- Reflex behavior
  - $\forall t\ Glitter(t) \Rightarrow BestAction(Grab,t)$

28

These slides are based on the slides at AIMA book webpage

Knowledge engineering in FOL

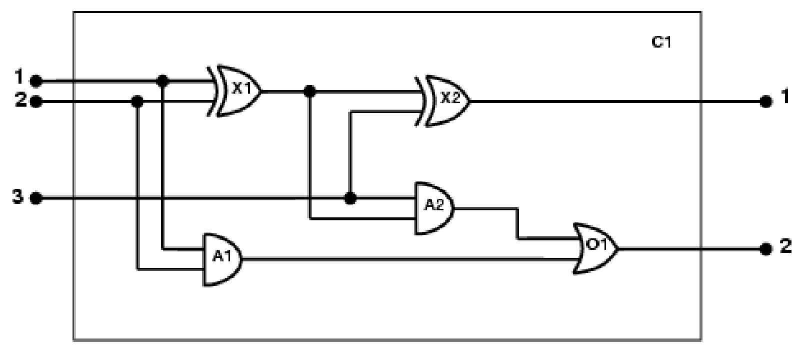
1. Identify the task (what will the KB be used for)
2. Assemble the relevant knowledge
   Knowledge acquisition.
3. Decide on a vocabulary of predicates, functions, and constants
   Translate domain-level knowledge into logic-level names.
4. Encode general knowledge about the domain
   define axioms
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

29

These slides are based on the slides at AIMA book webpage

The electronic circuits domain

One-bit full adder



30

These slides are based on the slides at AIMA book webpage

The electronic circuits domain

1. Identify the task
   - Does the circuit actually add properly? (circuit verification)

2. Assemble the relevant knowledge
   - Composed of wires and gates;
   - Types of gates (AND, OR, XOR, NOT)
   - Connections between terminals
   - Irrelevant: size, shape, color, cost of gates

31

These slides are based on the slides at AIMA book webpage

The electronic circuits domain

3. Decide on a vocabulary
   - Constants: X1, X2, ..., AND, OR, ..., 1, 0
   - Functions: Type(X1), In(2,X1), Out(1,X1), Signal(Out(1,X1))
   - Alternatives:
     $Type(X_1)$ = XOR (encode that a gate can have one type)
     $Type(X_1$, XOR)
     $XOR(X_1)$
   - Predicates: Connected(Out(1,X1), In(2,X2))

32

These slides are based on the slides at AIMA book webpage

The electronic circuits domain

4. Encode general knowledge of the domain

- $\forall t_1,t_2\ Connected(t_1, t_2) \Rightarrow Signal(t_1) = Signal(t_2)$
- $\forall t \quad Signal(t) = 1 \vee Signal(t) = 0$
- $1 \neq 0$
- $\forall t_1,t_2\ Connected(t_1, t_2) \Rightarrow Connected(t_2, t_1)$
- $\forall g\ Type(g)=OR \Rightarrow [Signal(Out(1,g))=1 \Leftrightarrow \exists n\ Signal(In(n,g)) =1]$
- $\forall g\ Type(g)=AND \Rightarrow [Signal(Out(1,g))=0 \Leftrightarrow \exists n\ Signal(In(n,g))=0]$
- $\forall g\ Type(g) = XOR \Rightarrow [Signal(Out(1,g)) = 1 \Leftrightarrow Signal(In(1,g)) \neq Signal(In(2,g))]$
- $\forall g\ Type(g) = NOT \Rightarrow Signal(Out(1,g)) \neq Signal(In(1,g))$

The electronic circuits domain

5. Encode the specific problem instance

$Type(X_1) = XOR$ $\quad Type(X_2) = XOR$
$Type(A_1) = AND$ $\quad Type(A_2) = AND$
$Type(O_1) = OR$

$Connected(Out(1,X_1),In(1,X_2))$ $\quad Connected(In(1,C_1),In(1,X_1))$
$Connected(Out(1,X_1),In(2,A_2))$ $\quad Connected(In(1,C_1),In(1,A_1))$
$Connected(Out(1,A_2),In(1,O_1))$ $\quad Connected(In(2,C_1),In(2,X_1))$
$Connected(Out(1,A_1),In(2,O_1))$ $\quad Connected(In(2,C_1),In(2,A_1))$
$Connected(Out(1,X_2),Out(1,C_1))$ $\quad Connected(In(3,C_1),In(2,X_2))$
$Connected(Out(1,O_1),Out(2,C_1))$ $\quad Connected(In(3,C_1),In(1,A_2))$

The electronic circuits domain

6. Pose queries to the inference procedure

   What are the possible sets of values of all the terminals for the adder circuit?

   $\exists i_1,i_2,i_3,o_1,o_2$ *Signal(In(1,C_1)) =* $i_1 \wedge$ *Signal(In(2,*C_1*)) =* $i_2 \wedge$ *Signal(In(3,*C_1*)) =* $i_3 \wedge$ *Signal(Out(1,*C_1*)) =* $o_1 \wedge$ *Signal(Out(2,*C_1*)) =* o_2

7. Debug the knowledge base

   May have omitted assertions like $1 \neq 0$

35

These slides are based on the slides at AIMA book webpage

Summary

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

36

These slides are based on the slides at AIMA book webpage