Propositional Logic

넘어간다 ~

First-Order Logic

Jihoon Yang

Machine Learning Research Laboratory
Department of Computer Science
Sogang University

Pros and Cons of Propositional Logic

- 😊 Propositional logic is **declarative**: pieces of syntax correspond to facts
- 😊 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 world contains **facts**,
first-order logic (like natural language) assumes the world contains

- Objects: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries . . .
- Relations: red, round, bogus, prime, multistoried . . .,
  brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, . . .
- Functions: father of, best friend, third inning of, one more than, end of . . .

FOL: objects with relations between them that hold or do not hold

Syntax of FOL: Basic Elements

Constants	$KingJohn,\ 2,\ UCB, \ldots$
Predicates	$Brother,\ >, \ldots$
Functions	$Sqrt,\ LeftLegOf, \ldots$
Variables	$x,\ y,\ a,\ b, \ldots$
Connectives	$\wedge\ \ \vee\ \ \neg\ \ \Rightarrow\ \ \Leftrightarrow$
Equality	$=$
Quantifiers	$\forall\ \exists$

Atomic Sentences

$$\begin{aligned}\text{Atomic sentence} \;&=\; predicate(term_1, \ldots, term_n)\\ &\quad\;\text{or } term_1 = term_2\end{aligned}$$

$$\begin{aligned}\text{Term} \;&=\; function(term_1, \ldots, term_n)\\ &\quad\;\text{or } constant \text{ or } variable\end{aligned}$$

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

Complex Sentences

Complex sentences are made from atomic sentences using connectives

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

E.g. $Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn)$
$>(1,2) \vee \leq(1,2)$
$>(1,2) \wedge \neg{>}(1,2)$

Quantifiers

- Universal quantification
  $\forall < variables > < sentence >$
  E.g. Everyone at Sogang is smart: $\forall x\, At(x, Sogang) \Rightarrow Smart(x)$
- Existential quantification
  $\exists < variables > < sentence >$
  E.g. Someone at Sogang is smart: $\exists x\, At(x, Sogang) \wedge Smart(x)$

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* $\rightarrow$ *object in domain D*
  - *predicate symbol* $\rightarrow$ *relation*
  - *function symbol* $\rightarrow$ *functional relation*
- Each predicate symbol of arity k is mapped to a relation $\{< object_1, \cdots, object_k >\}$, 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) → return 값이 없어도 뭔가를 반환하게는 해야 되서

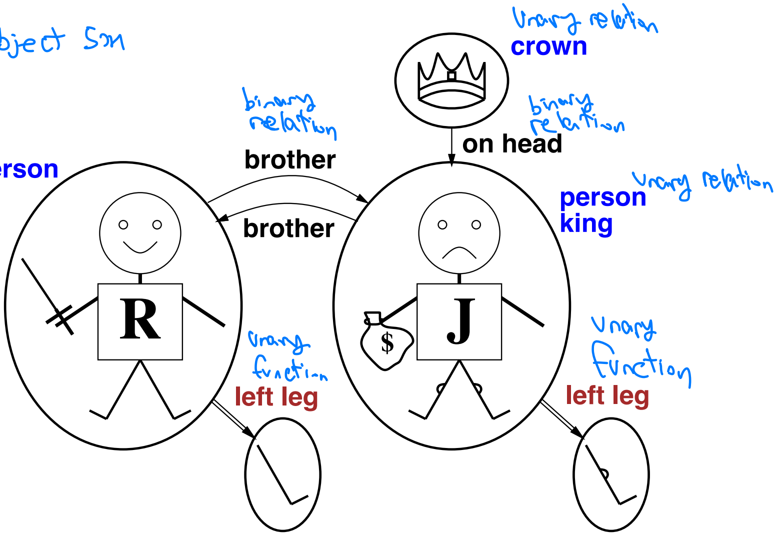
Truth in First-order Logic

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

Models for FOL: Example

O 친 : object 5개



Models for FOL: Example

- The intended interpretation
  - Constant *Richard* $\rightarrow$ Richard the Lionheart
  - Constant *John* $\rightarrow$ King John
  - Predicate *Brother* $\rightarrow$ the brotherhood relation
    $\{< Richard\ the\ Lionheart, King\ John >,$
    $< King\ John, Richard\ the\ Lionheart >\}$
  - Predicates *OnHead*, *Person*, *King*, *Crown*, $\cdots$
  - Function *LeftLeg*
    $< Richard\ the\ Lionheart > \rightarrow$ Richard's left leg
    $< King\ John > \rightarrow$ John's left leg
- Another interpretation
  - Constant *Richard* $\rightarrow$ the crown
  - Constant *John* $\rightarrow$ King John's left leg
  - Predicate *Brother* $\rightarrow$
    $\{< Richard\ the\ Lionheart, the\ crown >\}$
  - ...

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 ∞ objects

  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

Quantifiers

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

- Universal: "for all" $\forall$
- Existential: "there exists" $\exists$

Universal Quantification

$\forall \langle variables \rangle \ \ \langle sentence \rangle$

Everyone at Berkeley is smart:
$\forall x \ \ At(x, Berkeley) \ \Rightarrow \ Smart(x)$

$\forall x \ \ P$ 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

$$
\begin{aligned}
& (At(KingJohn, Berkeley) \ \Rightarrow \ Smart(KingJohn)) \\
\land \ & (At(Richard, Berkeley) \ \Rightarrow \ Smart(Richard)) \\
\land \ & (At(Berkeley, Berkeley) \ \Rightarrow \ Smart(Berkeley)) \\
\land \ & \ldots
\end{aligned}
$$

A Common Mistake to Avoid

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

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

$$\forall x \;\; At(x, Berkeley) \wedge Smart(x)$$

means "Everyone is at Berkeley and everyone is smart"

Existential Quantification

$\exists \langle variables \rangle \;\; \langle sentence \rangle$

Someone at Stanford is smart:
$\exists x \;\; At(x, Stanford) \land Smart(x)$

$\exists x \;\; P$ 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

$$
\begin{aligned}
& (At(KingJohn, Stanford) \land Smart(KingJohn)) \\
\lor \; & (At(Richard, Stanford) \land Smart(Richard)) \\
\lor \; & (At(Stanford, Stanford) \land Smart(Stanford)) \\
\lor \; & \ldots
\end{aligned}
$$

Another Common Mistake to Avoid

Typically, $\wedge$ is the main connective with $\exists$

Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists\, x \;\; At(x, Stanford) \;\Rightarrow\; Smart(x)$$

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??) $\forall_{x,y}$

$\exists x \ \exists y$ is the same as $\exists y \ \exists x$ (why??) $\exists_{x,y}$

$\exists x \ \forall y$ 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

$\forall x \ \ Likes(x, IceCream) \qquad \neg\exists x \ \ \neg Likes(x, IceCream)$

$\exists x \ \ Likes(x, Broccoli) \qquad \neg\forall x \ \ \neg Likes(x, Broccoli)$

Using FOL

숙제.

Brothers are siblings

$\forall x, y \;\; Brother(x, y) \Rightarrow Sibling(x, y)$.

"Sibling" is symmetric

$\forall x, y \;\; Sibling(x, y) \;\Leftrightarrow\; Sibling(y, x)$.

One's mother is one's female parent

$\forall x, y \;\; Mother(x, y) \;\Leftrightarrow\; (Female(x) \wedge Parent(x, y))$.

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

$\forall x, y \;\; FirstCousin(x, y) \;\Leftrightarrow\; \exists p, ps \;\; Parent(p, x) \wedge Sibling(ps, p) \wedge Parent(ps, y)$

Example Knowledge for Wumpus World

- Use list term to represent the square objects (or $Square(x, y)$) instead of naming each square

$$\forall x, y, a, b\ Adjacent([x, y], [a, b]) \Leftrightarrow$$

$$(x = a \wedge (y = b - 1 \vee y = b + 1)) \vee (y = b \wedge (x = a - 1 \vee x = a + 1))$$

- Constant *Wumpus*
- Function $Home(Wumpus)$ to name the one square with wumpus
- Unary predicate $Pit(), Breezy(), Smelly()$

Example Knowledge for Wumpus World

- Squares are breezy near a pit:
  - **Diagnostic** rule: infer cause from effect

$$\forall s\ Breezy(s) \Rightarrow \exists r\ Adjacent(r, s) \wedge Pit(r)$$

  - **Causal** rule: infer effect from cause

$$\forall r, s\ Pit(r) \wedge Adjacent(r, s) \Rightarrow Breezy(s)$$

- 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:

$$\forall s\ Breezy(s) \Leftrightarrow \exists r\ Adjacent(r, s) \wedge Pit(r)$$

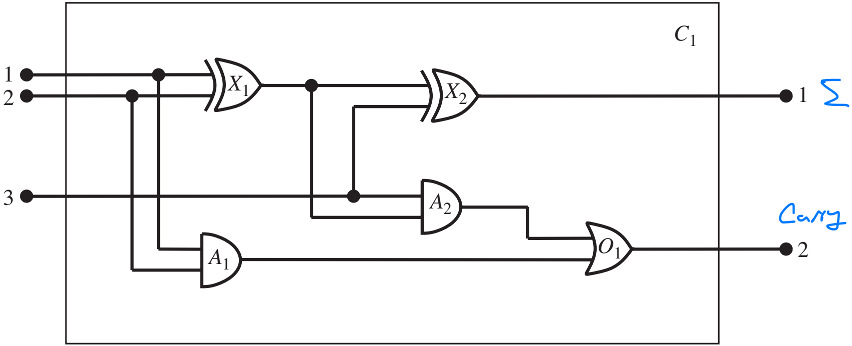
Example Knowledge for Wumpus World

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

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 (*ontology*)
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

One-bit full adder



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

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))

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)
   - $\forall t$ 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))

이 것을 위해
위에서
1≠0을
해줘야 한다.

5. Encode the specific problem instance

Type(X_1) = XOR
Type(A_1) = AND
Type(O_1) = OR

Type(X_2) = XOR
Type(A_2) = AND

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

Connected$(In(1,C_1),In(1,X_1))$
Connected$(In(1,C_1),In(1,A_1))$
Connected$(In(2,C_1),In(2,X_1))$
Connected$(In(2,C_1),In(2,A_1))$
Connected$(In(3,C_1),In(2,X_2))$
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 $1 \neq 0$

Summary

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