First-Order Logic

Russell and Norvig Chapter 8

Propositional logic

☺ Propositional logic is declarative

☺ 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 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

- Examples of things we can say:
  - All men are mortal:
    $\forall x\ Man(x) \Rightarrow Mortal(x)$
  - Everybody loves somebody
    $\forall x\ \exists y\ Loves(x, y)$
  - The meaning of the word “above”
    $\forall x\ \forall y\ above(x,y) \Leftrightarrow (on(x,y) \vee \exists z\ (on(x,z) \wedge above(z,y))$

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, …
  - Relations: red, round, prime, brother of, bigger than, part of, …
  - Functions: father-of, plus, …

Logics in General

- Ontological Commitment: What exists in the world — TRUTH
  - PL : facts hold or do not hold.
  - FOL : objects with relations between them that hold or do not hold
- Epistemoligical Commitment: state of knowledge allowed with respect to a fact

Language	Ontological Commitment	Epistemological Commitment
Propositional logic	facts	true/false/unknown
First-order logic	facts, objects, relations	true/false/unknown
Temporal logic	facts, objects, relations, times	true/false/unknown
Probability theory	facts	degree of belief $\in [0,1]$
Fuzzy logic	degree of truth $\in [0,1]$	known interval value

Syntax of FOL

- User defines these primitives:
  - Constant symbols (i.e., the "individuals" in the world) E.g., Mary, 3
  - Function symbols (mapping individuals to individuals) E.g., father-of(Mary) = John, color-of(Sky) = Blue
  - Predicate/relation symbols (mapping from individuals to truth values) E.g., greater(5,3), green(Grass), color(Grass, Green)

Syntax (cont.)

- FOL supplies these primitives:
  - Variable symbols. E.g., x,y
  - Connectives. Same as in PL: ¬, ⇒, ∧, ∨, ⇔
  - Equality =
  - Quantifiers: Universal (∀) and Existential (∃)

Atomic sentences

Atomic sentence = *predicate* (*term*$_1$,...,*term*$_n$)
or *term*$_1$ = *term*$_2$

Term = *function* (*term*$_1$,...,*term*$_n$)
or *constant* or *variable*

Examples:

Brother(KingJohn,RichardTheLionheart)

Greater(Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))

Complex sentences

Complex sentences are made from atomic sentences using connectives

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

and by applying quantifiers.

Examples:

Sibling(KingJohn,Richard) ⇒ *Sibling(Richard,KingJohn)*

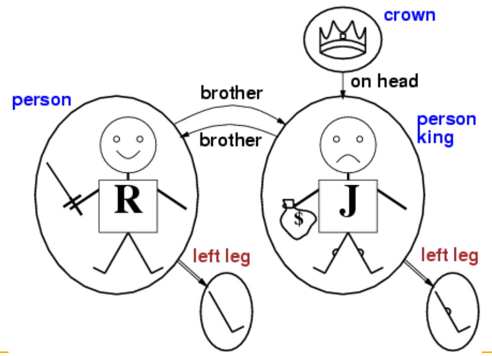
greater(1,2) ∨ less-or-equal(1,2)

∀x,y Sibling(x,y) ⇒ Sibling(y,x)

Truth in first-order logic

- Need to specify
  constant symbols → objects
  predicate symbols → relations
  function symbols → functional relations
- 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



Models for FOL

- 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 !!

Quantifiers

- Allow us to express properties of collections of objects instead of enumerating objects by name
- Universal: “for all” ∀
- Existential: “there exists” ∃

Universal quantification

∀<variables> <sentence>

Everyone at CSU is smart:

∀x At(x, CSU) ⇒ Smart(x)

- ∀x P is true in a model m iff P is true with x being each object in the model
- Roughly speaking, equivalent to the conjunction of instantiations of P

  At(KingJohn,CSU) ⇒ Smart(KingJohn)
  ∧ At(Richard,CSU) ⇒ Smart(Richard)
  ∧ At(CSU,CSU) ⇒ Smart(CSU)
  ∧ ...

Using universal quantifiers

- Typically ⇒ is the main connective with ∀
- Do not make the following mistake:
  ∀x At(x, CSU) ∧ Smart(x)
  means “Everyone is at CSU and everyone is smart”

Existential quantification

∃<variables> <sentence>

Someone at CSU is smart:

∃x At(x, CSU) ∧ Smart(x)

- ∃x P is true in a model m iff P is true with x being some object in the model
- Roughly speaking, equivalent to the disjunction of instantiations of P

  At(KingJohn,CSU) ∧ Smart(KingJohn)
  ∨ At(Richard, CSU) ∧ Smart(Richard)
  ∨ At(CSU, CSU) ∧ Smart(CSU)
  ∨ ...

Existential quantification (cont.)

- Typically, ∧ is the main connective with ∃
- Common mistake: using ⇒ with ∃:

  ∃x At(x, CSU) ⇒ Smart(x)

  When is this true?

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"

- 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 if and only if $term_1$ and $term_2$ refer to the same object
- E.g., definition of *Sibling* in terms of *Parent*:

  ∀*x,y Sibling(x,y)* ⇔ [¬(x = y) ∧ ∃m,f ¬(m = f) ∧
  Parent(m,x) ∧ Parent(f,x) ∧ Parent(m,y) ∧ Parent(f,y)]

Interacting with FOL KBs

- Suppose a wumpus-world agent is using a FOL KB and perceives a smell and a breeze (but no glitter) at position [i,j]:

  `Tell`(KB,Percept([Smell,Breeze,NoGlitter],[i,j])) (= assertion)
  `Ask`(KB,∃a BestAction(a,[i,j]))
  (=query)
  i.e., does the KB entail some best action?

- Answering yes without the best action is not very helpful so:
  Answer: *Yes*, {*a/Shoot*} : substitution (binding list)

- `Ask`(KB, α) returns some/all s such that KB entails SUBST(s, α).

KB for wumpus world

- Perception
  - $\forall$ t,s,g,m,c Percept([s,Breeze,g,m,c],t) $\Rightarrow$ Breeze(t)
- Action
  - $\forall$t Glitter(t) $\Rightarrow$ BestAction(Grab, t)

The wumpus world

Squares are breezy near a pit:

- First define the concept of adjacency:
  $\forall x,y,a,b\ \mathit{Adjacent}([x,y],[a,b]) \Leftrightarrow$
  $(x = a \land (y=b-1 \lor y=b+1)) \lor (y=b \land (x=a-1 \lor x=a+1))$
- Represent time with additional parameter
  At(Agent,s,t) means Agent at square s and time t
- Infer properties
  $\forall s,t\ \ At(Agent,s,t) \land Breeze(t) \Rightarrow Breezy(s)$

How would we say that there is a single wumpus?

Creating a KB using 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
5. Encode a description of the specific problem instance
6. Pose queries to the inference procedure and get answers
7. Debug the knowledge base

Examples

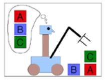
The *kinship* domain

- Basic predicates: Female, Parent…

Other predicates in this domain:

- One's mother is one's female parent
  $\forall m,c\ (Mother(c) = m) \Leftrightarrow (Female(m) \land Parent(m,c))$
- This means?
  $\forall x,y\ Sibling(x,y) \Leftrightarrow [\neg(x = y) \land \exists p\ Parent(p,x) \land Parent(p,y)]$
- These are the axioms of the domain (they are also definitions since they use biconditionals).
- Some sentences are "theorems" -- they can be derived from the axioms:
  - "Sibling" is symmetric
    $\forall x,y\ Sibling(x,y) \Leftrightarrow Sibling(y,x)$

Examples (cont)



The ***natural numbers*** domain

- 0 is a natural number:
  $NatNum(0)$
- The successor of a natural number is a natural number:
  $\forall n\ NatNum(n) \Rightarrow NatNum(S(n))$
- Constraints on the successor function:
  $\forall n\ \neg(0 = S(n))$
  $\forall m,n\ m\ \neg(m = n) \Rightarrow \neg(S(m) = S(n))$
- Defining addition:
  $\forall n\ NatNum(n) \Rightarrow +(0, n) = n$
  $\forall m,n\ NatNum(m) \wedge NatNum(n) \Rightarrow +(S(m), n) = S(+(m,n))$