FIRST ORDER LOGIC

Outline

- Why FOL?
- Syntax and semantics of FOL
- Fun with sentences
- Wumpus world in FOL

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 $B_{1,1} \wedge P_{1,2}$ 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 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 …

Logics in General

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
Fuzzy logic	facts + degree of truth	known interval value

Syntax of FOL: Basic Elements

- Constants
  - KingJohn, 2, UCB, …
- Predicates
  - Brother, >, …
- Functions
  - Sqrt, LeftLegOf, …
- Variables
  - x, y, a, b, …
- Connectives
  - $\wedge$ $\vee$ $\neg$ $\Rightarrow$ $\Leftrightarrow$
- 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)))

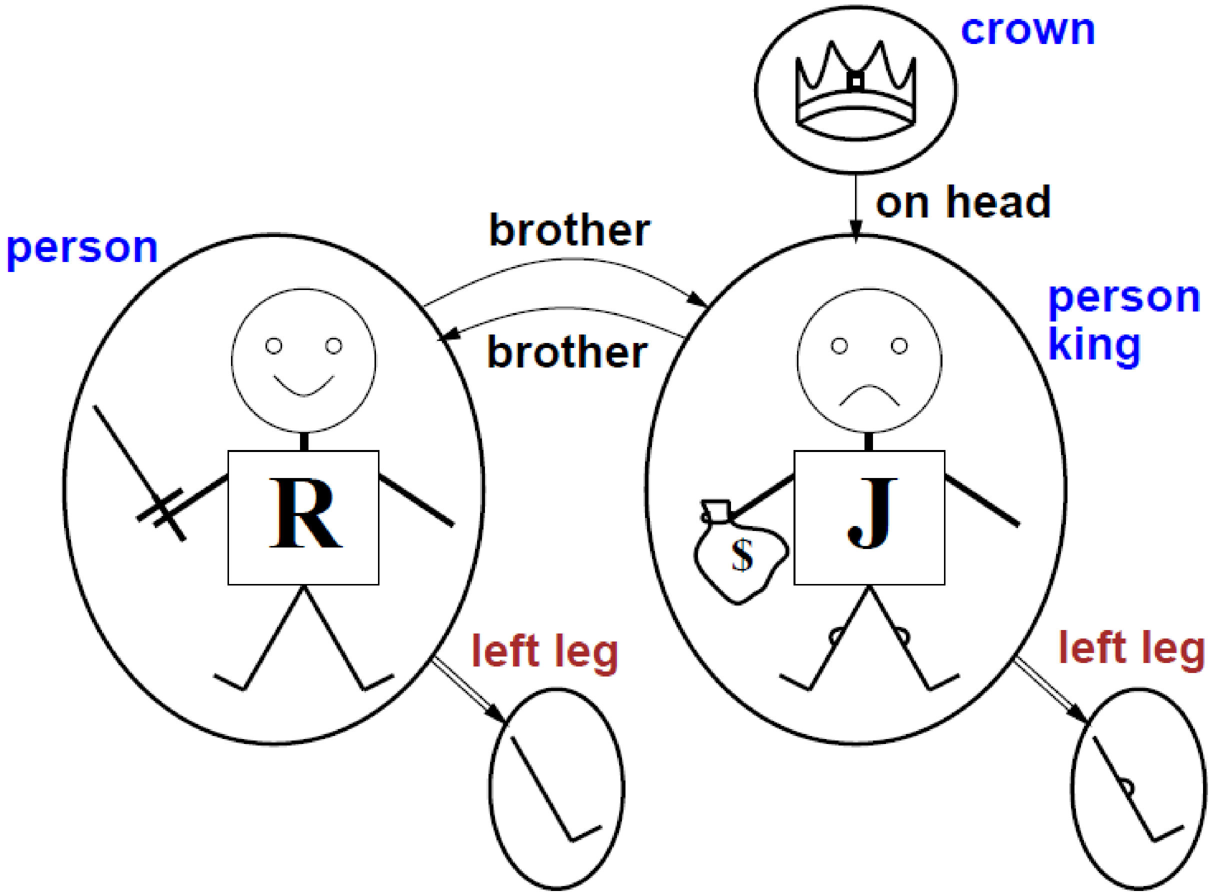
Complex Sentences

- Complex sentences are made from atomic sentences using connectives
  - $\neg S$, $S1 \wedge S2$, $S1 \vee S2$, $S1 \Rightarrow S2$, $S1 \Leftrightarrow S2$
- E.g.
  - $Sibling(KingJohn, Richard)\ ) \Rightarrow Sibling(Richard, KingJohn)$
  - $>(1, 2) \vee \leq (1, 2)$
  - $>(1, 2) \wedge \neg >(1, 2)$

Truth in First-Order Logic

- Sentences are true with respect to a model and an interpretation
- Model contains ≥ 1 objects (domain elements) and relations among them
- Interpretation:
  - 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



Truth Example

- Consider the interpretation in which
  - Richard → Richard the Lionheart
  - John → the evil King John
  - Brother → the brotherhood relation
- Under this interpretation, Brother(Richard, John) is true just in case Richard the Lionheart and the evil King John are in the brotherhood relation in the model

Models for FOL: Lots!

- Entailment in propositional logic can be computed by enumerating models
- We can enumerate the FOL 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 FOL models is not easy!

Universal Quantification

- ∀ <variables> <sentence>

- Everyone at MontanaTech is smart:

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

- ∀ 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, MontanaTech)) ⇒ Smart(KingJohn))

  ∧ (At(Richard, MontanaTech)) ⇒ Smart(Richard))

  ∧ (At(MontanaTech, MontanaTech)) ⇒ Smart(MontanaTech))

  ∧ …

A common Mistake to Avoid

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

  $\forall x\ At(x, MontanaTech) \wedge Smart(x)$
- means "Everyone is at MontanaTech and everyone is smart"

Existential Quantification

- ∃ <variables> <sentence>

- Someone at MSU is smart:

  $\exists x\ At(x, MSU) \land Smart(x)$
- $\exists x\ P$ 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

  (At(KingJohn, MSU) ∧ Smart(KingJohn))

  ∨ (At(Richard, MSU) ∧ Smart(Richard))

  ∨ (MSU, MSU) ∧ MSU))

  ∨ …

Another Common Mistake to Avoid

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

  ∃ x At(x; MSU) ⇒ Smart(x)
- is true if there is anyone who is not at MSU!

Properties of Quantifiers

- $\forall x\ \forall y$ is the same as $\forall y\ \forall x$ (why??)
- $\exists x\ \exists y$ is the same as $\exists y\ \exists x$ (why??)
- $\exists x\ \forall y$ 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)
  $\neg\exists x\ \neg$Likes(x, IceCream)

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

Fun with Sentences

- Brothers are siblings

Fun with Sentences

- Brothers are siblings

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

- “Sibling” is symmetric

Fun with Sentences

- 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

Fun with Sentences

- 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

Fun with Sentences

- 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

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

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.,
  - 1 = 2 and ∀x Times(Sqrt(x), Sqrt(x)) = x are satisfiable
  - 2 = 2 is valid
- E.g., definition of (full) 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 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 Action(a, 5))
- I.e., does KB entail any particular actions 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 $\models$ S

Knowledge Base for the Wumpus World

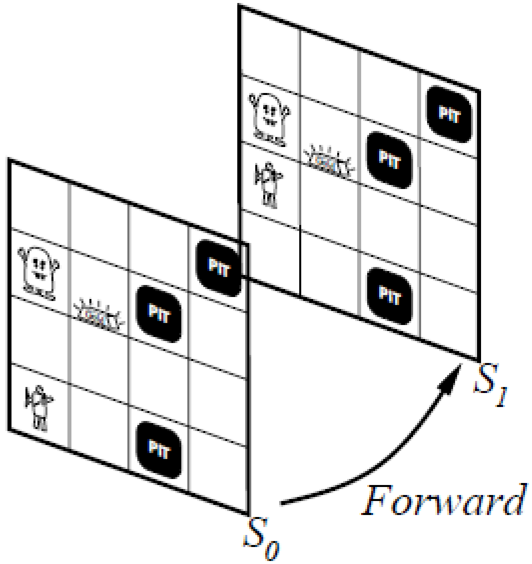
- "Perception"
  - $\forall b,g,t$ Percept([Smell, b, g], t) $\Rightarrow$ Smelt(t)
  - $\forall s,b,t$ Percept([s, b, Glitter], t) $\Rightarrow$ AtGold(t)
- Reflex:
  - $\forall t$ AtGold(t) $\Rightarrow$ Action(Grab, t)
- Reflex with internal state: do we have the gold already?
  - $\forall t$ AtGold(t) $\wedge$ $\neg$Holding(Gold, t) $\Rightarrow$ Action(Grab, t)
- Holding(Gold, t) cannot be observed
  - $\Rightarrow$ keeping track of change is essential

Deducing Hidden Properties

- Properties of locations:
  - $\forall x,t\ At(Agent, x, t) \land Smelt(t) \Rightarrow Smelly(x)$
  - $\forall x,t\ At(Agent, x, t) \land Breeze(t) \Rightarrow Breezy(x)$
- Squares are breezy near a pit:
  - Diagnostic rule - infer cause from effect
    - $\forall y\ Breezy(y) \Rightarrow \exists x\ Pit(x) \land Adjacent(x, y)$
  - Causal rule - infer effect from cause
    - $\forall x,y\ Pit(x) \land Adjacent(x, y) \Rightarrow Breezy(y)$
- 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:
  - $\forall y\ Breezy(y) \Leftrightarrow [\exists x\ Pit(x) \land Adjacent(x, y)]$

Keeping Track of Change

- Facts hold in situations, rather than eternally
  - E.g., Holding(Gold, Now) rather than just Holding(Gold)
- Situation calculus is one way to represent change in FOL:
  - Adds a situation argument to each non-eternal predicate
  - E.g., Now in Holding(Gold, Now) denotes a situation
- Situations are connected by the Result function Result(a, s) is the situation that results from doing a in s



Describing Actions

- “Effect” axiom - describe changes due to action
  - $\forall s$ AtGold(s) $\Rightarrow$ Holding(Gold, Result(Grab, s))
- “Frame” axiom - describe non-changes due to action
  - $\forall s$ HaveArrow(s) $\Rightarrow$ HaveArrow(Result(Grab, s))
- Frame problem: find an elegant way to handle non-change
  - (a) representation - avoid frame axioms
  - (b) inference - avoid repeated “copy-overs” to keep track of state
- Qualification problem: true descriptions of real actions require endless caveats - what if gold is slippery or nailed down or …
- Ramification problem: real actions have many secondary consequences - what about the dust on the gold, wear and tear on gloves, …

Describing Actions

- Successor-state axioms solve the representational frame problem
- Each axiom is “about” a predicate (not an action per se):
  - P true afterwards $\Leftrightarrow$ [an action made P true $\vee$ P true already and no action made P false]
- For holding the gold:
  - $\forall a,s$ Holding(Gold,Result(a, s)) $\Leftrightarrow$ [(a=Grab $\wedge$ AtGold(s)) $\vee$ (Holding(Gold, s) $\wedge$ a $\neq$ Release)]

Making Plans

- Initial condition in KB:
  - At(Agent, [1, 1], S_0)
  - At(Gold, [1, 2], S_0)
- Query: Ask(KB, ∃s Holding(Gold, s))
  - i.e., in what situation will I be holding the gold?
- Answer: {s/Result(Grab,Result(Forward, S_0))}
  - i.e., go forward and then grab the gold
- This assumes that the agent is interested in plans starting at S_0 and that S_0 is the only situation described in the KB

Making Plans: A Better Way

- Represent plans as action sequences $[a_1, a_2, \ldots, a_n]$
- PlanResult(p, s) is the result of executing p in s
- Then the query Ask(KB, $\exists$p Holding(Gold,PlanResult(p, S_0))) has the solution {p/[Forward, Grab]}
- Definition of PlanResult in terms of Result:
  - $\forall$s PlanResult([ ], s) = s
  - $\forall$a,p,s PlanResult([a|p], s) = PlanResult(p,Result(a, s))
- Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner

Summary

- First-order logic:
  - Objects and relations are semantic primitives
  - Syntax: constants, functions, predicates, equality, quantifiers
- Increased expressive power: sufficient to define wumpus world
- Situation calculus:
  - Conventions for describing actions and change in FOL
  - Can formulate planning as inference on a situation calculus KB

An Exercise

- Write in first-order logic the assertion that every key and at least one of every pair of socks will eventually be lost forever, using only the following vocabulary:
- Key(x)
  - x is a key
- Sock(x)
  - x is a sock
- Pair(x,y)
  - x and y are a pair
- Now
  - the current time
- Before(t_1, t_2)
  - time t_1 comes before time t_2
- Lost(x,t)
  - object x is lost at time t