First-Order Logic

Contents

First-Order Logic

- AKA First-Order Predicate Logic
- AKA First-Order Predicate Calculus

- Much more powerful the propositional (Boolean) logic
  - Greater expressive power than propositional logic
    - We no longer need a separate rule for each square to say which other squares are breezy/pits
  - Allows for facts, objects, and relations
    - In programming terms, allows classes, functions and variables

Pros and Cons of Propositional Logic

- + Propositional logic is declarative: pieces of syntax correspond to facts
- + Propositional logic allows for partial / disjunctive / negated information (unlike most data structures and DB
- + Propositional logic is compositional: the meaning of B_{11} ^ P_{12} is derived from the meaning of B_{11} and P_{12}
- + Meaning of propositional logic is context independent: (unlike natural language, where the meaning depends on the 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

Pros of First-Order Logic

- First-Order Logic assumes that the world contains:
  - Objects
    - E.g. people, houses, numbers, theories, colors, football games, wars, centuries, ...
  - Relations
    - E.g. red, round, prime, bogus, multistoried, brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, ...
  - Functions
    - E.g. father of, best friend, third quarter of, one more than, beginning of, ...

Logics in General

Language	**Ontological Commitment**	**Epistemological Commitment**
Propositional Logic	Facts	True / False / Unknown
First-Order Logic	Fact, 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 First-Order Logic

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

Components of First-Order Logic

- Term
  - Constant, e.g. Red
  - Function of constant, e.g. Color(Block1)

- Atomic Sentence
  - Predicate relating objects (no variable)
    - Brother (John, Richard)
    - Married (Mother(John), Father(John))

- Complex Sentences
  - Atomic sentences + logical connectives
    - Brother (John, Richard) $\wedge\neg$Brother (John, Father(John))

Components of First-Order Logic

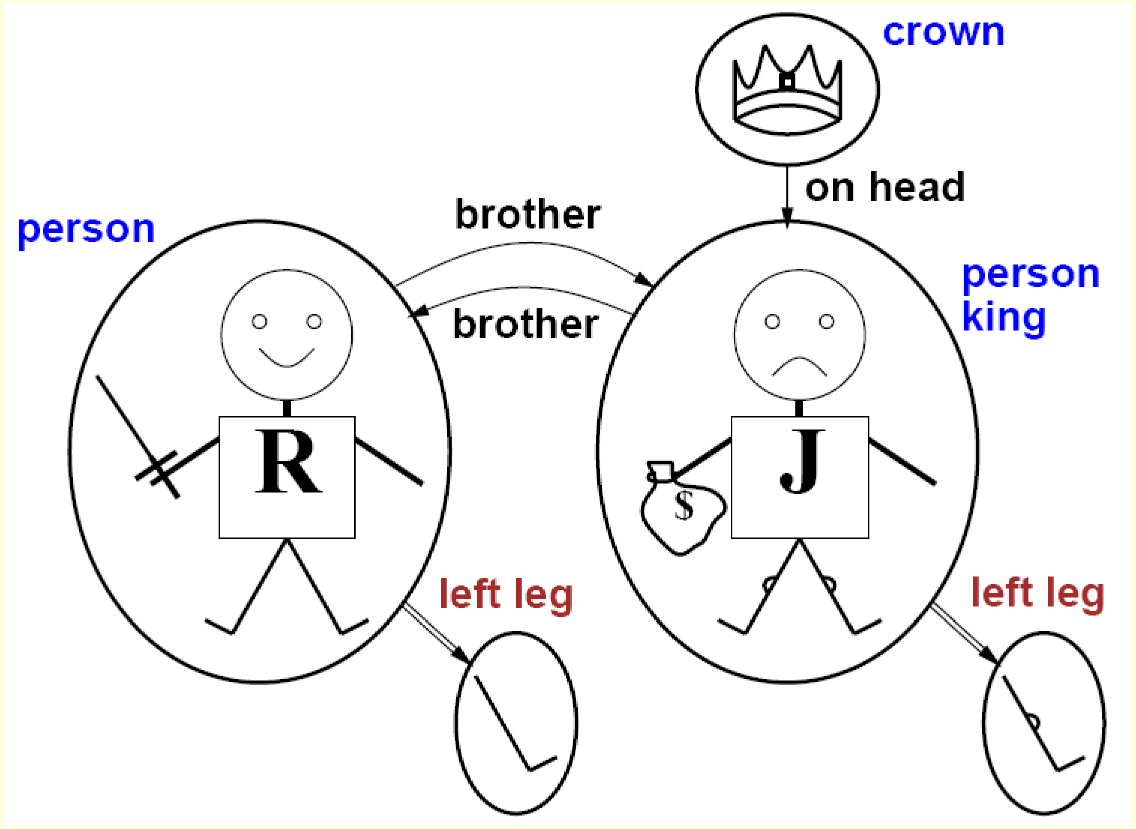
- Quantifiers
  - Each quantifier defines a variable for the duration of the following expression, and indicates the truth of the expression...
- Universal quantifier “for all” $\forall$
  - The expression is true for every possible value of the variable
- Existential quantifier “there exists” $\exists$
  - The expression is true for at least one value of the variable

Truth in First-Order Logic

- Sentences are true with respect to a model and an interpretation
- Model contains >= 1 object (domain elements) and relations among them
- Interpretation specifies referents for
  - constant symbols -> objects
  - predicate symbols -> relations
  - function symbols -> functional relations

- An atomic sentence predicate($\text{term}_1,\ldots,\text{term}_n$) is true iff the objects referred to by $\text{term}_1,\ldots,\text{term}_n$ are in the relation referred to by predicate

First-Order Logic Example



Universal Quantification

- $\forall$ <variables> <sentence>
- $\forall x\ P$ is true in a model m iff P with x being each possible object in the model

- Equivalent to the conjunction of instantiations of P
  - At(Mike, KSU) $\Rightarrow$ Smart(Mike) $\wedge$
  - At(Laurie, KSU) $\Rightarrow$ Smart(Laurie) $\wedge$
  - At(Sarah, KSU) $\Rightarrow$ Smart(Sarah) $\wedge$
  - ...

A Common Mistake to Avoid

- Typically $\Rightarrow$ is the main connective with $\forall$
- Common mistake: using $\wedge$ as the main connective with $\forall$
- $\forall x\ At(x, KSU) \wedge Smart(x)$

Existential Quantification

- $\exists$ <variables> <sentence>
- $\exists$ x   P is true in a model m iff P with x being at least one possible object in the model

- Equivalent to the disjunction of instantiations of P
  - At(Mike, KSU) $\wedge$ Smart(Mike) $\vee$
  - At(Laurie, KSU) $\wedge$ Smart(Laurie) $\vee$
  - At(Sarah, KSU) $\wedge$ Smart(Sarah) $\vee$
  - ...

Another Common Mistake to Avoid

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\Rightarrow$ as the main connective with $\exists$
- $\exists x \ At(x, KSU) \Rightarrow Smart(x)$

Examples

- Everyone likes McDonalds
  - $\forall x$, likes(x, McDonalds)
- Someone likes McDonalds
  - $\exists x$, likes(x, McDonalds)
- All children like McDonalds
  - $\forall x$, child(x) $\Rightarrow$ likes(x, McDonalds)
- Everyone likes McDonalds unless they are allergic to it
  - $\forall x$, likes(x, McDonalds) $\vee$ allergic(x, McDonalds)
  - $\forall x$, $\neg$allergic (x, McDonalds) $\Rightarrow$ likes(x, McDonalds)

Properties of Quantifiers

- $\forall x\ \forall y$ is the same as $\forall y\ \forall x$
- $\exists x\ \exists y$ is the same as $\exists y\ \exists x$
- $\exists x\ \forall y$ 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"

Nesting Quantifiers

- Everyone likes some kind of food

  $\forall y\ \exists x,\ \ \text{food}(x) \wedge \text{likes}(y, x)$

- There is a kind of food that everyone likes

  $\exists x\ \forall y,\ \ \text{food}(x) \wedge \text{likes}(y, x)$

- Someone likes all kinds of food

  $\exists y\ \forall x,\ \ \text{food}(x) \wedge \text{likes}(y, x)$

- Every food has someone who likes it

  $\forall x\ \exists y,\ \ \text{food}(x) \wedge \text{likes}(y, x)$

Examples

- Quantifier Duality
  - Not everyone like McDonalds

    $\neg(\forall x,\ likes(x, McDonalds))$

    $\exists x, \neg likes(x, McDonalds)$

  - No one likes McDonalds

    $\neg(\exists x,\ likes(x, McDonalds))$

    $\forall x, \neg likes(x, McDonalds)$

Fun with Sentences

- Brothers are siblings

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

- Sibling is "symmetric"

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

Fun with Sentences

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

Other Comments About Quantification

- To say "everyone likes McDonalds", the following is too broad!
  - $\forall x$, likes(x, McDonalds)
  - Rush's example: likes (McDonalds, McDonalds)

- We mean: Every one (who is a human) likes McDonalds
  - $\forall x$, person(x) $\Rightarrow$ likes(x, McDonalds)

- Essentially, the left side of the rule declares the class of the variable x

- Constraints like this are often called "domain constraints"

Equality

- We allow the usual infix = operator
  - Father(John) = Henry
  - $\forall x,\ sibling(x, y) \Rightarrow \neg(x=y)$
- Generally, we also allow mathematical operations when needed, e.g.
  - $\forall x,y,\ NatNum(x) \wedge NatNum(y) \wedge x = (y+1) \Rightarrow x > y$
- Example: (Sibling in terms of Parent)

  $\forall x,y\ Sibling(x,y) \Leftrightarrow [\neg(x=y) \wedge \exists m,f\ \neg(m=f) \wedge Parent(m,x) \wedge Parent(f,x) \wedge Parent(m,y) \wedge Parent(f,y)]$

Example Domains

- Kinship domain
  - What is a second cousin once removed, anyway?
- Numbers, sets, and lists
  - This one is a classic. You should understand these, even if you don't memorize them.
- The Wumpus World
  - Note how much simpler the description is in FOL!
- "Whatever your domain, if the axioms correctly and completely describe how the world works, any complete logical inference procedure will infer the strongest possible description of the world, given the available percepts" (AIMA, p. 260)

Interacting with FOL KBs

- Tell the system assertions
  - Facts :
    - Tell (KB, person (John))
  - Rules:
    - Tell (KB,∀x, person(x) ⇒ likes(x, McDonalds))
- Ask questions
  - Ask (KB, person(John))
  - Ask (KB, likes(John, McDonalds))
  - Ask (KB, likes(x, McDonalds))

Types of Answers

- Fact is in the KB
  - Yes.
- Fact is not in the KB
  - Yes (if it can be proven from the KB)
  - No (otherwise)
- Fact contains variables
  - List of bindings for which the fact can be proven, e.g. ((x Fred) (x Mary) ...)

Interacting with FOL KBs

- Suppose a wumpus-world agent is using a FOL KB and perceive a smell and breeze (but no glitter) at t=5

- TELL(KB, Percept([Smell, Breeze, None],5))
- ASK(KB, ∃a Action(a, 5))
  - i.e. does the KB entail any particular action at t=5?

- Answer: Yes, {a/Shoot}   <- substitution (binding list)

Interacting with FOL KBs

- Given a Sentence S and a substitution σ,
- Ss denotes the result of plugging σ in to S;
- Example:
  - S = Taller(x, y)
  - σ = {x/Mike, y/Laurie}
  - Sσ = Taller(Mike, Laurie)
- ASK(KB,S) returns some/all σ such that KB |=Sσ

Knowledge Base for 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"
  - $\forall$ t AtGold(t) $\wedge$ $\neg$Holding(Gold, t) $\Rightarrow$ Action(Grab, t)

- Holding(Gold, t) cannot be observed
  - Keeping track of change is essential!!!!

Deducing Hidden Properties

- Properties of locations:

  $\forall x, t\ At(Agent, x, t) \wedge Smelt(t) \Rightarrow Smelly(t)$

  $\forall x, t\ At(Agent, x, t) \wedge Breeze(t) \Rightarrow Breezy(t)$

- Squares are breezy near a pit:
  - Diagnostic Rule – infer cause from effect
    - $\forall y\ Breezy(y) \Rightarrow \exists x\ Pit(x) \wedge Adjacent(x, y)$
  - Causal Rule – infer effect from cause
    - $\forall x,y\ Pit(x) \wedge Adjacent(x,y) \Rightarrow Breezy(x,y)$

- Neither is complete
  - E.g. the causal rule doesn’t say whether squares far away from pits can be breezy

Deducing Hidden Properties

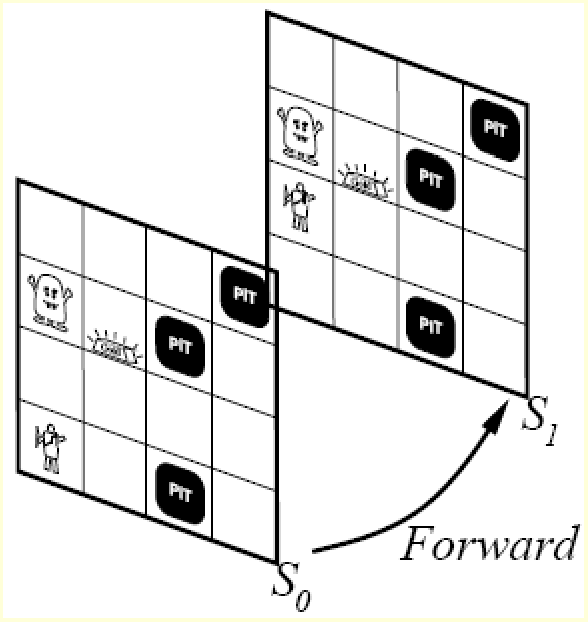
- **Definition for the Breezy predicate:**
  - If a square is breezy, some adjacent square must contain a pit
    - $\forall y\ \mathrm{Breezy}(y) \Rightarrow \exists x\ \mathrm{Pit}(x) \wedge \mathrm{Adjacent}(x, y)$
  - If a square is not breezy, no adjacent pit contains a pit
    - $\forall y\ \neg\mathrm{Breezy}(y) \Rightarrow \neg\exists x\ \mathrm{Pit}(x) \wedge \mathrm{Adjacent}(x, y)$
  - **Combining these two…**
    - $\boldsymbol{\forall y\ \mathrm{Breezy}(y) \Leftrightarrow \exists x\ \mathrm{Pit}(x) \wedge \mathrm{Adjacent}(x, y)}$

Keeping Track of Change

- Often, 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

Keeping Track of Change

- Situations are connected by the <u>*Result*</u> function Result(a, s) is the situation that results from doing *a* in <u>s</u>



Describing Actions

- “Effect” axiom – describe changes due to action
- “Frame” axiom – describe non-changes due to action
- 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

Describing Actions

- “Effect” axiom – describe changes due to action
- “Frame” axiom – describe non-changes due to action
- 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

Describing Actions

- Qualification Problem:
  - True descriptions of real actions require endless caveats...
    - “What if the 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 and no action made P false]

- Example: 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 to make plans is to represent plans as a sequence of actions [a1, a2, ..., an]

- PlanResult(p, s) is the result of executing p in s

- Then the query
  - ASK(KB, ∃p Holding(Gold, PlanResult(p, S_0)))
  - has the solution {p / [Forward, Grab]}

Making Plans

- 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