First Order Logic

First Order Logic (FOL)

Propositional logic (PL)

- Not expressive enough
- Need huge amount of rules
- No power to handle groups of similar objects
- Object is specified individually

First order logic (FOL)

- Introduce objects and properties concepts
- Overcome PL weak expressiveness

New Concepts

Relations

- Links among objects
- Functions are also relations
  - Unique output for a given input

Examples:

- Objects (Nouns)
  - People, houses, number, colors, baseball
- Relations (Adjectives)
  - Unary: involves only 1 object (called property)
    - Tall, large, small, red, round
  - N-ary: involves 2 objects or more
    - Brother of, greater than, part of, inside
- Functions: father of, best friend

New Concepts

Fact (sentences) can be thought of

- Objects
- Properties or relations

"One plus two equals three"

- **Objects**: one, two, three, one plus two
- **Function**: plus
- **Relation**: equals

A name of the object obtained by applying the function *plus* to the objects *one* and *two*

"Squares neighboring the wumpus are smelly."

- **Objects**: Squares, wumpus
- **Property**: smelly
- **Relation**: neighboring

Not a function because many squares may satisfy the constraints, but there is only one *three*

First Order Logic (FOL)

FOL is important

- Express almost any concept/knowledge

Drawbacks

- Categories / Classification
- Time (Temporal Logic)
- Events

Advantages

- Express anything that can be programmed
- Directly translated to Prolog programs

Difference between FOL and PL

PL

- Fact x = True or False
- Semantic interpretation
  - Sentence is true or false

FOL

- Consider relations with objects
  - $Brother(x, y)$ = True or False
  - where x, y = any object, not only True or False
- Semantic interpretation

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Models for FOL

PL model

- Combination of truth values
  - For variables in sentence
- Only True or False exists for the variables

P	Q	$\neg P$	$P \wedge Q$
false	*false*	*true*	*false*
false	*true*	*true*	*false*
true	*false*	*false*	*false*
true	*true*	*false*	*true*

FOL model

- Values for variables
  - Not only True or False
  - Also objects
- E.g. $\text{father}(X, Y) \Rightarrow \text{male}(X)$
  - Objects that make $\text{father}(X, Y)$ true
  - X = peter? Y = john? … over the whole world?
- Domain of FOL model
  - Set of possible objects

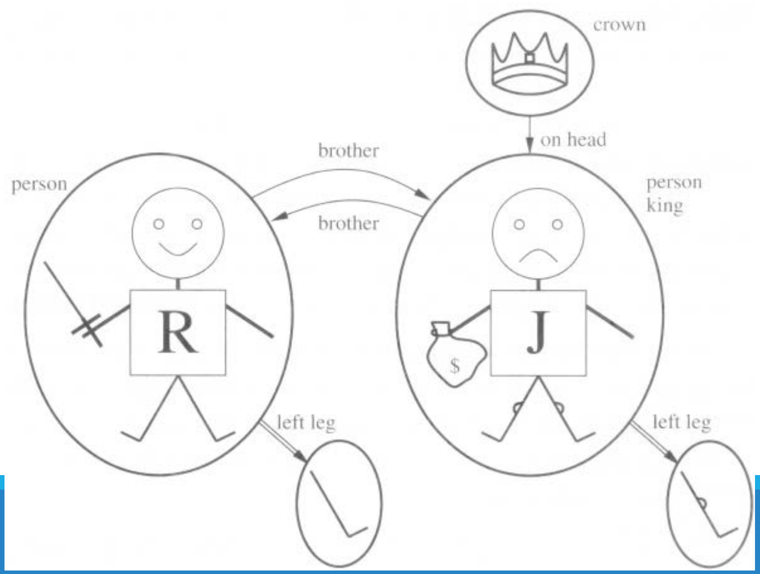
Domain of FOL

Domain of the figure

- Five objects (domain elements)

Relations

- Two binary relations: brother, onhead
- Three unary relations: person, king, crown
- One unary function: left-leg



Syntax of FOL

Atomic sentence

- Relation + Objects
- Facts in Prolog
  - Predicate symbol + Term
  - E.g. brother(richard, john)
  - married(father(richard), mother(john))

Term

- Logical expression of object
- Term =
  - *function symbols*
    - fatherof(peter), plus(1,2)
  - *constant symbols (1, A, B, Peter)*
  - *variables* (x, y, human)

$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \\
 & | & (\ \mathit{Sentence}\ \mathit{Connective}\ \mathit{Sentence}\ ) \\
 & | & \mathit{Quantifier}\ \mathit{Variable}, \ldots\ \mathit{Sentence} \\
 & | & \neg\ \mathit{Sentence} \\
 & & \\
\mathit{AtomicSentence} & \rightarrow & \mathit{Predicate}(\mathit{Term}, \ldots) \mid \mathit{Term} = \mathit{Term} \\
 & & \\
\mathit{Term} & \rightarrow & \mathit{Function}(\mathit{Term}, \ldots) \\
 & | & \mathit{Constant} \\
 & | & \mathit{Variable} \\
 & & \\
\mathit{Connective} & \rightarrow & \Rightarrow \mid \wedge \mid \vee \mid \Leftrightarrow \\
\mathit{Quantifier} & \rightarrow & \forall \mid \exists \\
\mathit{Constant} & \rightarrow & A \mid X_1 \mid \mathit{John} \mid \cdots \\
\mathit{Variable} & \rightarrow & a \mid x \mid s \mid \cdots \\
\mathit{Predicate} & \rightarrow & \mathit{Before} \mid \mathit{HasColor} \mid \mathit{Raining} \mid \cdots \\
\mathit{Function} & \rightarrow & \mathit{Mother} \mid \mathit{LeftLeg} \mid \cdots
\end{array}
$$

Syntax of FOL

Complex sentences

- Multiple atomic sentences
- Combined with logical connectives
- Example
  - brother(richard, john) $\wedge$ brother(john, richard)
  - older(john, 30) $\Rightarrow$ $\neg$younger(john, 30)

Quantifiers

Quantifiers

Expressing properties / constraints

- For entire collection of objects

Universal quantification ($\forall$)

- All domain elements
  - Read as "For all"
- Example: "All Kings are Persons"

$$\forall x\ King(x) \Rightarrow Person(x)$$

If x is a King, then x is a Person

- a **variable**,
- if it's a constant,

a **ground term**

Universal Quantification (∀)

$\forall x\ King(x) \Rightarrow Person(x)$ is true

- *x* = any domain element, sentence is still true
  - *x* → Richard
  - *x* → John
  - *x* → Richard’s left leg
  - *x* → John’s left leg
  - *x* → the crown
- List is called extended interpretation

Richard the Lionheart is a king $\Rightarrow$ Richard the Lionheart is a person.
King John is a king $\Rightarrow$ King John is a person.
Richard’s left leg is a king $\Rightarrow$ Richard’s left leg is a person.
John’s left leg is a king $\Rightarrow$ John’s left leg is a person.
The crown is a king $\Rightarrow$ the crown is a person.

Universal Quantification ($\forall$)

All the models are true

Only for interpretation

- Implication ($\Rightarrow$)
  - Whenever premise is false
  - Result is true, regardless of the conclusion

Universal quantifier

- Asserts / produces a list of similar sentences
- In PL, all of these sentences are made ourselves
- Reduce our works

Existential Quantification (∃)

Some domain elements

- Read as “There exist” or “For some”

Example

- *$\exists x$ Crown(x) $\wedge$ OnHead(x, John)*

True if at least one domain element satisfies the sentence

Richard the Lionheart is a crown $\wedge$ Richard the Lionheart is on John’s head;
King John is a crown $\wedge$ King John is on John’s head;
Richard’s left leg is a crown $\wedge$ Richard’s left leg is on John’s head;
John’s left leg is a crown $\wedge$ John’s left leg is on John’s head;
The crown is a crown $\wedge$ the crown is on John’s head.

Quantifiers

$$\forall x \;\; King(x) \land Person(x)$$

would be equivalent to asserting

Richard the Lionheart is a king $\land$ Richard the Lionheart is a person,
King John is a king $\land$ King John is a person,
Richard's left leg is a king $\land$ Richard's left leg is a person,

If $\land$ with $\forall$, too strong

If => with $\exists$, too weak

Hence

- $\Rightarrow$ is natural connective with $\forall$
- while $\land$ with $\exists$

Nested Quantifiers

Using multiple quantifiers

- $\forall x\ \forall y$ Brother(x, y) $\Rightarrow$ Sibling (x, y)
- Can be written as $\forall x, y$

$\forall x\ \exists y$ Loves (x, y)

- Everybody x loves somebody y
- $\exists y\ \forall x$ Loves (x, y)? Any difference?
- There is somebody y, whom is loved by everybody x.

Quantifiers are not commutative

- Order cannot be interchanged

To specify precedence

- Should use ( ), e.g. $\exists y\ (\ \forall x$ Loves (x, y))

Connections between $\forall$ and $\exists$

$\forall$ is a conjunction over the universe

$\exists$ is a disjunction

- DeMorgan rules can apply to them
  - $\forall x \neg P \equiv \neg \exists x P$
  - $\forall x P \equiv \neg \exists x \neg P$
  - $\neg \forall x P \equiv \exists x \neg P$
  - $\exists x P \equiv \neg \forall x \neg P$

They are equivalent

- Only one of $\forall$ or $\exists$ is necessary
- Do not need both, PROLOG uses only $\forall$

Uniqueness Quantifier ∃!

∃ specifies

- One or more objects

∃! is used to specify

- A unique one object

Example: “There is only one king”

- $\exists! x\ King(x)$
- $\exists x\ King(x) \wedge \exists y\ King(y) \Rightarrow (x = y)$
- If X is a King & Y is a King, then X must be Y

Equality

Represented as “=”

- Example: ***FatherOf(John) = Henry***

Ensure two objects are not the same

- Negation with equality is used
- E.g. $\exists x,y\ Sister(Felix, x) \land Sister(Felix, y) \land \neg(x = y)$

Using First Order Logic

Using First Order Logic

Domain

- Application or a section of the world
  - In expressing knowledge

Examples

- The kinship domain
- The domain of numbers
- The domain of sets and lists

The Kinship Domain

Family relationships

- Objects in the domain are people
- Properties of the objects
  - Gender (Male or female)
  - Age
  - Height, …
- Relations
  - Parenthood
  - Brotherhood
  - Marriage, …

Domain Axioms (Rules)

$\forall m,c\ Mother(c)=m \Leftrightarrow Female(m) \wedge Parent(m,c)$

$\forall w,h\ Husband(h,w) \Leftrightarrow Male(h) \wedge Spouse(h,w)$

Disjoint categories:

- $\forall x\ Male(x) \Leftrightarrow \neg Female(x)$

Inverse relations:

- $\forall p,c\ Parent(p,c) \Leftrightarrow Child(c,p)$

$\forall g,c\ Grandparent(g,c) \Leftrightarrow \exists p\ Parent(g,p) \wedge Parent(p,c)$

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

Many more **axioms** like these

Defining Axioms

A set of primitive predicates is firstly identified

- Male, Female, Parent, …
  - i.e., Prolog facts
  - E.g., location(kitchen, apple), door(office, kitchen), …
- Other predicates can be used
  - as the primitive set
  - Ensure axioms can later be defined correctly

Some domains

- No clearly identifiable primitive set

Domain of Numbers

Basic theory of natural numbers

- Natural Number $N \in Z_0^+$
- Check if a number is natural
  - NatNum: N → {True, False}
    - Constant symbol (basis)
      - *0*
    - Function symbol *S*, meaning successor
      - i.e. $S(0) = 0 + 1 = 1$.

$$NatNum(0).$$

$$\forall n\ NatNum(n) \Rightarrow NatNum(S(n)).$$

Domain of Numbers

Constraints about the function S

$\forall n\ S(n) \neq 0$

$\forall m, n\ m \neq n \Leftrightarrow S(m) \neq S(n)$

Addition of natural numbers

$\forall m\ NatNum(m) \Rightarrow (+ (m, 0) = m)$

$\forall m, n\ NatNum(m) \wedge NatNum(n) \Rightarrow +(S(m), n) = S(+(m,n))$

Defined base on idea of Natural number

- Express the idea in FOL

Domain of Sets

Represent sets, including empty set

- Way to build up a set
  - Add element to a set (adjoining)
  - Union of two sets
  - Intersection of two sets
- Checking of an object
  - A set?
  - Member of a set?
  - Subset of a certain set?

Constant symbol: { }
Predicates: *Set, Member, Subset*
Functions: *Adjoining, Union, Intersection*

1. The only sets are the empty set and those made by adjoining something to a set:

$$\forall s \;\; Set(s) \;\Leftrightarrow\; (s = \{\,\}) \vee (\exists x, s_2 \;\; Set(s_2) \wedge s = \{x|s_2\}) \;.$$

2. The empty set has no elements adjoined into it, in other words, there is no way to decompose $EmptySet$ into a smaller set and an element:

$$\neg \exists x, s \;\; \{x|s\} = \{\,\} \;.$$

3. Adjoining an element already in the set has no effect:

$$\forall x, s \;\; x \in s \;\Leftrightarrow\; s = \{x|s\} \;.$$

4. The only members of a set are the elements that were adjoined into it. We express this recursively, saying that x is a member of s if and only if s is equal to some set s_2 adjoined with some element y, where either y is the same as x or x is a member of s_2:

$$\forall x, s \;\; x \in s \;\Leftrightarrow\; [\exists y, s_2 \;\; (s = \{y|s_2\} \wedge (x = y \vee x \in s_2))] \;.$$

5. A set is a subset of another set if and only if all of the first set's members are members of the second set:

$$\forall s_1, s_2 \;\; s_1 \subseteq s_2 \;\Leftrightarrow\; (\forall x \;\; x \in s_1 \;\Rightarrow\; x \in s_2) \; .$$

6. Two sets are equal if and only if each is a subset of the other:

$$\forall s_1, s_2 \;\; (s_1 = s_2) \;\Leftrightarrow\; (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1) \; .$$

7. An object is in the intersection of two sets if and only if it is a member of both sets:

$$\forall x, s_1, s_2 \;\; x \in (s_1 \cap s_2) \;\Leftrightarrow\; (x \in s_1 \wedge x \in s_2) \; .$$

8. An object is in the union of two sets if and only if it is a member of either set:

$$\forall x, s_1, s_2 \;\; x \in (s_1 \cup s_2) \;\Leftrightarrow\; (x \in s_1 \vee x \in s_2) \; .$$

Domain of Lists

Similar to sets

Differences

- Element can appear more than once
- Ordered

$\emptyset$ ={ }	[] = Nil
{x} = {x | { } }	[x] = Cons(x, Nil)
{x, y} = {x | {y | { } } }	[x,y] = Cons(x, Cons(y, Nil))
{x, y|s} = {x | { y | s} }, s is a set	[x,y|l] = Cons(x, Cons(y, l))
r $\cup$ s = Union(r, s)	
r $\cap$ s = Intersection(r, s)	
x $\in$ s = Member(x, s)	
r $\subseteq$ s = Subset(r, s)	

First Order Logic in Wumpus World

The Wumpus World

Agent percept vector

- [Stench, Breeze, Glitter, Bump, Scream]

Percept is time critical

- Add a time step
- percept([S, B, G, None, None], 5)

Action

- Turn(Right), Turn(Left), Forward, Grab…
- Objective: Take best action for any time

Best Action

BestAction(a, t)

- E.g. glitter is perceived at t
- a = Grab

Tell KB what happens

- Transform perception
  - $\forall$ s,g,u,c,t Percept([s, Breeze, g, u, c], t) $\Rightarrow$ Breeze(t)
  - $\forall$ s,b,u,c,t Percept([s, b, Glitter, u, c], t) $\Rightarrow$ Glitter(t)

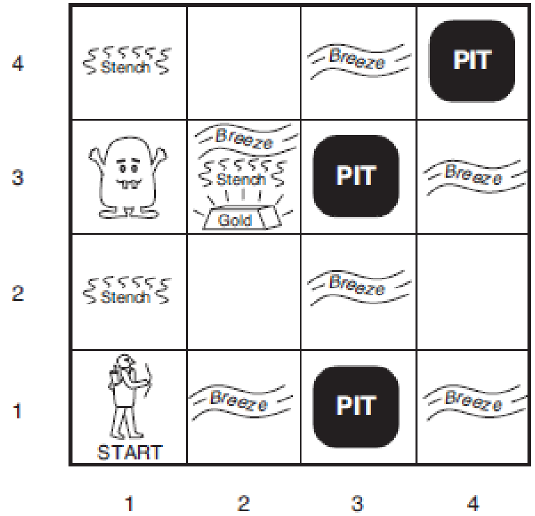
With “telled” information

- Additional rules are defined
- $\forall$t Glitter(t) => BestAction(Grab, t)

Define Environment

Objects

- Squares
- Pits
- Wumpus



Square

- $S_{1,1}$, $S_{1,2}$, so on

Adjacent squares

$$\forall x, y, a, b \; Adjacent([x, y],[a, b]) \Leftrightarrow$$
$$[a, b] \in \{[x+1, y],[x-1, y],[x, y+1],[x, y-1]\}$$

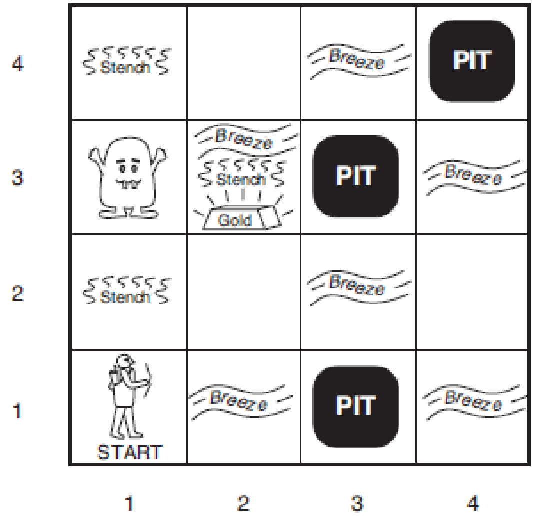
Define Environment

Pits

- No need to name individually
- Use unary predicate
  - Pit([$S_{3,1}$, $S_{3,3}$, $S_{4,4}$])

Wumpus

- Only one square
- Function: Home(wumpus)
  - Return the square $S_{1,3}$
- Multiple wumpuses
  - Similar to Pit(), i.e. Wumpus([$W_{1,3}$, $W_{3,4}$])



Define Environment

Agent moves

- Changes location $L_{x,y}$ over time
- At(agent, *s*, *t*)
  - At time step *t*, agent is at *s*

Properties of environment

- Constant
- Square is breezy

  $\forall s, t\ At(agent, s, t) \land Breeze(t) \Rightarrow Breezy(s)$
- Same for smelly

  $\forall s, t\ At(agent, s, t) \land Stench(t) \Rightarrow Smelly(s)$

Diagnostic Rules (→)

From given facts, find reason/cause

- E.g. square is breezy
  - Some adjacent square has a pit
  - $\forall s\ Breezy(s) \Rightarrow \exists r\ Adjacent(r, s) \land Pit(r)$
  - Percept → Cause
- Reverse direction is true
  - $\forall s\ Breezy(s) \Leftrightarrow \exists r\ Adjacent(r, s) \land Pit(r)$

Causal Rules (←)

From given cause, conclude with facts/results

- Cause → Percept
- r is a pit
  - All adjacent squares of r are breezy
  - $\forall r\ Pit(r) \Rightarrow [\forall s\ Adjacent(r, s) \Rightarrow Breezy(s)]$
- All squares adjacent to square s are not pits
  - s is not breezy
  - $\forall s\ [\forall r\ Adjacent(r, s) \Rightarrow \neg Pit(r)] \Rightarrow \neg Breezy(s)$

Equivalent to previous bidirectional rule

Conclusion

No matter which kind of representation

Axioms are correct and complete

- The way the world works
- The way percepts are produced

Complete logical inference procedure

- With given available percepts
- Infer strongest possible description of the world state