Lecture 8:
First Order Logic

Artificial Intelligence

CS-6364

Outline

- Why FOL?
- Syntax and semantics of FOL
- Using FOL
- Wumpus world in FOL
- Knowledge engineering in FOL

Pros and cons of propositional logic

- ☺ Propositional logic is declarative
- ☺ 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 the world contains facts,
- first-order logic (like natural language) assumes the world contains
  - Objects: people, houses, numbers, colors, baseball games, wars, …
  - Relations: red, round, prime, brother of, bigger than, part of, comes between, …
  - Functions: father of, best friend, one more than, plus, …

Ontological commitments

- The world consists of objects having specific properties
- Various relations hold between objects
    - some are functions
- Examples:
    - Objects: people, houses, numbers, theories, wars
    - Relations: brother-of, bigger-than, inside, has-color
    - Properties: red, round, bogus, multistoried
    - Functions: father-of, best-friend

Syntax and semantics

- In Propositional logic, every expression is a sentence, representing a fact
- FOL has sentences, but also terms, representing objects

Constant symbols

Example: John $\rightarrow$ King John, king of England from 1199 to 1216

king

Syntax and semantics

Predicate symbols

- defined by a set of tuples that satisfy it

Example:

brother

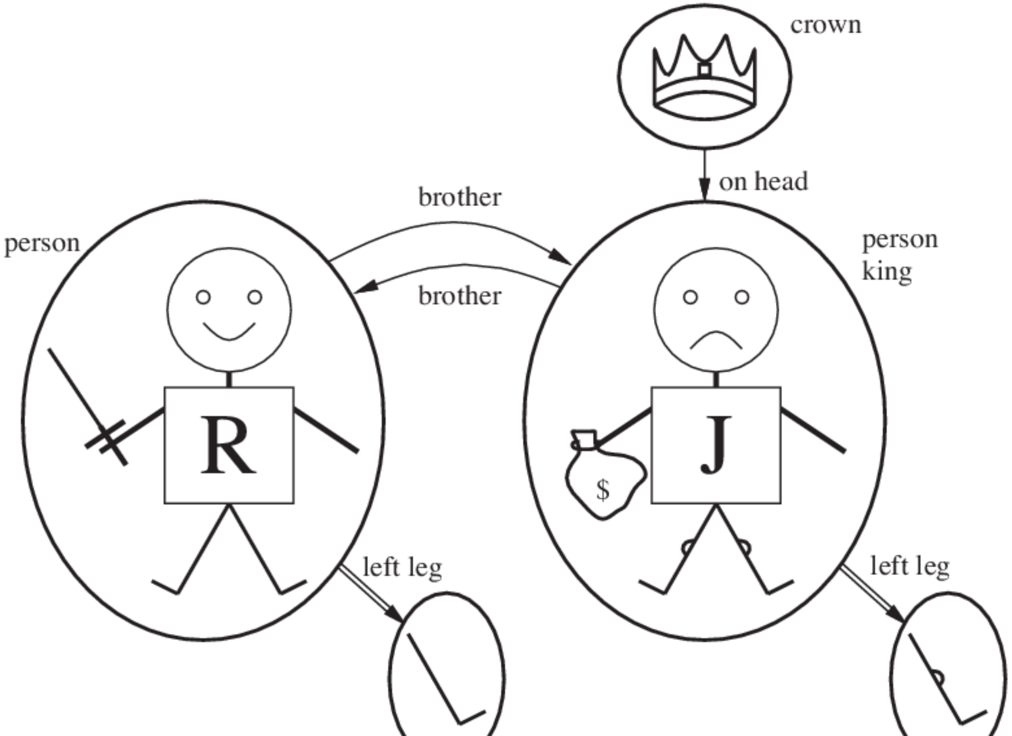
Pairs ⇒ { <King_John, Richard_the_Lionheart>,
<Gianni_Versace, Donatella_Versace>
..... }

⇒ Why tuples?

→ Because predicates have different types

A Model with Total Functions

- Relations are tuples: *OnHead*(CROWN, KING_JOHN); *brother*(RICHARD_THE_LIONHEART, KING_JOHN); *king*(KING_JOHN);
- BUT some relations are better considered as **<u>functions: </u>** e.g. each person has **one** left leg, expressed as a function" *LeftLegOf*(RICHARD_THE_LIONHEART) → the left leg of Richard. Moreover, FOL requires **total functions**, i.e. there must be a value for each functional tuple in FOL



Syntax of FOL in BNF

Sentence → AtomicSentence
| Sentence Connective Sentence
| Quantifier Variable,... Sentence
| ¬Sentence
| (Sentence)

AtomicSentence → Predicate(Term,...) | Term=Term

Term → Function(Term,...)
| Constant
| Variable

Connective → ⇒ | ∧ | ∨ | ⇔

Quantifier → ∀ | ∃

Constant → A | X_1 | John | ...

Variable → a | x | s | ...

Predicate → True|False| Before | HasColor | Raining | ...

Function → Mother | LeftLegOf | ...

Syntax of FOL: Basic elements

- Constants KingJohn, 2, NUS,...
- Predicates *brother*, >,...
- Functions Sqrt, LeftLegOf,...
- Variables x, y, a, b,...
- Connectives $\neg$, $\Rightarrow$, $\wedge$, $\vee$, $\Leftrightarrow$
- Equality $=$
- Quantifiers $\forall$, $\exists$

Atomic sentences

Atomic sentence = *predicate* $(term_1,\ldots,term_n)$
or $term_1 = term_2$

Term = *function* $(term_1,\ldots,term_n)$
or *constant* or *variable*

- E.g., *Brother(KingJohn,RichardTheLionheart), (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))*

Atomic sentences

- made from a single predicate symbol and its arguments

Examples

1) Brother(Richard, John)
2) Married(FatherOf(Richard), MotherOf(John))

- The truth of a sentence depends on:
  - interpretation
  - the world

Terms

 Definition A term is a logical expression that refers to an object

! Sometimes it is easier to use an expression to refer to an object rather than giving it a name !

Example: "King John's left leg"

LeftLegOf(John)

Function symbols

Examples: Cosine, FatherOf, LeftLegOf

$\Rightarrow$ Some relations are functional, because any given object is related to exactly one other object by the relation

- Predicate symbols represent relations between objects
- Functional symbols are used to refer to particular objects

Complex sentences

- Complex sentences are made from atomic sentences using connectives

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

E.g. *Sibling(KingJohn,Richard)* $\Rightarrow$ *Sibling(Richard,KingJohn)*

$>(1,2) \vee \leq (1,2)$

$>(1,2) \wedge \neg >(1,2)$

Complex sentences

- Using logical connectives, we can construct more complex sentences

Examples:

- $Brother(Richard,John) \wedge Brother(John,Richard)$
- $Older(John,30) \vee Younger(John,40)$
- $Older(John,30) \Rightarrow \neg Younger(John,30)$
- $\neg Brother(Robin,John)$

Truth in first-order logic

- Sentences are true with respect to a model and an interpretation

- Model contains objects (domain elements) and relations among them
- Interpretation specifies referents for

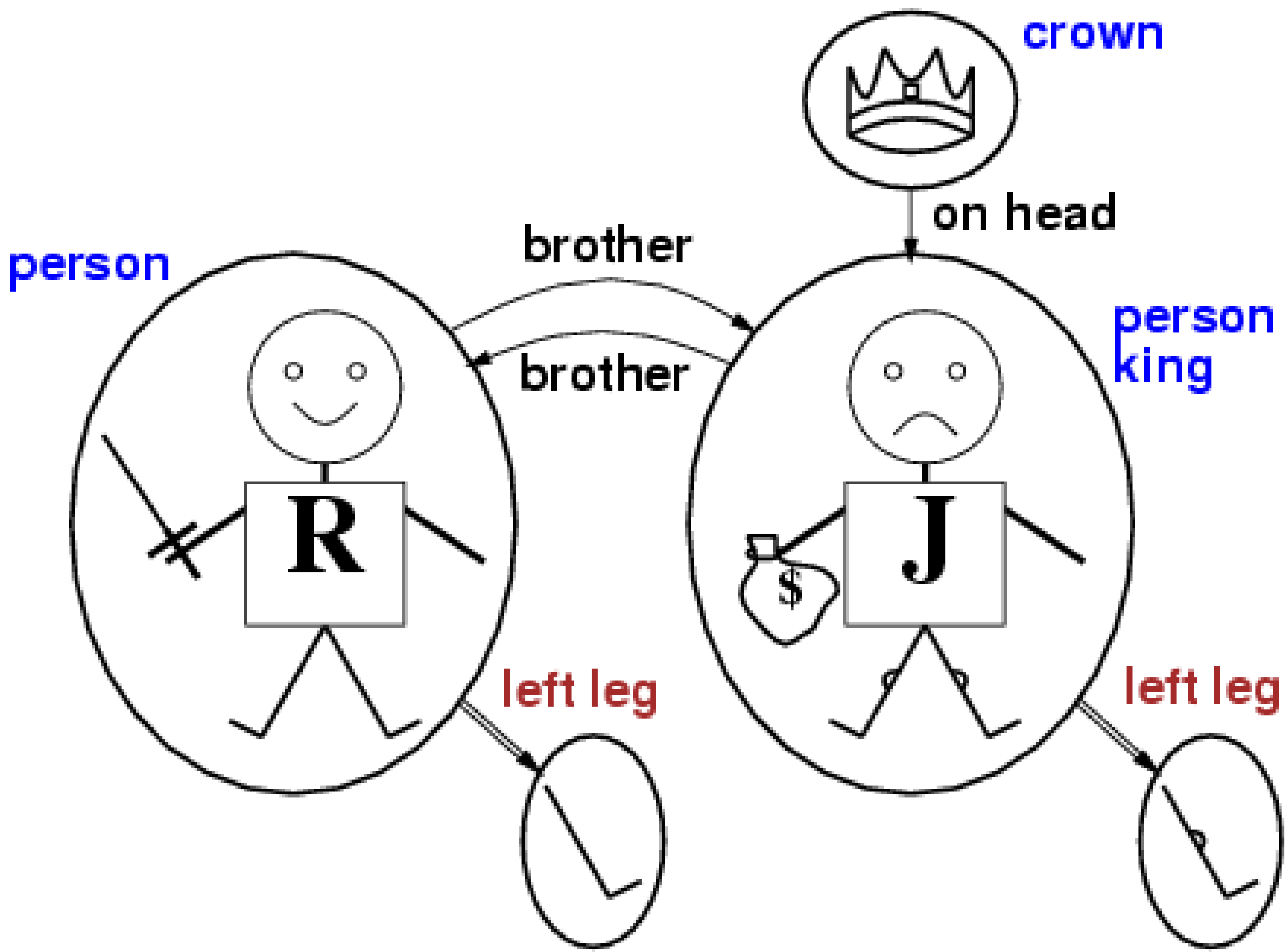
  constant symbols → objects

  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



Quantifiers

- If a logic allows objects, we also want to express *properties* of entire collections of objects, rather than having to enumerate the objects by name

$\Rightarrow$ This is possible with quantifiers

First order logic contains two quantifiers

- Universal quantifier ($\forall$)
- Existential quantifier ($\exists$)

Universal quantification

- $\forall$<*variables*> <*sentence*>

Everyone at NUS is smart:

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

- $\forall x\ P$ 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*

$$
\begin{array}{ll}
 & At(KingJohn,NUS) \Rightarrow Smart(KingJohn) \\
\wedge & At(Richard,NUS) \Rightarrow Smart(Richard) \\
\wedge & At(NUS,NUS) \Rightarrow Smart(NUS) \\
\wedge \ldots &
\end{array}
$$

Universal quantification

- In propositional logic it is difficult to express general rules such as "All cats are mammals"
  - Such rules are the bread and butter of FOL

    In FOL $\forall x\ Cat(x) \Rightarrow Mammal(x)$
- *Variables* are predicate arguments
- Predicates without variables are called *ground terms*

Intuitively, the sentence $\forall x\ P$ says P is true for every object. $\forall x\ P$ is true in all possible extended interpretations constructed from the given interpretation of the model. An extended interpretation specifies the domain element to which x refers.

Universal Quantification and Extended Interpretations

- Let us consider $\forall x\ King(x) \Rightarrow Human(x)$
  - x????

x → Richard the Lionheart

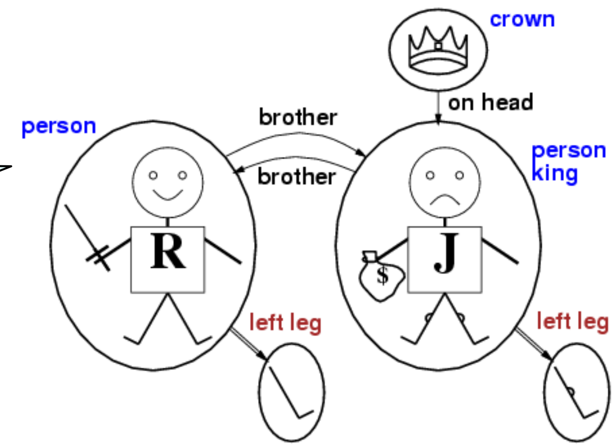
x → King John

x → Richard's left leg

x → King John's left leg

x → the crown

The extended Model



Then $\forall x\ King(x) \Rightarrow Human(x)$ is TRUE in the original model if it is true for each of the five extended interpretations:

I1: Richard the Lionheart is a king ⇒ Richard the Lionheart is human

I2: King John is a king ⇒ King John is human

I3: Richard's left leg is a king ⇒ Richard's left leg is human

I4: King John's left leg is a king ⇒ King John's left leg is human

I5: the crown is a king ⇒ the crown is human

A common mistake to avoid

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

  $\forall x\ At(x, NUS) \wedge Smart(x)$

  means "Everyone is at NUS and everyone is smart"

➢ Let us consider $\forall x\ King(x) \Rightarrow Human(x)$

If my mistake we write $\forall x\ King(x) \wedge Human(x)$

We would assert a lot of untrue sentences!!!

Existential quantification

- $\exists$<*variables*> <*sentence*>

➢ Someone at NUS is smart:

$\exists x\ At(x,NUS) \land Smart(x)$

➢ King John has a crown on his head

$\exists x\ Crown(x) \land OnHead(x, John)$

$\exists x\ P$ 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*

$$
\begin{aligned}
& At(Joe,NUS) \land Smart(Joe) \\
\lor\ & At(Sarah,NUS) \land Smart(Sarah) \\
\lor\ & At(NUS,NUS) \land Smart(NUS) \\
\lor\ & \ldots
\end{aligned}
$$

Existential quantification

- Universal quantification makes statements about EVERY object!
- Similarly, we can make a statement about SOME object in the universe, without naming it!
- How $\Rightarrow$ using the existential quantifier $\exists$

- Example $\Rightarrow$ the fact that Spot has a sister who is a cat

$$\exists x \; Sister(x, Spot) \wedge Cat(x)$$

Another common mistake to avoid

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

$$\exists x\ \mathrm{At}(x,\mathrm{NUS}) \Rightarrow \mathrm{Smart}(x)$$

is true <u>only</u> if both premise and conclusion are true! *Then why not use* $\wedge$ *????*

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"

- Quantifier duality: each can be expressed using the other

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

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

Nested quantifiers

- Case 1 The quantifiers are of the same type
  Example: For all x and all y, if x is the parent of y, then y is the child of x
  This becomes: $\forall x,y\ Parent(x,y) \Rightarrow Child(y,x)$

$\forall x\ \forall y \Leftrightarrow \forall x,y$
$\exists\ x\ \exists\ y \Leftrightarrow \exists\ x,y$

- Case 2 Mixed quantifiers $\boxed{\text{Order counts!!}}$
  Example: "Everybody loves somebody" - means that for every person, there is someone that person loves

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

Order in nested quantifiers

Everybody loves somebody → $\forall x\ \exists y\ Loves(x,y)$

There is somebody who is loved by everyone → $\exists y\ \forall x\ Loves(x,y)$

- What if 2 quantifiers are used with the same variable?

Example: $\forall x\ [Cat(x) \lor (\exists x\ Brother(Richard,x))]$

Rule: a) The variable belongs to the innermost quantifier that mentions it!

b) Then it will not be subject to the outer quantification anymore!

Similar to scoping in block-structures programming languages

Connections between ∀ and ∃

- Their connection is done through negation

"Everyone dislikes parsnips"

⇕

"There does not exist someone who likes parsnips"

$$\forall x \ \neg Likes(x, parsnips) \Leftrightarrow \neg\exists x \ Likes(x, parsnips)$$

- If we have

"Everyone likes icecream"

⇕

"There is no one that does not like icecream"

$$\forall x \ Likes(x, Icecream) \Leftrightarrow \neg\exists x \ \neg Likes(x, Icecream)$$

Connections between ∀ and ∃

$\forall x \neg P(x) \equiv \neg \exists x P(x)$

$\neg \forall x P(x) \equiv \exists x \neg P(x)$

$\forall x P(x) \equiv \neg\exists x \neg P(x)$

$\exists x P(x) \equiv \neg \forall x \neg P(x)$

De Morgan Rules for Quantification!!!!

$\forall x (p(x) \wedge q(x)) \equiv \forall x p(x) \wedge \forall y q(y)$

$\exists x (p(x) \vee q(x)) \equiv \exists x p(x) \vee \exists y q(y)$

Extensions and notational variations

$\Rightarrow$ Higher-order logic

- In First Order Logic, one can quantify over objects, but not on relations or functions
- Higher-Order Logic, allows us to quantify over objects, relations, functions
- Examples:

  1) $\forall x,y\ (x=y) \Leftrightarrow (\forall p,\ p(x) \Leftrightarrow p(y))$
  2) $\forall f,g\ \ (f=g) \Leftrightarrow (\forall x,\ f(x) = g(x))$

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., definition of *Sibling* in terms of *Parent*:

$$\forall x,y\ Sibling(x,y) \Leftrightarrow [\neg(x = y) \land \exists m,f\ \neg(m = f) \land \mathrm{Parent}(m,x) \land \mathrm{Parent}(f,x) \land \mathrm{Parent}(m,y) \land \mathrm{Parent}(f,y)]$$

Equality

- *It is one of the ways to create atomic sentences in FOL*
- *It is used to indicate that two terms refer to the same object:*

  Father(John) = Henry :: says that the object referred by Father(John) and Henry are the same

When used with negation it shows that two objects are not the same!

- *To say "Richard has <u>at least two brothers</u>", we write:*

  $\exists x,y\ Brother(x,Richard) \wedge Brother(y,Richard) \wedge \neg (x=y)$

- *To say "Richard has <u>only one brother</u>", we write:*

  $\exists x\ Brother(x,Richard) \wedge \forall\ y\ Brother(y,Richard) \Rightarrow (x=y)$

- *To say "Richard has <u>only two brothers</u>", we write:*

  $\exists x,y\ Brother(x,Richard) \wedge Brother(y,Richard) \wedge \neg (x=y)$

  $\wedge \forall\ z\ Brother(z,Richard) \Rightarrow ((x=z) \vee (y=z))$

Using FOL - substitutions

- Sentences are added to a KB by using the TELL function

TELL (KB, *king(John)*)

TELL (KB, *Person(Richard)*)

- We can ask questions using ASK

ASK(KB, *Person(John)*)

→ Returns YES/NO

- If we want to ask ASK(KB, ∃x *Person(x)*) an answer YES/NO does not work – we ask which values of *x* make the sentence TRUE!!! We could call ASKVARS(KB, *Person(x)*) instead!

→ Returns 2 answers: {x/John} and {x/Richard} ←a **<u>substitution</u>** or **binding list**!!!!!

Using FOL

- In knowledge representation, a domain is a section of the world about which we wish to express some knowledge.

Example: the **kinship domain**

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

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

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

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

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

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

Predicates in the Kinship domain

- How to express family relationships?

<u>Example</u>: Elisabeth is the mother of Charles, and Charles is the father of William. *Binary predicates??? How about "One's grandmother is the mother of one's parent"???*

- *What are the objects???* People!
- *What predicates??*
  - *Unary predicates:* male(x), female(x)
  - *Binary predicates:* parent(x,y), sibling(x,y), brother(x,y), sister(x,y), child(x,y), daughter(x,y), son(x,y), spouse(x,y), wife(x,y), husband(x,y), grandparent(x,y),grandchild(x,y), cousin(x,y), aunt(x,y), uncle(x,y)
- *What are the functions?*
  - *2 functions:* mother(x), father(x).

Using FOL

The kinship domain:

- Brothers are siblings

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

- One's mother is one's female parent

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

- "Sibling" is symmetric

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

Using FOL

- A "Sibling" is another child of one's parents

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

Syntactic Sugar

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

- A grandparent is a parent of one's parent

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

- *Parent* and *child* are inverse relations

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

The Set Domain-1

- *The only sets are those made of the empty set and those made by adjoining something to a set*

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

- *The empty set has no elements adjoined into it.*

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

- *Adjoing an element already in the set has no effect.*

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

- *The only elements to a set are those that were adjoined to it.*

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

The Set Domain-1

- *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*

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

- *Two sets are equal if and only if each is a subset of the other*

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

- *An element is in the intersection of two sets if and only if it is a member of both sets*

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

- *An element is in the union of two sets if and only if it is a member of either sets*

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

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 BestAction(a,5))

- I.e., does the KB entail some best action 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 ⊨ σ

Axioms, definitions and theorems

- In mathematics, axioms capture basic facts about a domain

⇒Concepts are also defined in terms of axioms

⇒Axioms encode basic factual information from which useful conclusions can be derived.

⇒ Theorems are proven based on axioms (theorems are entailed by axioms from a KB)

- In AI, the sentences from a KB are also related to as axioms

Definitions are axioms of the form

$$\forall x,y\ P(x,y) \equiv \ldots$$

E.g. $\forall x,y\ Sibling(x,y) \Leftrightarrow Sibling(y,x)$

FOL Knowledge base for the wumpus world

- Perception (remember from propositional logic Lecture 7, slide 11)

> The agent gets them in the form of a list of 5 symbols
> - Example: there is a stench, a breeze, a glitter, but no bump or scream:
>
>   *Percept*([Stench, Breeze, Glitter, Bump, Scream],time5)

- Let us create a binary predicate:

Percept(PerceptSequence, time)

- We also have constants*: Stench, Breeze, Glitter, etc*
- There are also variables: *s,b,g,m,c +* time represented as *t*

$$\forall t,s,g,m,c\ Percept([s,Breeze,g,m,c],t) \Rightarrow Breeze(t)$$

$$\forall t,s,g,m,c\ Percept([s,b,Glitter,m,c],t) \Rightarrow Glitter(t)$$

FOL Knowledge base for the wumpus world-2

- **Actions in the Wumpus world** (remember from propositional logic Lecture 7, slide 9)

- Go forward
- Turn right 90°
- Turn left 90°

+ Grab (to pick up an object that is in the same square as the agent)
+ Shoot (to fire an arrow in straight line in the direction the agent is facing)
+ Climb (to leave the cave !!)

- How do we represent these actions in FOL???

TERMS: *Turn(Right), Turn(Left), Forward, Shoot, Grab, Climb*

- ***To determine which actions is best to execute, the agent asks:***

ASK(∃a, BestAction(a,5))

A simple reflex agent

- Connects percepts to actions = through <u>rules</u>

  <u>Example:</u> the agent sees a glitter, it should do a <u>grab</u> to pick up the gold

$\forall s,b,u,c,t\ \ Percept([s,b,Glitter,u,c],t) \Rightarrow Action(Grab,t)$

Mediate the connection by rules of perception

$\forall b,g,u,c,t\ \ Percept([Smell,b,g,u,c],t) \Rightarrow Smelt(t)$

$\forall s,g,u,c,t\ \ Percept([s,Breeze,g,u,c],t) \Rightarrow Breeze(t)$

$\forall s,b,u,c,t\ \ Percept([s,b,Glitter,u,c],t) \Rightarrow AtGold(t)$

Then connect these predicates to action choices:

$\forall t\ AtGold(t) \Rightarrow Action(Grab,t)$

FOL Knowledge base for the wumpus world-3

- Reflex logical agent:

ASK($\exists$a, BestAction(a,5))

- If in the FOL KB we have:

$\forall t$ Glitter(t) $\Rightarrow$ BestAction($Grab$,t)

Then the agent will conclude that it performs the action Grab

- Percepts and actions represent the logical agent's INPUTs and OUTPUTs. How about the environment???

To represent the square $S_{1,2}$ we can use a list term: [1,2]

Similarly, the square $S_{x,y}$ is represented a [x,y]

Deducing hidden properties

- How do we represent adjacent squares?

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

- How do we represent the fact that the agent's location changes in time??? *If we represent the agent's location through the predicate At(object, location, time), and consider that objects can be only at one location at a time"*

$$\forall x,s_1,s_2,t \quad At(x,s_1,t) \wedge At(x,s_2,t) \Rightarrow s_1=s_2$$

Deducing hidden properties-2

- **<u>Properties of squares:</u>**
  - *If the agent is at a square and perceives a breeze, the square is breezy. In FOL:*

    $\forall s,t\ \mathit{At}(\mathrm{Agent},s,t) \land \mathrm{Breeze}(t) \Rightarrow \mathrm{Breezy}(s)$

  - *Squares are breezy near a pit:*

    $\forall s\ \mathrm{Breezy}(s) \Leftrightarrow \exists\ r\ \mathrm{Adjacent}(r,s) \land \mathrm{Pit}(r)$

- Important: in FOL we <u>quantify over time</u>

  $\forall t\ \mathrm{HaveArrow}(t+1) \Leftrightarrow (\mathrm{HaveArrow}(t) \land \neg\ \mathrm{Action}(\mathrm{Shoot},t))$

Knowledge engineering in FOL

1. Identify the task
2. Assemble the relevant knowledge
3. Decide on a vocabulary of predicates, functions, and constants
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

Knowledge Engineering in FOL

- Encode the following facts:
  - Some honors students take AI in Spring 2020
  - Every honors student that takes AI passes it
  - Only one honors student took Algorithms in Spring 2020
  - The worst score in AI is always higher than the best score in Algorithms
- *First decide on the vocabulary:*
  - Takes (x, c, d) means "*honors student x takes course c in time interval d*"
  - Spring (y) means "*spring of year y*"
  - Passes (x, c) means "*honors student x passes course c*"
  - Score (x, c) means "*the score obtained by the honors student x in course c*"

Encoding in FOL

- *Some honors students take AI in Spring 2020*

  $\exists x$ takes (x, AI, Spring(2020))

- *Every honors student that takes AI passes it*

  $\forall x\ \forall d$ takes (x, AI, d) $\Rightarrow$ Passes(x,AI)

- *Only one honors student took Algorithms in Spring 2020*

  $\exists$ x $\forall$ y takes(x, Algorithms, Spring(2020)) ^ takes(y, Algorithms, Spring(2020)) $\Rightarrow$ x = y

- *The worst score in AI is always higher than the best score in Algorithms*

  $\exists$ x $\forall$ y Score(x, AI) > Score(y, Algorithms)

More Knowledge Engineering in FOL

- Encode the following facts:
  - *Every student that knows programming is smart*
  - *No youngster buys an expensive car*
  - *Actors can interpret some characters all the time, and sometimes actors can interpret any character but they cannot interpret all characters all the time*
  - First decide on the vocabulary:
    - Student (x) means " *x is a student* "
    - Knows (x,y) means " *x knows y* "
    - Smart (x) means " *x is smart* "
    - Youngster (x) means " *x is a youngster* "
    - Buys (x,y) means " *x buys y* "
    - Car (x) means " *x is a car* "
    - Expensive (x) means " *x is expensive* "
    - Actor (x) means " *x is an actor* "
    - Character (x) means "*x is a character (in a play)* "
    - Plays (x, c,t) means "*x plays the character c at time t* "

Encoding in FOL

- *Every student that knows programming is smart*

  $\forall x$ student(x) $\wedge$ knows(x, programming) $\Rightarrow$ smart(x)

- *No youngster buys an expensive car*

  $\forall x,y$ Youngster(x) $\wedge$ Car(y) $\wedge$ Expensive(y) $\Rightarrow$ $\neg$Buys(x,y)

- *Actors can interpret some characters all the time, and sometimes actors can interpret any character but they cannot interpret all characters all the time*

$\forall x$ Actor(x) $\Rightarrow$ ($\exists y$ $\forall t$ Character(y) $\wedge$ plays(x, y, t)) $\wedge$

$\wedge$ ($\exists t$ $\forall y$ Character(y) $\Rightarrow$ plays(x,y,t)) $\wedge$

$\wedge$ $\neg$ ($\forall t$ $\forall y$ Character(y) $\Rightarrow$ plays(x,y,t))