First-Order Logic

Outline

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

Prop Logic: Wumpus World

- Model the Physics:
  - breeze $B_{x,y} \Rightarrow (P_{x,y+1} \vee P_{x,y-1} \vee P_{x+1,y} \vee P_{x-1,y})$
  - stench $S_{x,y} \Rightarrow (W_{x,y+1} \vee W_{x,y-1} \vee W_{x+1,y} \vee W_{x-1,y})$
  - one wumpus

    at least one: $W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,3} \vee W_{4,4}$

Problems?

- Physics about breezes and stenches for every single square $B_{1,1}$ $B_{1,2}$ $B_{2,1}$…
- Prefer to have two sentences to say how breezes arise in all squares; e.g.

  $\forall s\ Breezy(s) \Rightarrow \exists r\ Adjacent(r,s) \land Pit(r)$

  $\forall s\ \neg Breezy(s) \Rightarrow \neg\exists r\ Adjacent(r,s) \land Pit(r)$

  or, $\forall s\ Breezy(s)$ **ó** $\exists r\ Adjacent(r,s) \land Pit(r)$

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, father, bigger than, part of, comes between, …
  - Functions: father of, best friend, one more than, plus, …

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*$_1$,...,*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

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

E.g.

Sibling(KingJohn,Richard) $\Rightarrow$ *Sibling(Richard,KingJohn)*

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

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

User provides

- **Constant symbols,** which represent individuals in the world
  - Mary
  - 3
  - Green
- **Function symbols,** which map individuals to individuals
  - father-of(Mary) = John
  - color-of(Sky) = Blue
- **Predicate symbols,** which map individuals to truth values
  - greater(5,3)
  - green(Grass)
  - color(Grass, Green)

FOL Provides

- **Variable symbols**
  - E.g., x, y, foo
- **Connectives**
  - Same as in PL: not (¬), and (∧), or (∨), implies (→), if and only if (biconditional ↔)
- **Quantifiers**
  - Universal " **x** or **(Ax)**
  - Existential $**x** or **(Ex)**

Sentences are built from terms and atoms

- A **term** (denoting a real-world individual) is a constant symbol, a variable symbol, or an n-place function of n terms.

  x and f(x_1, ..., x_n) are terms, where each x_i is a term.

  A term with no variables is a **ground term**
- An **atomic sentence** (which has value true or false) is an n-place predicate of n terms
- A **complex sentence** is formed from atomic sentences connected by the logical connectives:

  $\neg$P, P$\vee$Q, P$\wedge$Q, P$\rightarrow$Q, P$\leftrightarrow$Q where P and Q are sentences
- A **quantified sentence** adds quantifiers $\forall$ and $\exists$
- A **well-formed formula (wff)** is a sentence containing no "free" variables. That is, all variables are "bound" by universal or existential quantifiers.

  ($\forall$x)P(x,y) has x bound as a universally quantified variable, but y is free.

A BNF for FOL

```
S := <Sentence> ;
<Sentence> := <AtomicSentence> |
          <Sentence> <Connective> <Sentence> |
          <Quantifier> <Variable>,... <Sentence> |
          "NOT" <Sentence> |
          "(" <Sentence> ")";
<AtomicSentence> := <Predicate> "(" <Term>, ... ")" |
                    <Term> "=" <Term>;
<Term> := <Function> "(" <Term>, ... ")" |
          <Constant> |
          <Variable>;
<Connective> := "AND" | "OR" | "IMPLIES" | "EQUIVALENT";
<Quantifier> := "EXISTS" | "FORALL" ;
<Constant> := "A" | "X1" | "John" | ... ;
<Variable> := "a" | "x" | "s" | ... ;
<Predicate> := "Before" | "HasColor" | "Raining" | ... ;
<Function> := "Mother" | "LeftLegOf" | ... ;
```

Semantics of FOL

- **Domain M:** the set of all objects in the world (of interest)
- **Interpretation I:** includes
  - Assign each constant to an object in M
  - Define each function of n arguments as a mapping M^n => M
  - Define each predicate of n arguments as a mapping M^n => {T, F}
  - Therefore, every ground predicate with any instantiation will have a truth value
  - In general there is an infinite number of interpretations because |M| is infinite
- **Define logical connectives: ~, ^, v, =>, <=>** as in PL
- **Define semantics of (" x) and ($x)**
  - (∀x) P(x) is true iff P(x) is true under all interpretations
  - (∃x) P(x) is true iff P(x) is true under some interpretation

- **Model:** an interpretation of a set of sentences such that every sentence is *True*
- **A sentence is**
  - **satisfiable** if it is true under some interpretation
  - **valid** if it is true under all possible interpretations
  - **inconsistent** if there does not exist any interpretation under which the sentence is true
- **Logical consequence:** S |= X if all models of S are also models of X

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

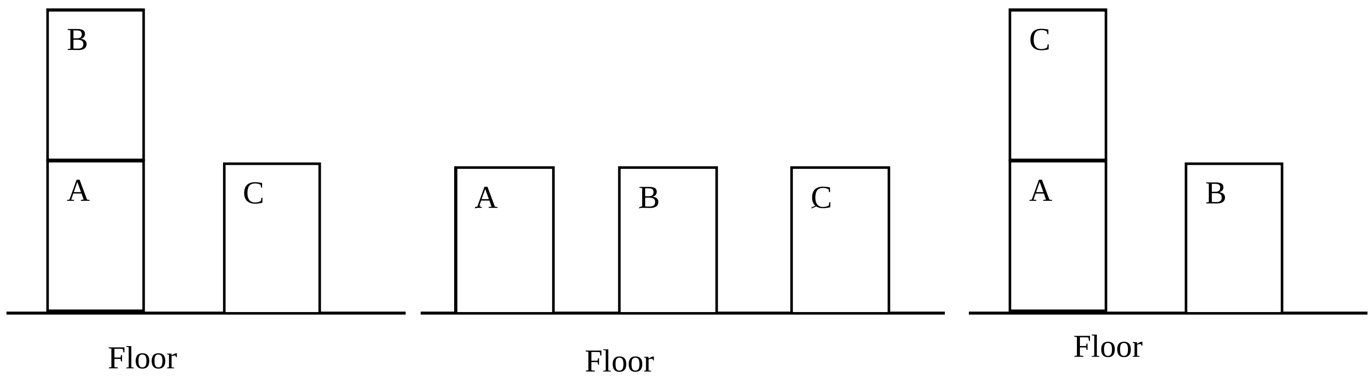
  function symbols → functional relations

- An atomic sentence $predicate(term_1,\ldots,term_n)$ is true iff the objects referred to by $term_1,\ldots,term_n$ are in the relation referred to by *predicate*

Truth in first-order logic

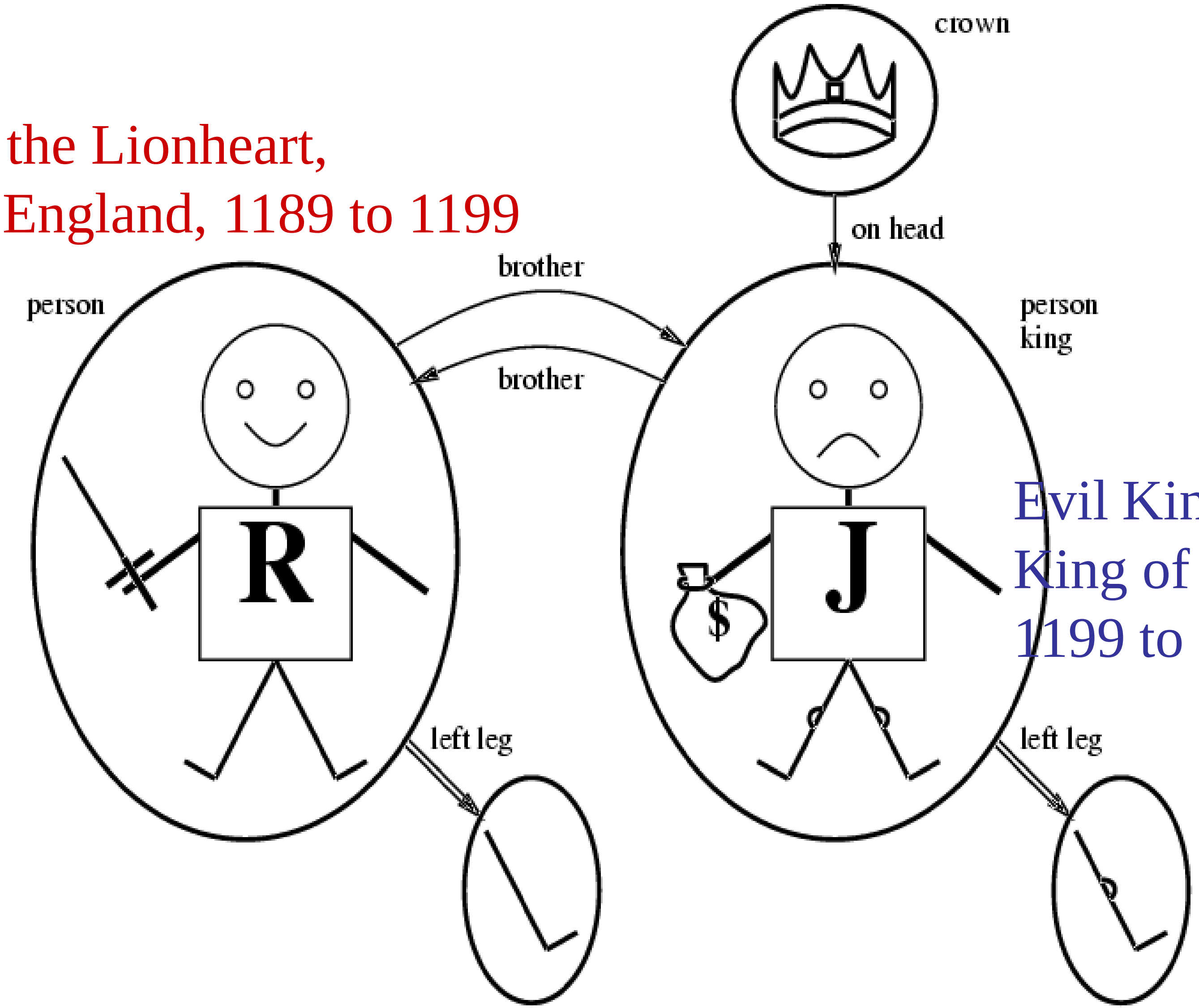
On(A,Fl) É Clear(B)
Clear(B) Ù Clear(C) É On(A,Fl)
Clear(B) Ú Clear(A)
Clear(B)
Clear(C)

Truth in first-order logic



Models for FOL: Example

Richard the Lionheart,
King of England, 1189 to 1199



Evil King John,
King of England,
1199 to 1215

FOL Representation

- Brother(Richard,John)
- Married(FatherOf(Richard),MotherOf(John))
- $\neg$ Brother(LeftLegOf(Richard),John)
- Brother(Richard,John) $\wedge$ Brother(John,Richard)
- King(Richard) $\vee$ King(John)
- $\neg$ King(Richard) $\Rightarrow$ King(John)
- $\forall$ x, King(x) $\Rightarrow$ Person(x)
- $\exists$ x, Crown(x) $\wedge$ OnHead(x,John)

Convention: *variables* in lower case, everything else in UpperCase.

Examples

- All crows are black.

  $\forall$ x Crow(x) => Black(x)

- Mary likes the color of one of John's ties

  $\exists$ x Like(*Mary*, color(*x*)) $\wedge$ Tie(*x*) $\wedge$ Owner(*x*,*John*)

Limitations of Prop Logic

- Cannot draw connections or refer to individuals
  - P1: Paul is tall
  - P2: Barbara is short
  - P3: All tall people bang their heads in the Tokyo subway station.

what can be inferred?

FOL: Subway example

- Can draw connections and refer to individuals:
  - P1: Tall(Paul).
  - P2: $\neg$ Tall(Barbara).
  - P3: $\forall$ x, Tall(x) =>BangHead(x,TokyoSubway)

Able to draw inference that Paul will bang his head on the Tokyo Subway.

Quantifiers

- **Universal quantification**
  - (" x)P(x) means that P holds for **all** values of x in the domain associated with that variable
  - E.g., (" x) dolphin(x) $\rightarrow$ mammal(x)
- **Existential quantification**
  - ($ x)P(x) means that P holds for **some** value of x in the domain associated with that variable
  - E.g., ($ x) mammal(x) $\wedge$ lays-eggs(x)
  - Permits one to make a statement about some object without naming it

Quantifiers

- Universal quantifiers are often used with “implies” to form “rules”:

  $(\forall x)$ student(x) $\rightarrow$ smart(x) means “All students are smart”
- Universal quantification is *rarely* used to make blanket statements about every individual in the world:

  $(\forall x)$student(x)$\wedge$smart(x) means “Everyone in the world is a student and is smart”
- Existential quantifiers are usually used with “and” to specify a list of properties about an individual:

  $(\exists x)$ student(x) $\wedge$ smart(x) means “There is a student who is smart”
- A common mistake is to represent this English sentence as the FOL sentence:

  $(\exists x)$ student(x) $\rightarrow$ smart(x)

  – But what happens when there is a person who is *not* a student?

Quantifier Scope

- Switching the order of universal quantifiers *does not* change the meaning:
  - $(\forall x)(\forall y)P(x,y) \leftrightarrow (\forall y)(\forall x)\ P(x,y)$
- Similarly, you can switch the order of existential quantifiers:
  - $(\exists x)(\exists y)P(x,y) \leftrightarrow (\exists y)(\exists x)\ P(x,y)$
- Switching the order of universals and existentials *does* change meaning:
  - Everyone likes someone: $(\forall x)(\exists y)$ likes(x,y)
  - Someone is liked by everyone: $(\exists y)(\forall x)$ likes(x,y)

Connections between All and Exists

We can relate sentences involving $\forall$ and $\exists$ using De Morgan's laws:

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

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

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

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

Universal quantification

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

Everyone at IT is smart:
$\forall x\ At(x,IT) \Rightarrow Smart(x)$

- $\forall x\ P$ 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*

$$
\begin{array}{ll}
 & At(Sami, IT) \Rightarrow Smart(Sami) \\
\wedge & At(John, IT) \Rightarrow Smart(John) \\
\wedge & \ldots \\
\wedge \ldots &
\end{array}
$$

A common mistake to avoid

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

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

means “Everyone is at IT and everyone is smart”

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

Existential quantification

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

Someone at IT is smart:

$\exists x$ At(x,IT) $\wedge$ 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(Sami,IT) $\wedge$ Smart(Sami)
  $\vee$ At(John,IT) $\wedge$ Smart(John)
  $\vee$ . . .
  $\vee$ ...

Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$

- Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

$$\exists x \text{ At}(x,IT) \Rightarrow \text{Smart}(x)$$

is true if there is anyone who is not at IT!

$$\exists x \text{ At}(x,IT) \wedge \text{Smart}(x)$$

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*:

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

Translating English to FOL

- Every gardener likes the sun.
  `(∀ x) gardener(x) ⇒ likes(x,Sun)`

- You can fool some of the people all of the time.
  `(∃ x)(∀ t) (person(x) ^ time(t)) ⇒ can-fool(x,t)`

- You can fool all of the people some of the time.
  `(∀ x)(∃ t) (person(x) ^ time(t) ⇒ can-fool(x,t)`

- All purple mushrooms are poisonous.
  `(∀ x) (mushroom(x) ^ purple(x)) ⇒ poisonous(x)`

Translating English to FOL…

- No purple mushroom is poisonous.
  ¬(∃ x) purple(x) ^ mushroom(x) ^ poisonous(x)
  or, equivalently,
  (∀ x) (mushroom(x) ^ purple(x)) ⇒ ~poisonous(x)

- There are exactly two purple mushrooms.
  (∃ x)(∃ y) mushroom(x) ^ purple(x) ^ mushroom(y) ^ purple(y) ^ ¬(x=y) ^ (∀ z) (mushroom(z) ^ purple(z)) ⇒ ((x=z) v (y=z))

- Deb is not tall.
  ¬ tall(Deb)

- X is above Y if X is on directly on top of Y or else there is a pile of one or more other objects directly on top of one another starting with X and ending with Y.
  (∀ x)(∀ y) above(x,y) <=> (on(x,y) v (∃ z) (on(x,z) ^ above(z,y)))

Using FOL

The kinship domain:

- Brothers are siblings

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

- One's mother is one's female parent

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

- "Sibling" is symmetric

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

Using FOL

All packets in room 27 are smaller than any packets in room 28

(" x,y) {[Package(x) ÙPackage(y) ÙInroom(x,27) ÙInroom(y,28] ÉSmaller(x,y)}

Using FOL

All packets in room 27 are smaller than some packet in room 28

Using FOL

$$(\exists y)(\forall x)\ \{[Package(x) \wedge Package(y) \wedge Inroom(x,27) \wedge Inroom(y,29] \supset Smaller(x,y)\}$$

Using FOL

(" x)($y) {[Package(x) ÙPackage(y) ÙInroom(x,27) ÙInroom(y,29] ÉSmaller(x,y)}

Example: A simple genealogy KB by FOL

- **Build a small genealogy knowledge base using FOL that**
  - contains facts of immediate family relations (spouses, parents, etc.)
  - contains definitions of more complex relations (ancestors, relatives)
  - is able to answer queries about relationships between people
- **Predicates:**
  - parent(x, y), child(x, y), father(x, y), daughter(x, y), etc.
  - spouse(x, y), husband(x, y), wife(x,y)
  - ancestor(x, y), descendant(x, y)
  - male(x), female(y)
  - relative(x, y)
- **Facts:**
  - husband(Joe, Mary), son(Fred, Joe)
  - spouse(John, Nancy), male(John), son(Mark, Nancy)
  - father(Jack, Nancy), daughter(Linda, Jack)
  - daughter(Liz, Linda)
  - etc.

- **Rules for genealogical relations**
  - $(\forall x,y)$ parent(x, y) $\leftrightarrow$ child (y, x)
    $(\forall x,y)$ father(x, y) $\leftrightarrow$ parent(x, y) $\wedge$ male(x) (similarly for mother(x, y))
    $(\forall x,y)$ daughter(x, y) $\leftrightarrow$ child(x, y) $\wedge$ female(x) (similarly for son(x, y))
  - $(\forall x,y)$ husband(x, y) $\leftrightarrow$ spouse(x, y) $\wedge$ male(x) (similarly for wife(x, y))
    $(\forall x,y)$ spouse(x, y) $\leftrightarrow$ spouse(y, x) (**spouse relation is symmetric**)
  - $(\forall x,y)$ parent(x, y) $\rightarrow$ ancestor(x, y)
    $(\forall x,y)(\exists z)$ parent(x, z) $\wedge$ ancestor(z, y) $\rightarrow$ ancestor(x, y)
  - $(\forall x,y)$ descendant(x, y) $\leftrightarrow$ ancestor(y, x)
  - $(\forall x,y)(\exists z)$ ancestor(z, x) $\wedge$ ancestor(z, y) $\rightarrow$ relative(x, y)
    (related by common ancestry)
    $(\forall x,y)$ spouse(x, y) $\rightarrow$ relative(x, y) (related by marriage)
    $(\forall x,y)(\exists z)$ relative(z, x) $\wedge$ relative(z, y) $\rightarrow$ relative(x, y) (**transitive**)
    $(\forall x,y)$ relative(x, y) $\leftrightarrow$ relative(y, x) (**symmetric**)
- **Queries**
  - ancestor(Jack, Fred) /* the answer is yes */
  - relative(Liz, Joe) /* the answer is yes */
  - relative(Nancy, Matthew)
    /* no answer in general, no if under closed world assumption */
  - $(\exists z)$ ancestor(z, Fred) $\wedge$ ancestor(z, Liz)

Axioms for Set Theory in FOL

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

   $\forall s\ \text{set}(s) \Leftrightarrow (s=\text{EmptySet}) \lor (\exists x,r\ \text{Set}(r) \wedge s=\text{Adjoin}(s,r))$

2. The empty set has no elements adjoined to it:

   $\sim \exists x,s\ \text{Adjoin}(x,s)=\text{EmptySet}$

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

   $\forall x,s\ \text{Member}(x,s) \Leftrightarrow s=\text{Adjoin}(x,s)$

4. The only members of a set are the elements that were adjoined into it:

   $\forall x,s\ \text{Member}(x,s) \Leftrightarrow \exists y,r\ (s=\text{Adjoin}(y,r) \wedge (x=y \lor \text{Member}(x,r)))$

5. A set is a subset of another iff all of the 1st set's members are members of the 2nd:

   $\forall s,r\ \text{Subset}(s,r) \Leftrightarrow (\forall x\ \text{Member}(x,s) \Rightarrow \text{Member}(x,r))$

6. Two sets are equal iff each is a subset of the other:

   $\forall s,r\ (s=r) \Leftrightarrow (\text{subset}(s,r) \wedge \text{subset}(r,s))$

7. Intersection

   $\forall x,s1,s2\ \text{member}(X,\text{intersection}(S1,S2)) \Leftrightarrow \text{member}(X,s1) \wedge \text{member}(X,s2)$

8. Union

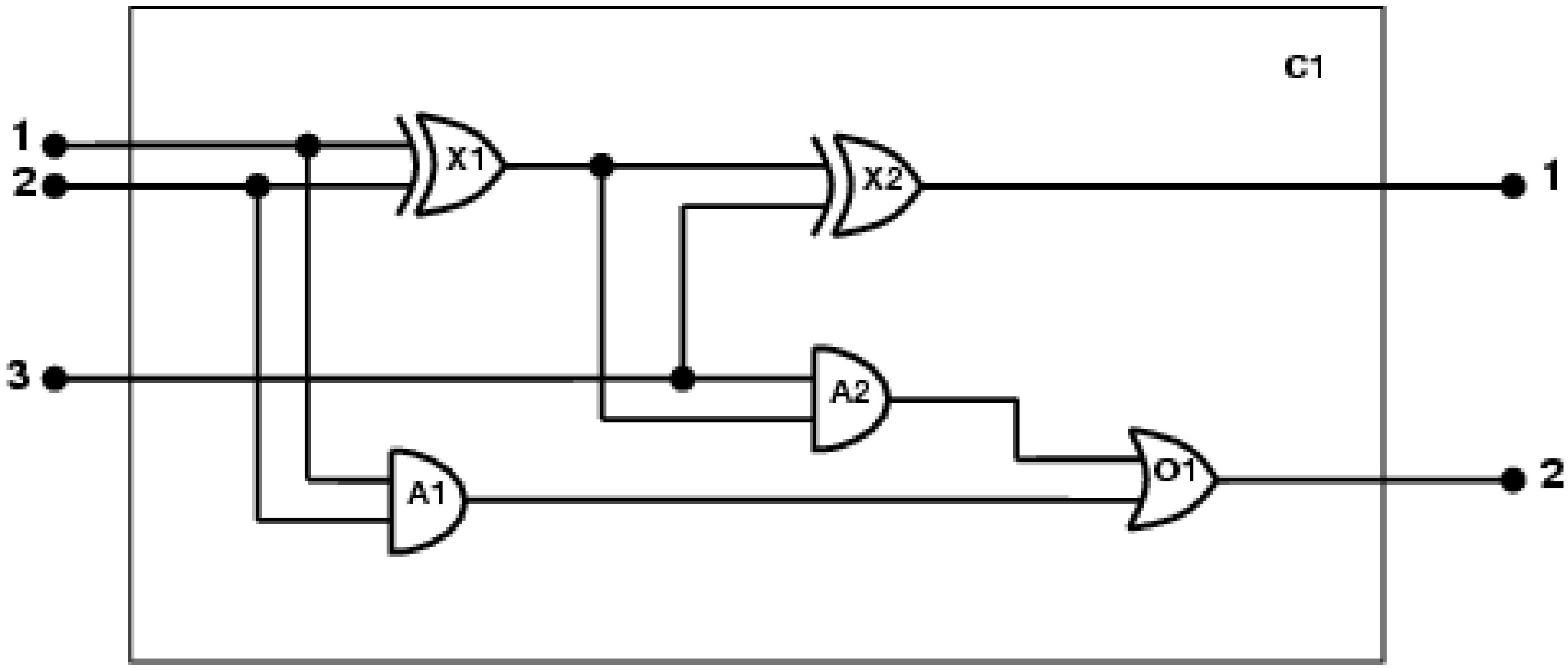
   $\exists x,s1,s2\ \text{member}(X,\text{union}(s1,s2)) \Leftrightarrow \text{member}(X,s1) \lor \text{member}(X,s2)$

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

The electronic circuits domain

One-bit full adder



The electronic circuits domain

1. Identify the task
   – Does the circuit actually add properly? (circuit verification)

2. Assemble the relevant knowledge
   – Composed of wires and gates; Types of gates (AND, OR, XOR, NOT)
   – Irrelevant: size, shape, color, cost of gates
3. Decide on a vocabulary
   – Alternatives:

     $\text{Type}(X_1) = \text{XOR}$

     $\text{Type}(X_1, \text{XOR})$

     $\text{XOR}(X_1)$

The electronic circuits domain

4. Encode general knowledge of the domain
   - $\forall t_1,t_2\ Connected(t_1, t_2) \Rightarrow Signal(t_1) = Signal(t_2)$
   - $\forall t\ Signal(t) = 1 \vee Signal(t) = 0$
   - $1 \neq 0$
   - $\forall t_1,t_2\ Connected(t_1, t_2) \Rightarrow Connected(t_2, t_1)$
   - $\forall g\ Type(g) = OR \Rightarrow Signal(Out(1,g)) = 1 \Leftrightarrow \exists n\ Signal(In(n,g)) = 1$
   - $\forall g\ Type(g) = AND \Rightarrow Signal(Out(1,g)) = 0 \Leftrightarrow \exists n\ Signal(In(n,g)) = 0$
   - $\forall g\ Type(g) = XOR \Rightarrow Signal(Out(1,g)) = 1 \Leftrightarrow Signal(In(1,g)) \neq Signal(In(2,g))$
   - $\forall g\ Type(g) = NOT \Rightarrow Signal(Out(1,g)) \neq Signal(In(1,g))$

The electronic circuits domain

5. Encode the specific problem instance

$Type(X_1) = XOR$ $\quad$ $Type(X_2) = XOR$

$Type(A_1) = AND$ $\quad$ $Type(A_2) = AND$

$Type(O_1) = OR$

$Connected(Out(1,X_1),In(1,X_2))$ $\quad$ $Connected(In(1,C_1),In(1,X_1))$

$Connected(Out(1,X_1),In(2,A_2))$ $\quad$ $Connected(In(1,C_1),In(1,A_1))$

$Connected(Out(1,A_2),In(1,O_1))$ $\quad$ $Connected(In(2,C_1),In(2,X_1))$

$Connected(Out(1,A_1),In(2,O_1))$ $\quad$ $Connected(In(2,C_1),In(2,A_1))$

$Connected(Out(1,X_2),Out(1,C_1))$ $\quad$ $Connected(In(3,C_1),In(2,X_2))$

$Connected(Out(1,O_1),Out(2,C_1))$ $\quad$ $Connected(In(3,C_1),In(1,A_2))$

The electronic circuits domain

6. Pose queries to the inference procedure

   What are the possible sets of values of all the terminals for the adder circuit?

   $\exists i_1,i_2,i_3,o_1,o_2$ Signal(In(1,C1)) = i_1 $\wedge$ Signal(In(2,C_1)) = i_2 $\wedge$ Signal(In(3,C_1)) = i_3 $\wedge$ Signal(Out(1,C_1)) = o_1 $\wedge$ Signal(Out(2,C_1)) = o_2

7. Debug the knowledge base

   May have omitted assertions like $1 \neq 0$

Summary

- First-order logic:
  - objects and relations are semantic primitives
  - syntax: constants, functions, predicates, equality, quantifiers

- Increased expressive power: sufficient to define wumpus world