ARTIFICIAL INTELLIGENCE

2023/2024 Semester 2

First-Order Logic: Chapter 8

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

The Others

Language	Ontological Commitment (What exists in the world)	Epistemological Commitment (What an agent believes about facts)
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 $\in [0, 1]$
Fuzzy logic	facts with degree of truth $\in [0, 1]$	known interval value

Limitations of propositional logic

- So far we studied propositional logic
- Some English statements are hard to model in propositional logic:
- “If your roommate is wet because of rain, your roommate must not be carrying **any** umbrella”
- Pathetic attempt at modeling this:
- RoommateWetBecauseOfRain => (NOT(RoommateCarryingUmbrella0) AND NOT(RoommateCarryingUmbrella1) AND NOT(RoommateCarryingUmbrella2) AND …)

Problems with propositional logic

- propositional logic assumes the world consists of facts
- No notion of objects
- No notion of relations among objects
- RoommateCarryingUmbrella0 is instructive **to us**, suggesting
  - there is an object we call Roommate,
  - there is an object we call Umbrella0,
  - there is a relationship Carrying between these two objects
- Formally, none of this meaning is there这一切都是有意义的
  - Might as well have replaced RoommateCarryingUmbrella0 by P

Elements of first-order logic

- Objects: can give these names such as Umbrella0, Person0, John, Earth, wheel, door, body …
- Relations: Carrying(., .), IsAnUmbrella(.)
  - Carrying(Person0, Umbrella0), IsUmbrella(Umbrella0), brother of, bigger than, part of,…
  - Relations with one object = unary relations = properties such as red, round, prime,
- Functions: Roommate(.), ColorOf(.), father of, best friend, one more than, plus, …
  - Roommate(Person0), ColorOf(car)
- Equality: Roommate(Person0) = Person1

Semantics

there is a correspondence between

- functions, which return values
- predicates, which are true or false

Function: father_of(Mary) = Bill

Predicate: father_of(Mary, Bill)

Functions are relations with single value for each object

Some examples:

- "One plus two equals three"
  - Objects: one, two, three, one plus two
  -
  - Relations: equals
  - Functions: plus
- "Squares neighboring the wumpus are smelly"
  - Objects: wumpus, squares
  - Property: smelly
  -
  - Relations: neighboring

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)*

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

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

Syntax of FOL

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

Syntax of Propositional Logic

$$\begin{array}{rcl}
Sentence & \rightarrow & AtomicSentence \mid ComplexSentence \\
AtomicSentence & \rightarrow & \mathbf{True} \mid \mathbf{False} \mid \mathrm{Symbol} \\
Symbol & \rightarrow & P \mid \mathbf{Q} \mid \mathbf{R} \mid \ldots \\
ComplexSentence & \rightarrow & \neg\ Sentence \\
& | & (\ Sentence \wedge Sentence\ ) \\
& | & (\ Sentence \vee Sentence\ ) \\
& | & (\ Sentence \Rightarrow Sentence\ ) \\
& | & (\ Sentence \Leftrightarrow Sentence\ )
\end{array}$$

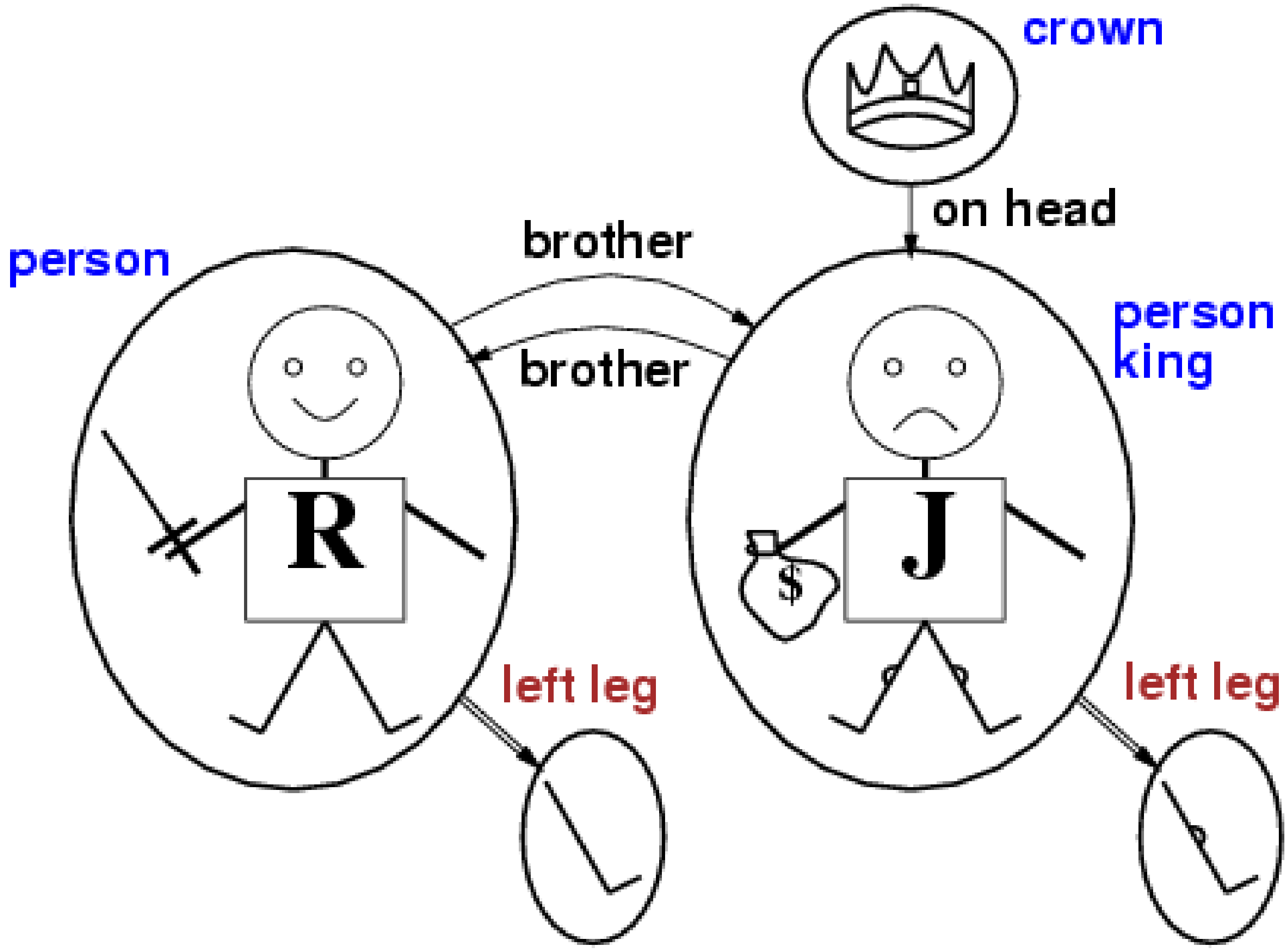
Truth in first-order logic

- 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*

Models for FOL: Example



Models for FOL: Example

- Five objects: Richard the Lionheart, the evil king John, the left legs of Richard and John, and a crown
- Two binary relations: brother, on head
- Three unary relations: person, king, crown
- One unary function: left-leg

Universal quantification

- $\forall$<*variables*> <*sentence*>   **$\forall$x P(x)**

  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*
-

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

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”

Existential quantification

- $\exists$*<variables> <sentence>* **$\exists$x P(x)**

- Someone at NUS is smart:
- $\exists x\ \mathrm{At}(x, \mathrm{NUS}) \land \mathrm{Smart}(x)\$$
- $\exists x\ P$ 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
-

$$
\begin{aligned}
& \mathrm{At}(\mathrm{KingJohn}, \mathrm{NUS}) \land \mathrm{Smart}(\mathrm{KingJohn}) \\
\lor\ & \mathrm{At}(\mathrm{Richard}, \mathrm{NUS}) \land \mathrm{Smart}(\mathrm{Richard}) \\
\lor\ & \mathrm{At}(\mathrm{NUS}, \mathrm{NUS}) \land \mathrm{Smart}(\mathrm{NUS}) \\
\lor\ & \ldots
\end{aligned}
$$

Another common mistake to avoid

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

$$\exists x \; At(x, NUS) \Rightarrow Smart(x)$$

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

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（量词对偶）: 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)

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., Father(John) = Henry
- E.g., definition of *Sibling* in terms of *Parent*:
-

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

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 (m, c))*

- “Sibling” is symmetric
-

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

Using FOL

The set domain:

- $\forall s\ Set(s) \Leftrightarrow (s = \{\}\ ) \vee (\exists x, s_2\ Set(s_2) \wedge s = \{x|s_2\})$
- $\neg\exists x, s\ \ \{x|s\} = \{\}$
- $\forall x, s\ \ \ x \in s \Leftrightarrow s = \{x|s\}$
- $\forall x, s\ \ x \in s \Leftrightarrow [\ \exists y, s_2\ (s = \{y|s_2\} \wedge (x = y \vee x \in s_2))]$
- $\forall s_1, s_2\ \ s_1 \subseteq s_2 \Leftrightarrow (\forall x\ x \in s_1 \Rightarrow x \in s_2)$
- $\forall s_1, s_2\ \ (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$
- $\forall x, s_1, s_2\ \ x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$
- $\forall x, s_1, s_2\ \ x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$

Why "First order"?

- FOL permits quantification over variables
- Higher order logics permit quantification over functions and predicates:

$$\forall P, x\ [P(x) \vee \neg P(x)]$$

$$\forall x, y\ (x{=}y) \Leftrightarrow [\forall P\ (P(x) \Leftrightarrow P(y))]$$

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 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,
- $S\sigma$ denotes the result of plugging σ into S; e.g.,

  S = Smarter(x, y)

  σ = {x/Hillary, y/Bill}

  $S\sigma$ = Smarter(Hillary, Bill)

- `Ask` (KB, S) returns some/all σ such that KB $\models \sigma$
-

Knowledge base for the wumpus world

- Perception
  - $\forall t, s, b$ Percept ([s, b, Glitter], t) $\Rightarrow$ Glitter(t)
  -

- Reflex

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

- Environment:

  $\forall x, y, a, b$ *Adjacent*([x, y], [a, b]) $\Leftrightarrow$

  $[a, b] \in \{[x+1,y], [x-1,y],[x,y+1],[x,y-1]\}$

Deducing hidden properties

Properties of squares:

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

Squares are breezy near a pit:

- Diagnostic rule---infer cause from effect

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

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

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

- Causal rule---infer effect from cause

  $\forall r \ \text{Pit}(r) \Rightarrow [\forall s \ \text{Adjacent}(r, s) \Rightarrow \text{Breezy}(s) \ ]$

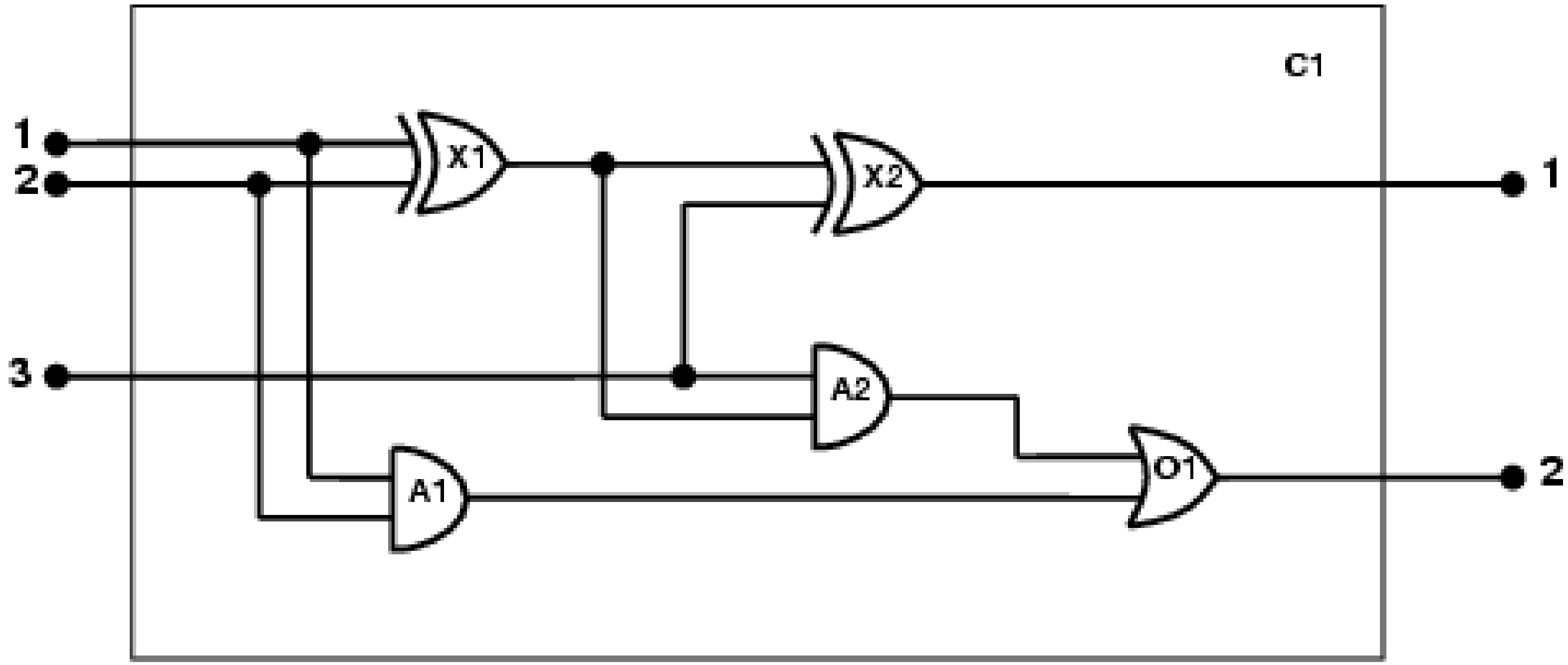
  $\forall s \ \ [\ \forall r \ \ \text{Adjacent}(r, s) \Rightarrow \neg \text{Pit}(r)] \Rightarrow \neg \text{Breezy}(s)$

Knowledge engineering in FOL

- Identify the task
- Assemble the relevant knowledge
- Decide on a vocabulary of predicates, functions, and constants
- Encode general knowledge about the domain
- Encode a description of the specific problem instance
- Pose queries to the inference procedure and get answers
- 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
   - Gate:

     $Type(X_1) = XOR$

     $Type(X_1, XOR)$

     $XOR(X_1)$
   - Terminal:

     $In(1, X_1)$

     $Connected(Out(1, X_1), In(1, X_2))$
   - Signal: Signal(t)

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 $\quad 1 \neq 0$
   - $\forall t_1,t_2$ 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

$Type(X_1) = XOR$ $Type(X_2) = XOR$
$Type(A_1) = AND$ $Type(A_2) = AND$
$Type(O_1) = OR$

$Connected(Out(1,X_1),In(1,X_2))$ $Connected(In(1,C_1),In(1,X_1))$
$Connected(Out(1,X_1),In(2,A_2))$ $Connected(In(1,C_1),In(1,A_1))$
$Connected(Out(1,A_2),In(1,O_1))$ $Connected(In(2,C_1),In(2,X_1))$
$Connected(Out(1,A_1),In(2,O_1))$ $Connected(In(2,C_1),In(2,A_1))$
$Connected(Out(1,X_2),Out(1,C_1))$ $Connected(In(3,C_1),In(2,X_2))$
$Connected(Out(1,O_1),Out(2,C_1))$ $Connected(In(3,C_1),In(1,A_2))$

The electronic circuits domain

6. Pose queries to the inference procedure

What combinations of inputs would cause the first output of C_1(the sum bit) to be 0 and the second output of C_1(the carry bit) to be 1?

$\exists i_1,i_2,i_3$ Signal(In(1, C_1)) = $i_1 \wedge$ Signal(In(2,C_1)) = $i_2 \wedge$ Signal(In(3,C_1)) = $i_3 \wedge$ Signal(Out(1,C_1)) = 0 $\wedge$ Signal(Out(2,C_1)) = 1

$\{i_1/1, i_2/1, i_3/0\}$ $\{i_1/1, i_2/0, i_3/1\}$ $\{i_1/0, i_2/1, i_3/1\}$

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

Questions?