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, …

Atomic sentences

Atomic sentence = *predicate* (*term*$_1$,...,*term*$_n$)
or *term*$_1$ = *term*$_2$

Term = *function* (*term*$_1$,...,*term*$_n$)
or *constant* or *variable*

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*

Universal quantification

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

Everyone at UNB is smart:
$\forall x$ At(x,UNB) $\Rightarrow$ 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*

At(John,UNB) $\Rightarrow$ Smart(John)
$\wedge$ At(Mary,UNB) $\Rightarrow$ Smart(Mary)
$\wedge$
$\wedge$ ...

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,UNB) $\wedge$ Smart(x)

  means “Everyone is at UNB and everyone is smart”

Existential quantification

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

- Someone at UNB is smart:
- $\exists x$ At(x,UNB) $\wedge$ 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(John,UNB) $\wedge$ Smart(John)
  $\vee$ At(Mary,UNB) $\wedge$ Smart(Mary)
  $\vee$ ...

Another common mistake to avoid

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

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

  is true if there is anyone who is **not** at UNB!

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"

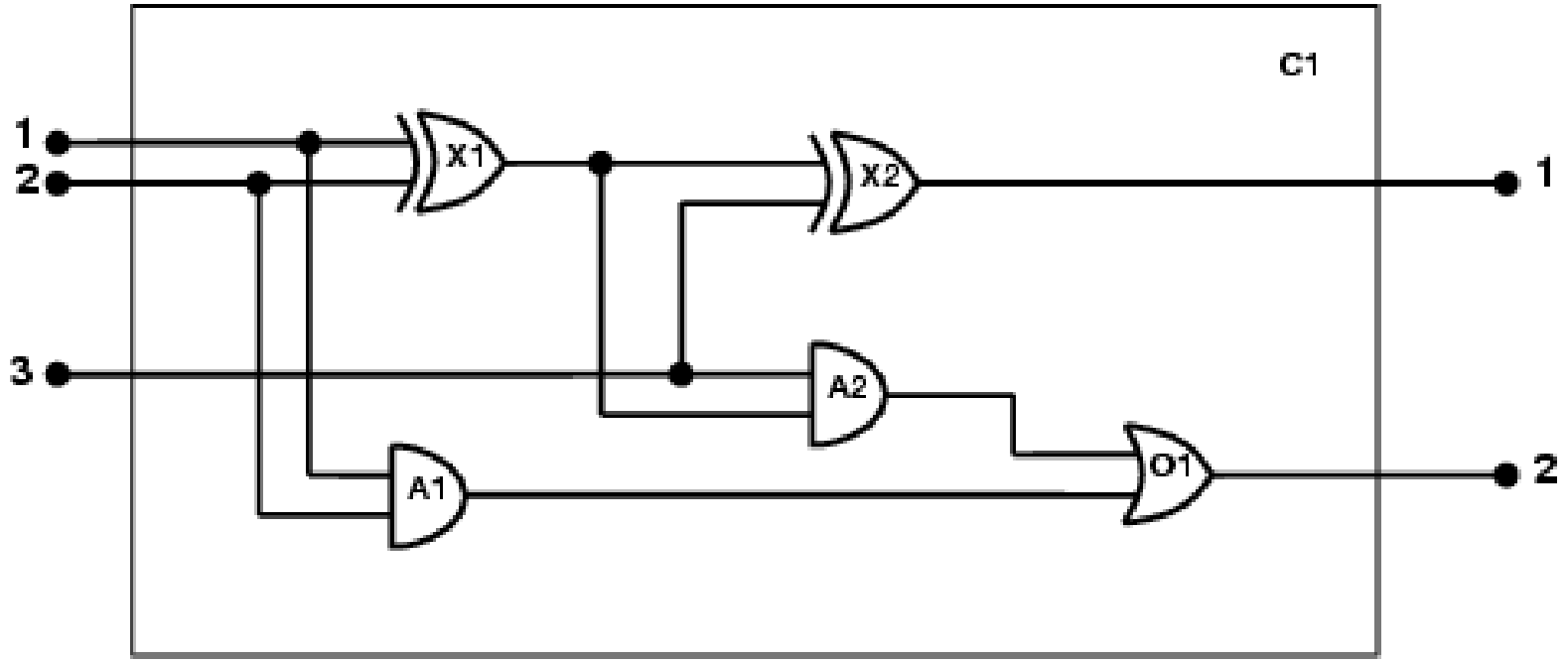
- Quantifier duality: each can be expressed using the other
- $\forall x$ Likes(x,IceCream) $\quad\neg\exists x\ \neg$Likes(x,IceCream)
- $\exists x$ Likes(x,Broccoli) $\quad\neg\forall x\ \neg$Likes(x,Broccoli)

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:

     Function: Type(X_1) = XOR

     Binary Predicate: Type(X_1, XOR)

     Individual type predicates: XOR(X_1)

The electronic circuits domain

4. Encode general knowledge of the domain

7 Rules describe all we need to know to make relevant queries!

- **If two terminals are connected, then they have the same signal**
  - $\forall t_1,t_2\ \text{Connected}(t_1, t_2) \Rightarrow \text{Signal}(t_1) = \text{Signal}(t_2)$

  The signal at every terminal is either 1 or 0, but not both
  - $\forall t\ \text{Signal}(t) = 1 \vee \text{Signal}(t) = 0$
  - $1 \neq 0$

  Connected is a commutative predicate
  - $\forall t_1,t_2\ \text{Connected}(t_1, t_2) \Rightarrow \text{Connected}(t_2, t_1)$

  An OR gate's output is 1 if and only if any of its inputs is 1
  - $\forall g\ \text{Type}(g) = \text{OR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \exists n\ \text{Signal}(\text{In}(n,g)) = 1$

  An AND gate's output is 0 if and only if any of its inputs is 0
  - $\forall g\ \text{Type}(g) = \text{AND} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 0 \Leftrightarrow \exists n\ \text{Signal}(\text{In}(n,g)) = 0$
  - ?
  - $\forall g\ \text{Type}(g) = \text{XOR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \text{Signal}(\text{In}(1,g)) \neq \text{Signal}(\text{In}(2,g))$

  ? $\forall g\ \text{Type}(g) = \text{NOT} \Rightarrow \text{Signal}(\text{Out}(1,g)) \neq \text{Signal}(\text{In}(1,g))$

The electronic circuits domain

An AND gate's output is 0 if and only if any of its inputs is 0

– $\forall g\ \text{Type}(g) = \text{AND} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 0 \Leftrightarrow \exists n\ \text{Signal}(\text{In}(n,g)) = 0$

An XOR gate's output is 1 of and only if its inputs are different

– $\forall g\ \text{Type}(g) = \text{XOR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \text{Signal}(\text{In}(1,g)) \neq \text{Signal}(\text{In}(2,g))$

A NOT gate's output is different from its input

– $\forall g\ \text{Type}(g) = \text{NOT} \Rightarrow \text{Signal}(\text{Out}(1,g)) \neq \text{Signal}(\text{In}(1,g))$

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 combination of inputs would cause the first output of C1 (the sum bit) to be 0 and the second output of C1 (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

Answer: ?

Answers

- Substitutions for the variables i_1, i_2, i_3 such that the resulting sentence is entailed by the KB
- Three possibilities:

1. $(i_1/1, i_2/1, i_3/0)$
2. $(i_1/1, i_2/0, i_3/1)$
3. $(i_1/0, i_2/1, i_3/1)$

More queries…

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,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)) = $o_1 \wedge$ Signal(Out(2,C_1)) = o_2

7. Debug the knowledge base

   May have omitted assertions like $1 \neq 0$