UNIVERSITÀ DEGLI STUDI DI NAPOLI
PARTHENOPE

Artificial Intelligence

First-Order Logic (Predicate Logic)

LESSON 14

prof. Antonino Staiano

M.Sc. In ''Machine Learning e Big Data'' - University Parthenope of Naples

Using FOL: Assertion and queries

- Let's use FOL in some domains
  - A domain is part of the world about which we want to express some knowledge
- To add an assertion in a KB, the TELL function can be used
  - *TELL(KB, King(John))*
  - *TELL(KB,Person(King))*
  - *TELL(KB, ∀x King(x) ⇒ Person(x))*
- For queries or goals, the ASK interface can be used
  - ASK(KB, King(John))
    - Any query that is logically entailed by the KB should be answered affirmatively
  - ASK(KB, ∃ x Person(x))
    - If we want to know what value x makes the sentence true, we get two possible answers
      - {x/John} and {x/Richard}
        - Substitution or binding list

The kinship domain

- The objects of the domain of family relationships are people
- Examples of facts
  - *Elizabeth is the mother of Charles*
  - *Charles is the father of William*
- Rules, e.g.
  - *One's grandmother is the mother of one's parent*
- Predicates
  - Unary: *Female and Male*
  - Binary: *Parent, Sibling, Brother, Sister, Child, Daughter; Spouse, Cousin, Aunt, Uncle, …*
  - Function: *Mother, Father*

Axioms in the Kinship Domain

- *One's husband is one's male spouse*
  - $\forall\, w, h\ Husband(h, w)\ \Leftrightarrow Male(h) \wedge Spouse(h, w)$
- *Parent and child are inverse relations*
  - $\forall\, p, c\ Parent(p, c)\ \Leftrightarrow Child(c, p)$
- *A grandparent is a parent of one's parent*
  - $\forall\ g, c\ Grandparent(g, c)\ \Leftrightarrow \exists p\ \ Parent(g, p) \wedge Parent(p, c)$
- *A sibling is another child of one's parent*
  - $\forall x, y\ Sibling(x, y)\ \Leftrightarrow (x \neq y) \wedge \exists p\ \ Parent(p, x) \wedge Parent(p, y)$
- Axioms provide the basic factual information from which useful conclusions can be derived
  - A kind of definitions: $\forall x, y\ P(x, y)\ \Leftrightarrow \ldots$

Theorems

- Some sentences about a domain are theorems rather than axioms
  - Entailed by axioms
- Example (assertion that siblinghood is symmetric):
  - $\forall x, y\ Sibling(x, y) \Leftrightarrow Sibling(y, x)$
    - a theorem that follows logically from the axiom defining the siblinghood
- Not all axioms are definition
  - Provide more general information about certain predicates without constituting a definition
- Axioms can also be plain facts
  - *Male(Jim)* and *Spouse(Jim, Laura)*
- Note that, when the expected answers are not forthcoming, this is a sign that an axiom is missing
  - From *Spouse(Jim, Laura)* one expects to infer *¬Spouse(George, Laura)* but this does not occur

The Wumpus World

- The Wumpus agent receives a percept vector with five elements
- The corresponding sentence stored in the KB must include both the percept and the time it occurred
  - The agent should know when it sees what
  - *Percept([Stench, Breeze, Glitter, None, None], 5)*
  - Percept is a binary predicate
  - Stench, Breeze … are constant
- Actions can be represented by logical terms
  - Turn(Right), Turn (Left), Forward, Shoot, Grab, Climb
- To determine which is the best the agent can ask
  - ASK(KB, BestAction(a,5)) which returns a binding list such as {a/Grab}

KB for the Wumpus World

- The raw percept data implies certain facts about the current state
  - ∀ b, g, t Percept([Smell, b, g], t) ⇒ Smelt(t)
  - ∀ s, b, t Percept([s, b, Glitter], t) ⇒ Glitter(t)
  - ∀ s, g, w, c, t Percept([s, Breeze, g, w, c], t) ⇒ Breeze(t)
- A simple reflex behavior can be expressed by quantified implication sentences, e.g.
  - ∀ t Glitter(t) ⇒ BestAction(Grab, t)

The Wumpus World: Environment

- Objects
  - Squares, pits, wumpus
    - For squares , we can use the list term [x,y]
- The adjacency of any two squares can be defined as
  - $\forall\, x, y, a, b\ Adjacent([x, y], [a, b]) \iff$
    $\big(x = a \land (y = b - 1 \lor y = b + 1)\big) \lor (y = b \land (x = a - 1 \lor x = a + 1))$
- We use a predicate Pit that is true for squares containing pits
- There is only one wumpus, so a constant Wumpus is enough
- The agent's location changes over time
  - *At(Agent, s, t)*
- The Wumpus is fixed to a specific location forever
  - $\forall\, t\ At(Wumpus, [1,3], t)$
- An object can be at only one location at a time
  - $\forall\, x, s_1, s_2, t\ At(\mathrm{x}, s_1, \mathrm{t}) \land At(\mathrm{x}, s_2, \mathrm{t}) \Rightarrow \mathrm{s}_1 = \mathrm{s}_2$

Deciding Hidden Properties

- Given the current location and the properties of its current percept, the agent infers the properties of the square
  - $\forall\, s,t\ At(\mathrm{Agent}, \mathrm{s}, \mathrm{t}) \land Breeze(\mathrm{t}) \Rightarrow Breezy(\mathrm{s})$
    - No time with Breezy
- The agent infers the cause from the effect
  - $\forall\, y\ Breezy(\mathrm{y}) \Rightarrow \exists\, x\ Pit(\mathrm{x}) \land Adjacent(\mathrm{x}, \mathrm{y})$
- … and the effect from the cause
  - $\forall\, x,y\ Pit(x) \land \mathrm{Adjacent}(\mathrm{x},\mathrm{y}) \Rightarrow Breezy(\mathrm{y})$
- If the agent discovered which places are breezy (or smelly) and, not breezy (or not smelly), it can deduce where the pits are (and where the Wumpus is)
  - $\forall\, s\ Breezy(\mathrm{s}) \Leftrightarrow \exists \mathrm{r}\ Adjacent(\mathrm{r},\mathrm{s}) \land Pit(\mathrm{r})$

Exercise 1

- Write out the axioms required for reasoning about the Wumpus's location, using a constant symbol Wumpus and a binary predicate At (Wumpus , Location)
  - Remember that there is only one Wumpus

Solution to Exercise 1

- $\forall s_1\ Smelly(s_1) \Leftrightarrow \exists s_2\ Adjacent(s_1,s_2) \wedge At(Wumpus,s_2)$

- $\exists s_1\ At(Wumpus,s_1) \wedge \forall s_2\ (s_1 \neq s_2) \Rightarrow \neg At(Wumpus,s_2)$

Exercise 2

- Arithmetic assertions can be written in first-order logic with the predicate symbol $<$, the function symbols $+$ and $\times$, and the constant symbols 0 and 1
- Additional predicates can also be defined with biconditionals
  1. Represent the property *"x is an even number."*
  2. Represent the property *"x is prime."*
  3. Goldbach's conjecture is the conjecture (unproven as yet) that *"every even number is equal to the sum of two primes"*
     - Represent this conjecture as a logical sentence

Solutions to the Exercise 2

- "x is an even number."
  - $\forall x \; Even(x) \Leftrightarrow \exists y \; x=y+y$

- "x is prime."
  - $\forall x \; Prime(x) \Leftrightarrow \forall y,z \;\; x=y \times z \Rightarrow y=1 \lor z=1$

- "every even number is equal to the sum of two primes."
  - $\forall x \; Even(x) \Rightarrow \exists y,z \; Prime(y) \land Prime(z) \land x=y+z$

Exercise 3

Assuming predicates *Parent(p,q)* and *Female(p)* and constants Joan and Kevin, with the obvious meanings, express each of the following sentences in first-order logic

- You may use the abbreviation $\exists^1$ to mean "there exists exactly one."

1. Joan has a daughter (possibly more than one, and possibly sons as well)
2. Joan has exactly one daughter (but may have sons as well).
3. Joan has exactly one child, a daughter
4. Joan and Kevin have exactly one child together
5. Joan has at least one child with Kevin, and no children with anyone else

Solutions to Exercise 3

- *Joan has a daughter* (possibly more than one, and possibly sons as well)
  - $\exists x \; Parent(Joan,x) \land Female(x)$
- *Joan has exactly one daughter* (but may have sons as well)
  - $\exists^1 x \; Parent(Joan,x) \land Female(x)$
- *Joan has exactly one child, a daughter*
  - $\exists x \; Parent(Joan,x) \land Female(x) \land [\forall y \; Parent(Joan,y) \Rightarrow y=x]$
- *Joan and Kevin have exactly one child together*
  - $\exists^1 c \; Parent(Joan,c) \land Parent(Kevin,c)$
- *Joan has at least one child with Kevin, and no children with anyone else*
  - $\exists c \; Parent(Joan,c) \land Parent(Kevin,c) \land \forall d,p \; [Parent(Joan,d) \land Parent(p,d)] \Rightarrow [p=Joan \lor p=Kevin]$

Knowledge Engineering

- Knowledge engineering is the process of constructing the KB
  - It consists of investigating a specific domain, identifying the relevant concepts (knowledge acquisition), and formally representing them
- This requires the interaction between
  - a domain expert (DE)
  - a knowledge engineer (KE), who is an expert in knowledge representation and inference, but usually not in the domain of interest
- A possible approach, suitable for special-purpose KBs (in predicate logic), is the following

Knowledge Engineering

1. Identify the task:
   - what range of queries will the KB support?
   - what kind of facts will be available for each problem instance?
2. Knowledge acquisition: eliciting from the domain expert the general knowledge about the domain (e.g., the rules of chess)
3. Choice of a vocabulary: what concepts must be represented as objects, predicates, or functions?
   - The result is the domain's ontology, which affects the complexity of the representation and the inferences that can be made
     - E.g., in the wumpus game, pits can be represented as objects or unary predicates on squares

Knowledge Engineering

4. Encoding the domain's general knowledge acquired in step 2 (this may require revising the vocabulary of step 3)
   - Axioms for all the vocabulary terms
5. Encoding a specific problem instance (e.g., a specific chess game)
   - Simple atomic sentences about instances of concepts from the ontology
6. Posing queries to the inference procedure and getting answers
   - Inference procedure applied to axioms and facts to derive new facts one is interested in
7. Debugging the KB, based on the results of step 6
   - Answers seldom correct on the first try, that is, if an axiom is missing some query won't be answerable from the KB

Knowledge Engineering in FOL

- The electronic circuits domain
  1. Identify the questions
     - Does the circuit in Figure 8.6 actually add properly?
     - If all the inputs are high, what is the output of gate A2?
     - Questions about the circuit's structure are also interesting
       - For example, what are all the gates connected to the first input terminal?
       - Does the circuit contain feedback loops?

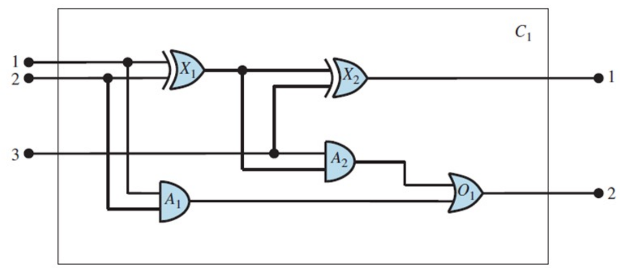
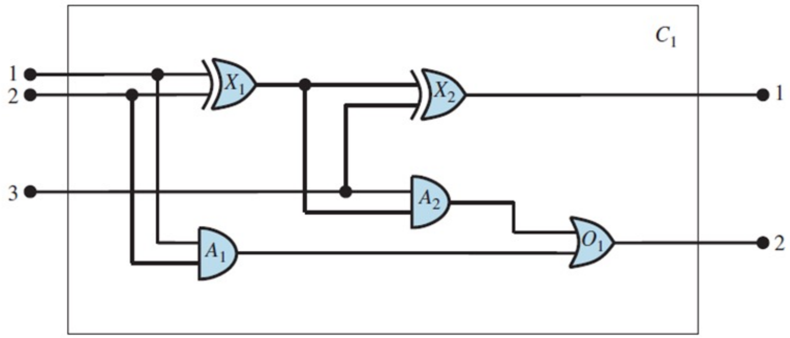


Figure 8.6 A digital circuit C_1, purporting to be a one-bit full adder. The first two inputs are the two bits to be added, and the third input is a carry bit. The first output is the sum, and the second output is a carry bit for the next adder. The circuit contains two XOR gates, two AND gates, and one OR gate.

Knowledge Engineering in FOL

2. Assemble the relevant knowledge

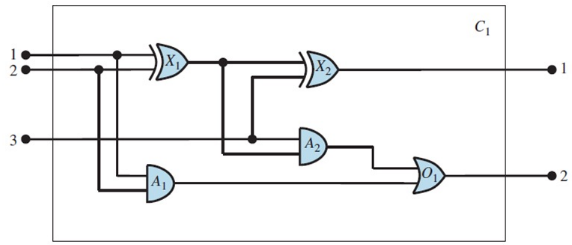
- Circuits composed of wires and gates
- Signals flow along wires to the input terminals of gates
- Each gate produces a signal on the output terminal that flows along another wire
- There are four types of gates: AND, OR, and XOR gates have two input terminals, and NOT gates have one



Knowledge Engineering in FOL

3. Decide on a vocabulary

- Choose functions, predicates, and constants to represent gates, terminals, signals, and circuits
- Each gate is represented as an object named by a constant, about which we assert that it is a gate with
  - Gate(X1)
  - The behavior of a gate is determined by its type: constants AND, OR, XOR, NOT
  - A gate has one type; we use a function Type(X1)=XOR
- Circuit(C1)
- Terminal(x)
- A circuit has n>=1 input terminals and m>=1 output terminals
  - In(1, X1) the first input terminal of X1
  - Out(n, c) is for output terminals
- The predicate Arity(c, i, j) means circuit c has i input and j output terminals
- Connected is a predicate for the connectivity between gates
  - Connected(Out(1,X1), In(1,X2))
- Signal(t) is a function denoting the signal value for the terminal t
  - We also introduce two objects for the signal value 0 (off) and 1 (on)

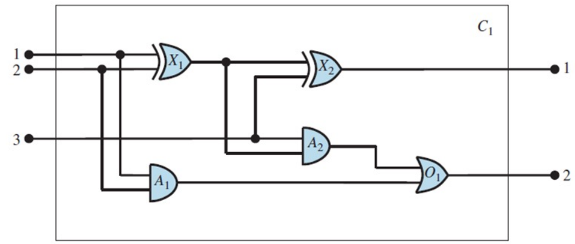


Knowledge Engineering in FOL

4. Encode general knowledge of the domain

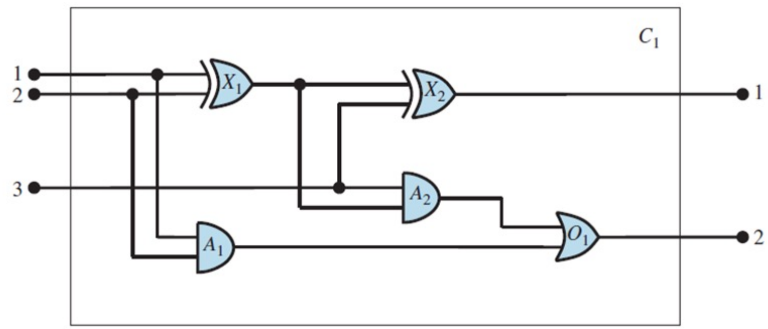
- One sign for a good ontology is that we require only a few general rules clearly and concisely stated

- Example:
  - *If two terminals are connected, then they have the same signal*:
    - $\forall\, t_1, t_2\ Terminal(t_1) \land Terminal(t_2) \land Connected(t_1, t_2) \Rightarrow Signal(t_1)=Signal(t_2)$
  - *The signal at every terminal is either 1 or 0*
    - $\forall\, t\ \text{Terminal}(t) \Rightarrow \text{Signal}(t) = 1 \lor \text{Signal}(t) = 0$
  - *Connected is commutative*
    - $\forall\, t_1, t_2\ \text{Connected}(t_1, t_2) \Leftrightarrow \text{Connected}(t_2, t_1)$
  - *There are four types of gates*
    - $\forall\, g\ \text{Gate}(g) \land k=\text{Type}(g) \Rightarrow k = \text{AND} \lor k = \text{OR} \lor k = \text{XOR} \lor k = \text{NOT}$
  - *A NOT gate's output is different from its input*
    - $\forall\, g\ \text{Gate}(g) \land \text{Type}(g) = \text{NOT} \Rightarrow \text{Signal}(\text{Out}(1,g)) \neq \text{Signal}(\text{In}(1,g))$
  - *An XOR gate's output is 1 iff its inputs are different*
    - $\forall\, g\ \text{Gate}(g) \land \text{Type}(g) = \text{XOR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \text{Signal}(\text{In}(1,g)) \neq \text{Signal}(\text{In}(2,g))$



Knowledge Engineering in FOL

5. Encode the specific problem instance
   - Categorize the circuit and its component gates
   - Circuit(C_1) ∧ Arity(C_1, 3, 2)
   - …
   - Show the connections:
     - Connected(Out(1,X1), In(1,X2))
     - …
     - Connected(In(1,C1); In(1,X1))
     - …



Knowledge Engineering in FOL

6. Pose queries to the inference procedure

- What combinations of inputs would cause the first output of C1 (the sum bit) to be 0 and the second output (the carry bit) to be 1?

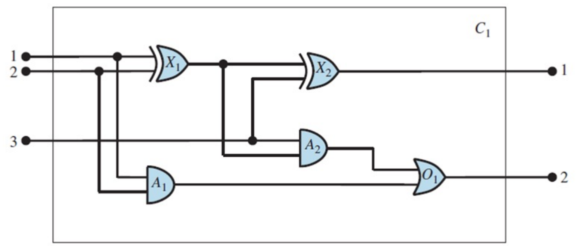
$$\exists i_1, i_2, i_3\ Signal(In(1, C_1)) = i_1 \wedge Signal(In(2, C_1)) = i_2 \wedge Signal(In(3, C_1)) = i_3 \wedge Signal(Out(1, C_1)) = 0 \wedge Signal(Out(2, C_1)) = 1$$

- ASK will return the substitutions that give the sentence entailed by the KB
  - $\{i_1/1, i_2/1, i_3/0\}, \{i_1/1, I_2/0, i_3/1\}, \{i_1/0, i_2/1, i_3/1\}$

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

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

- This final query will return a complete input–output table for the device to check that it adds its input correctly



Knowledge Engineering in FOL

7. Debug the knowledge base

- We can perturb the knowledge base in various ways to see what kinds of erroneous behaviors emerge
- Example if no assertion $1 \neq 0$
- Suppose that the system is unable to prove any outputs for the circuits, except for the input cases 000 and 110
- We can try to identify the problem by asking for the output of each gate, for instance
  - $\exists i_1, i_2, o\ Signal(In(1, C_1)) = i_1 \wedge Signal(In(2, C_1)) = i_2 \wedge Signal(Out(1, X_1)) = o$
  - It reveals that no outputs are known at X1 for the input cases 10 and 01
  - Then, looking at axiom for XOR gates, as applied to X1:
  - $Signal(Out(1, X_1)) = 1 \Leftrightarrow Signal(In(1, X_1)) \neq Signal(In(2, X_1))$
    - If the inputs are known to be 1 and 0, for instance, then this reduces to
    - $Signal(Out(1, X_1)) = 1 \Leftrightarrow 1 \neq 0,$
      - the system is unable to infer that $Signal(Out(1, X_1)) = 1$

        We need to tell it that $1 \neq 0$