INF5390 – Kunstig intelligens

Knowledge Engineering in FOL

Roar Fjellheim

Outline

- Knowledge engineering
- Electronics circuits domain
- Encoding knowledge
- Posing queries
- Summary

Chapter 8: First-Order Logic

Knowledge engineering

- Knowledge engineering is the process of building a knowledge base (KB) for a domain
- Carried out by *knowledge engineers* (KE) doing *knowledge acquisition*, often by interviewing domain experts
- The KE investigates the domain, learns important concepts, and creates a formal representation of domain objects and relations
- Most KBs are *special purpose*, covering a specific domain in detail. Other KBs are *general purpose,* valid across many domains

Knowledge engineering vs. programming

Knowledge engineering

- Choosing a logic for knowledge representation
- Building a knowledge base of facts and rules
- Implementing the inference procedure
- Inferring new facts

Programming

- Choosing a programming language
- Writing a program of data and control structures
- Choosing a compiler for the language
- Running the program

Declarative approach

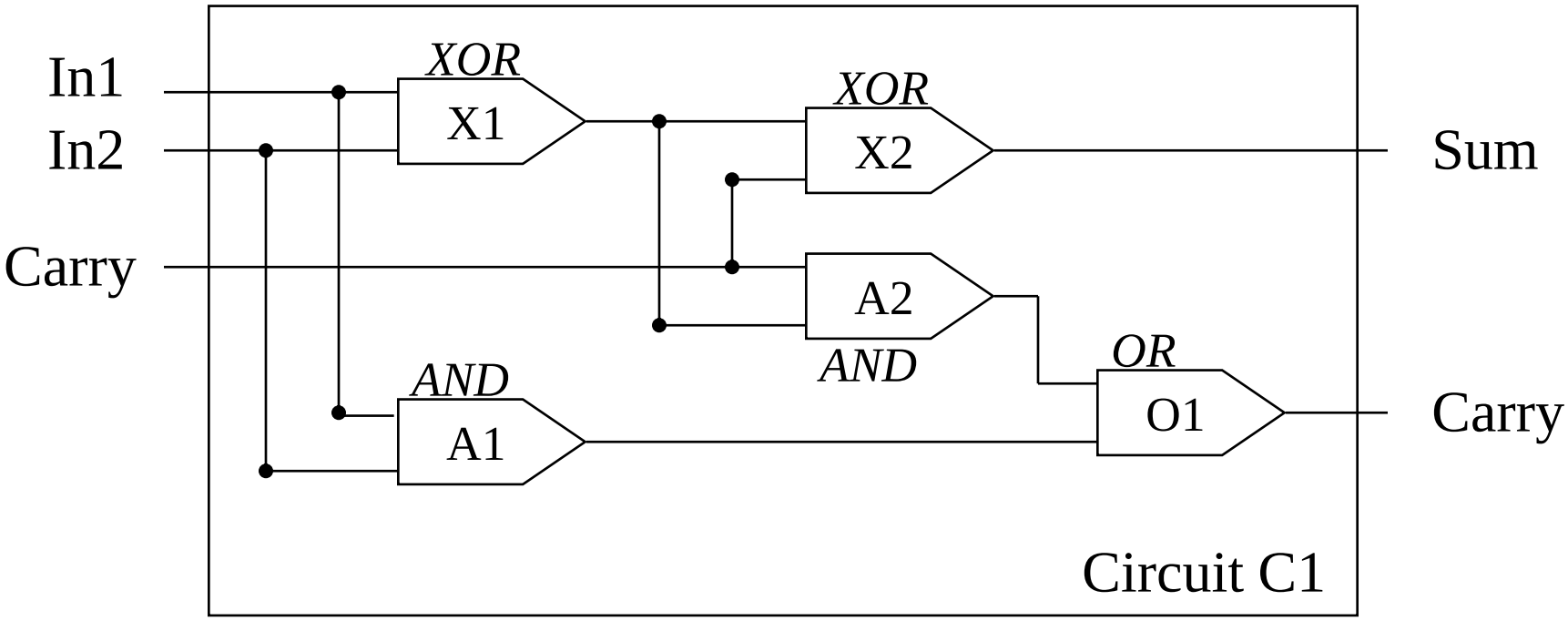
- Main point is that knowledge engineering is *declarative*
  - The knowledge engineer tells the system *what* is true
  - The system knows *how to* infer new facts and solutions
- Some advantages
  - More efficient/high-level development, less debugging
  - The knowledge base can be (re-)used for other tasks
  - The inference engine can be (re-)used for other knowledge bases

Knowledge engineering process (FOL)

1. Identify the task
2. Assemble relevant knowledge
3. Decide on a vocabulary of predicates, functions and concepts (ontology)
4. Encode general domain knowledge (axioms)
5. Encode specific problem instance
6. Pose queries to inference engine – get answers
7. Debug knowledge base

Electronic circuit domain

One-bit full adder circuit



One-bit adder desired behavior

In1	In2	Carry	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

1. Identify the task

- We are only interested in circuit topology and behavior, not physical properties (which would be needed for design)
- The purpose is to check circuit functional behavior, e.g.:
  - √ Does the circuit shown actually add properly?
- Also interested in circuit structure, e.g.
  - √ Does the circuit contain feedback loops?

2. Assemble relevant knowledge

- Digital *circuits* are composed of wires and *gates*
- A gate has input and output *terminals*
- *Signals* flow on wires to input terminals of gates
- A gate produces an output signal on the output wire
- There are four gate *types*: AND, OR, XOR, NOT

3. Decide on a vocabulary

- Constants for naming gates
  - X_1, X_2, etc.
- Function for type of gate
  - $Type(X_1) = XOR$
- Similar for circuit and terminal
  - $Circuit(C_1)$
  - $Terminal(x)$
- Terminal arity
  - $Arity(c, ins, outs)$
- Functions to select terminal
  - $Out(1, X_1)$, $In(2, X_2)$
- Predicate for connectivity between gates
  - $Connected(Out(1, X_1), In(2, X_2))$
- Function for signal at terminal
  - $Signal(terminal) = 1/0$

4. Encode general domain knowledge

- Two connected terminals have same signal

$$\forall t_1, t_2 \, Terminal(t_1) \wedge Terminal(t_2) \wedge Connected(t_1, t_2) \Rightarrow Signal(t_1) = Signal(t_2)$$

- Signal at a terminal is either on or off

$$\forall t \, Terminal(t) \Rightarrow Signal(t) = 1 \vee Signal(t) = 0$$

- Connected is commutative

$$\forall t_1, t_2 \, Connected(t_1, t_2) \Leftrightarrow Connected(t_2, t_1)$$

- OR gate behavior

$$\forall g \, Gate(g) \wedge Type(g) = OR \Rightarrow Signal(Out(1, g)) = 1 \Leftrightarrow \exists n \, Signal(In(n, g)) = 1$$

5. Encode specific problem instance

- Circuit

$$Circuit(C_1) \wedge Arity(C_1,3,2)$$

- Gates

$$Gate(X_1) \wedge Type(X_1) = XOR, Gate(X_2) \wedge Type(X_2) = XOR$$

$$Gate(A_1) \wedge Type(A_1) = AND, Gate(A_2) \wedge Type(A_2) = AND$$

$$Gate(O_1) \wedge Type(O_1) = OR$$

- Connections

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

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

$$Etc.$$

6. Pose queries – get answers

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

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

- The answer

$$(i_1 = 1 \wedge i_2 = 1 \wedge i_3 = 0) \vee$$
$$(i_1 = 1 \wedge i_2 = 0 \wedge i_3 = 1) \vee$$
$$(i_1 = 0 \wedge i_2 = 1 \wedge i_3 = 1)$$

6. Pose query – get answer (cont.)

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

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

- The answer is the full table of circuit behavior
- Compare with specified behavior (slide 8) to **verify** that circuit behaves as desired

Summary

- *Knowledge engineering* is a multi-stage process, incl. analyzing the domain, selecting a vocabulary and knowledge encoding, instantiating the encoding, and posing queries for solving problems
- The *declarative,* knowledge-based approach to building systems has advantages over programming: stating *what* is true instead of worrying about *how* problems are solved
- First-order logic (FOL) is well suited as a formal language supporting knowledge engineering tasks