First Order Logic (FOL)

CE417: Introduction to Artificial Intelligence
Sharif University of Technology
Spring 2016

Soleymani

"Artificial Intelligence: A Modern Approach", 3rd Edition, Chapter 8

Why FOL?

- To represent knowledge of complex environments concisely.
  - Expressive enough to represent a good deal of of our knowledge
    - E.g., "pits cause breezes in adjacent squares" needs many rules in propositional logic but one rule in FOL

Natural language elements & FOL elements

- Some basic elements of natural language (included also in FOL):
  - Nouns and noun phrases referring to objects (squares, pits, wumpus)
    - Some of objects are defined as functions of other objects
  - Verbs and verb phrases referring to relation among objects (is breezy, is adjacent to, shoot)

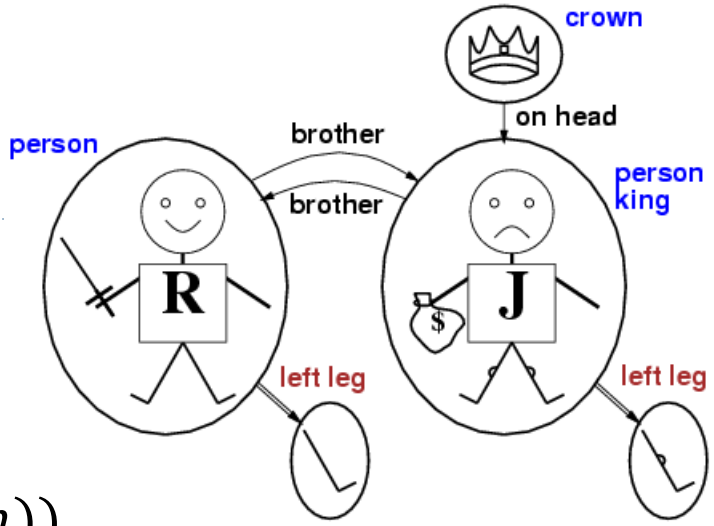
- Examples:
  - Objects: people, houses, numbers, baseball games, …
  - Relations
    - Unary relation or property: red, round, prime, …
    - *n*-ary: brother of, bigger than, inside, part of, owns, comes between, …
  - Functions: father of, best friend, one more than, beginning of, …

- FOL does not include all elements of natural language.

Symbols & interpretations

- Types of symbols
  - Constant for objects
  - Predicate for relations
  - Function for functional relations

Example



- $Brother(Richard, John)$
- $Father(John) = Henry$
- $Married(Father(Richard), Mother(John))$
- $\forall x \;\; King(x) \Rightarrow Person(x)$
- $\exists x \;\; Crown(x) \Rightarrow OnHead(x, John)$
- $\exists x, y \;\; Brother(x, Richard) \;\wedge\; Brother(y, Richard) \;\wedge\; \neg(x = y)$
- $\neg King(Richard) \Rightarrow King(John)$
- $\forall x, y \; Sibling(x, y) \Leftrightarrow [\neg(x = y) \wedge \exists m, f \; \neg(m = f) \wedge Parent(m, x) \wedge Parent(f, x) \wedge Parent(m, y) \wedge Parent(f, y)]$

Syntax of FOL: Basic elements

- Logical elements
  - Connectives $\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$
  - Quantifiers $\forall, \exists$
- Domain specific elements
  - Constants $John, Richard, \ldots$
  - Predicates $Brother, \ldots$
  - Functions $LeftLegOf, \ldots$
- Non-logical general elements
  - Variables $x, y, a, b, \ldots$
  - Equality $=$

Syntax of FOL: Atomic sentences

Atomic Sentence → *Predicate* | *Predicate* (*Term*,..., *Term*) | *Term*=*Term*

Term → *Constant*
| *variable*
| *Function*(*Term*,...)

Relation

Object

Constant → *Richard* | *A* | X_1 | ...

Variable → *a* | *x* | *s* | ...

Function → *Mother* | *LeftLeg* | ...

Predicate → *True* | *False* | *After* | *Loves* | *Hascolor* | ....

1) $Brother(Richard, John)$

2) $Father(John) = Henry$

3) $Married(Father(Richard), Mother(John))$

Syntax of FOL (BNF Grammar)

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{Atomic Sentence} \mid \textit{ComplexSentence} \\
\textit{Atomic Sentence} &\rightarrow \textit{Predicate} \mid \textit{Predicate}(\textit{Term},\ldots,\textit{Term}) \mid \textit{Term}=\textit{Term} \\
\textit{ComplexSentence} &\rightarrow (\textit{Sentence}) \mid \neg \textit{Sentence} \\
&\mid \textit{Sentence} \wedge \textit{Sentence} \mid \textit{Sentence} \vee \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
&\mid \textit{Quantifier variable, } \ldots \textit{Sentence}
\end{aligned}
$$

$$
\begin{aligned}
\textit{Term} &\rightarrow \textit{Function}(\textit{Term},\ldots) \\
&\mid \textit{Constant} \\
&\mid \textit{variable}
\end{aligned}
$$

$$
\begin{aligned}
\textit{Quantifier} &\rightarrow \forall \mid \exists \\
\textit{Constant} &\rightarrow A \mid X_1 \ldots \\
\textit{Variable} &\rightarrow a \mid x \mid s \mid \ldots \\
\textit{Predicate} &\rightarrow \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Hascolor} \mid \ldots. \\
\textit{Function} &\rightarrow \textit{Mother} \mid \textit{LeftLeg} \mid \ldots
\end{aligned}
$$

$\textit{Operator Precedence}: \neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow$

Universal quantification

- $\forall x\, P(x)$ is true in a model m iff $P(x)$ is true with x being each possible object in the model
  - conjunction of instantiations of $P(x)$

Existential quantification

- $\exists x\, P(x)$ is true in a model m iff $P(x)$ is true with x being some possible object in the model
  - disjunction of instantiations of $P(x)$

Properties of quantifiers

- $\forall x\ \forall y\ P$ is the same as $\forall y\ \forall x\ P$
- $\exists x\ \exists y\ P$ is the same as $\exists y\ \exists x\ P$
- $\exists x\ \forall y\ P$ is <u>not</u> the same as $\forall y\ \exists x\ P$
  - $\exists x\ \forall y\ Mother(x, y)$
    - "There is a person who is mother of everyone in the world"
  - $\forall y\ \exists x\ Mother(x, y)$
    - "Everyone in the world has a mother"

- Quantifier duality
  - $\forall x\ P \Leftrightarrow \neg\exists x \neg P$
  - $\exists x\ P \Leftrightarrow \neg\forall x\ \neg P$

FOL: Kinship domain example

- Objects
- Functions
- Predicates

FOL: Kinship domain example

- Objects: people
- Functions: $Mother, Father$
- Predicates:
  - Unary: $Male, Female$
  - Binary: $Parent, Sibling, Brother, Sister, Child, Daughter, Son, Spouse, Wife, Husband, Grandparent, Grandchild, Cousin, Aunt, Uncle$

FOL: Kinship domain example

- Sample sentences:
  - $\forall m, c \;\; Mother(c) = m \;\; \Leftrightarrow \;\; (Female(m) \;\wedge\; Parent(m, c))$
  - $\forall p, c \;\; Parent(p, c) \;\Leftrightarrow\; Child(c, p)$
  - $\forall g, c \;\; Grandparent(g, c) \;\; \Leftrightarrow \;\; \exists p \; Parent(g, p) \;\wedge\; Parent(p, c)$
  - $\forall x, y \;\; Sibiling(x, y) \;\Leftrightarrow\; x \neq y \;\wedge\; \exists p \; Parent(p, x) \;\wedge\; Parent(p, x)$

Assertions & queries in FOL KBs

- **Assertions**: sentences added to KB using $TELL$
  - $Tell(KB,\ King(John))$
  - $Tell(KB, (\forall x\ King(x) \Rightarrow Person(x)))$
- **Queries** or **goals**: questions asked from KB using ASK
  - $ASK(KB,\ King(John))$
  - $ASK(KB,\ \exists x\ King(x))$
- **Substitution** or **binding list**: AskVars
  - In KB of Horn clauses, every way of making the query true will bind the variables to specific values
    - AskVars$(KB, King(x))$: answer $\{x/John\}$
    - If KB contains $King(John) \vee King(Richard)$, there is no binding although $ASK(KB,\ \exists x\ King(x))$ is true

FOL: Set domain example

- Objects: sets, elements
- Functions: $s_1 \cap s_2, s_1 \cup s_2, \{x|s\}$
- Predicates:
  - Unary: Set
  - Binary: $x \in s, s_1 \subseteq s_2$

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

FOL: Wumpus world example

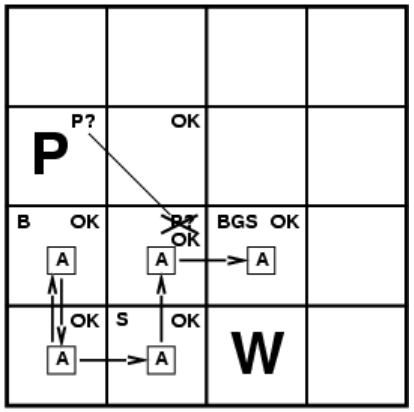
- Environment
  - Objects: pairs identifying squares $[i, j], Agent, Wumpus$
  - Relations: $Pit, Adjacent, Breezy, Stenchy,$
    $Percept, Action, At, HaveArrow, \ldots$

FOL: Wumpus world example

- Perceptions: perceives a smell, a breeze, and glitter at $t = 5$:
  - $Percept([Stench, Breeze, Glitter, None, None], 5)$
- Actions: $Turn(Left), Turn(Right), Forward, Shoot, Grab, Climb$

- Perceptions implies facts about current state
  - $\forall t, s, g, m, c \;\; Percept([s, Breeze, g, m, c], t) \Rightarrow Breeze(t)$
- Simple reflex behavior
  - $\forall t \;\; Glitter(t) \Rightarrow BestAction(Grab, t)$

- AskVars($\exists a \; BestAction(a, 5)$)
  - Binding list: e.g., $\{a/Grab\}$



FOL: Wumpus world example

- Samples of rules:
  - $\forall x, y, a, b \;\; Adj([x, y], [a, b]) \Leftrightarrow \big(x = a \;\wedge (y = b - 1 \;\vee y = b - 1)\big) \vee \big(y = b \;\wedge (x = a - 1 \;\vee x = a + 1)\big)$
  - $At(Agent, s, t)$
  - $\exists x, y \;\; \forall t \;\; At(Wumpus, [x, y], t)$
  - $\forall s, t \;\; At(Agent, s, t) \wedge Breeze(t) \;\Rightarrow\; Breezy(s)$
  - $\forall x, s_1, s_2, t \;\; At(Agent, s_1, t) \wedge At(Agent, s_2, t) \;\Rightarrow\; s_1 = s_2$
  - $\forall s, t \;\; Breezy(s) \;\Rightarrow\; \exists r \; Adj(r, s) \;\wedge Pit(r)$

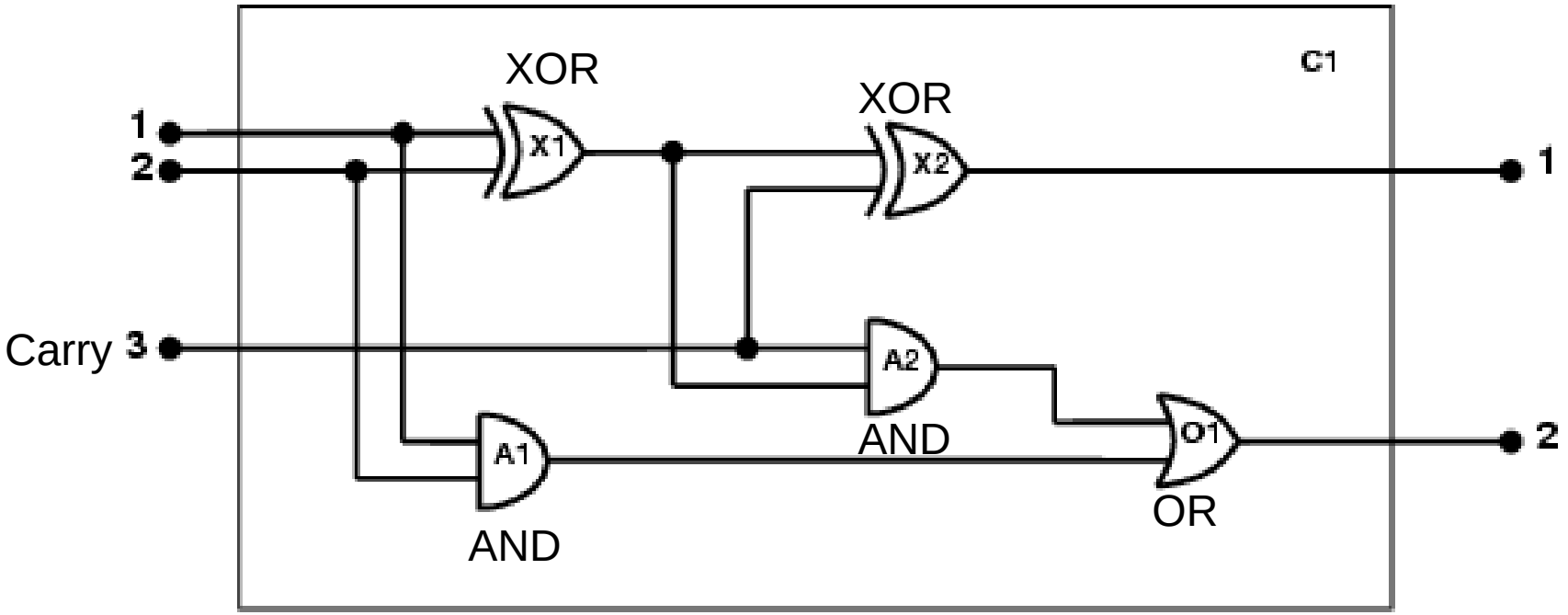
- One successor-state axiom for each predicate
  - $\forall t \;\; HaveArrow(t + 1) \;\Leftrightarrow (\; HaveArrow(t) \;\wedge\; \neg Action(Shoot, t))$

Knowledge Engineering (KE) in FOL

1) Identify the task
2) Assemble the relevant knowledge
3) Decide on a vocabulary of predicates, functions, and constants (Ontology)
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

KE: electronic circuits example

One-bit full adder

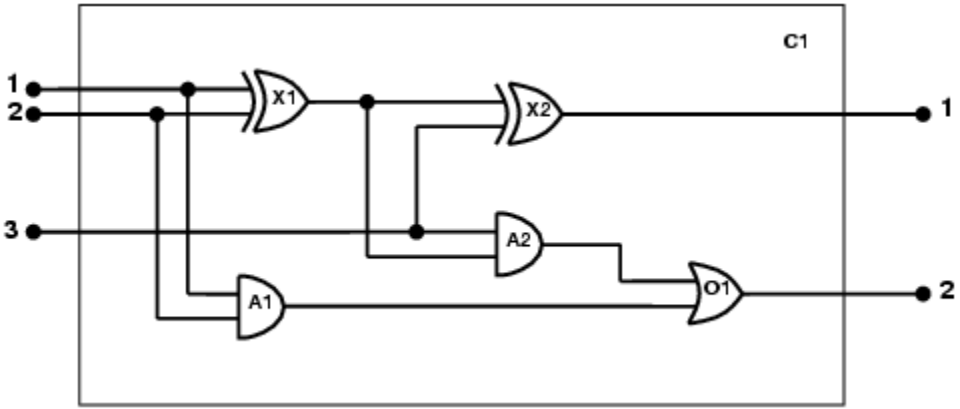


KE: electronic circuits example (steps)

1) Identify the task
   - Does the circuit add properly? (circuit verification)

2) Assemble the relevant knowledge
   - Signals, input and output terminals, gates, wires connecting gates, types of gates (AND, OR, XOR, NOT)

3) Decide on a vocabulary (ontology)
   - types of gates, terminals of gates, connections between gates, signal on terminals of gates
     - e.g., alternatives for determining the type of a gate:
       - $Type(X_1) = XOR$
       - $XOR(X_1)$
       - $Type(X_1, XOR)$



KE: electronic circuits example (steps)

4) Encode general knowledge of the domain

- $\forall t_1, t_2 \; Connected(t_1, t_2) \Rightarrow Signal(t_1) = Signal(t_2)$
- $\forall t \; Signal(t) = 1 \vee Signal(t) = 0$
- $\forall t_1, t_2 \; Connected(t_1, t_2) \Leftrightarrow Connected(t_2, t_1)$
- $\forall g \; Type(g) = OR \Rightarrow (Signal(Out(1, g)) = 1 \Leftrightarrow \exists n \; Signal(In(n, g)) = 1)$
- $\forall g \; Type(g) = AND \Rightarrow (Signal(Out(1, g)) = 0 \Leftrightarrow \exists n \; Signal(In(n, g)) = 0)$
- $\forall g \; Type(g) = XOR \Rightarrow (Signal(Out(1, g)) = 0 \Leftrightarrow Signal(In(1, g)) = Signal(In(2, g)))$
- $\forall g \; Type(g) = NOT \Rightarrow (Signal(Out(1, g)) \neq Signal(In(1, g)))$
- $1 \neq 0$

[You can see a more accurate encoding in 3rd edition of AIMA]

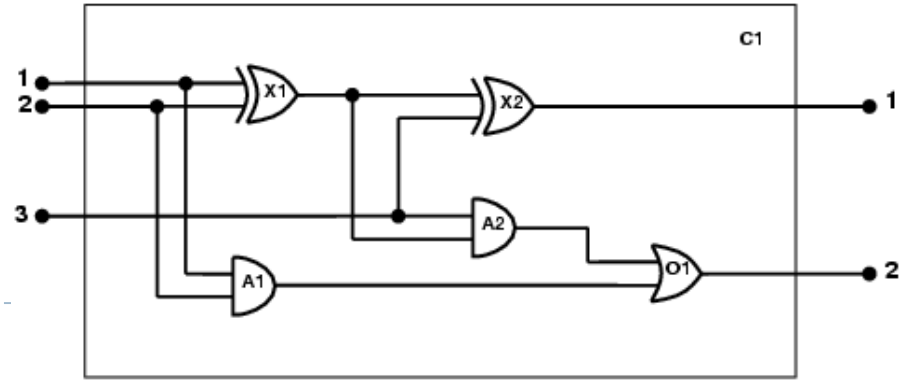
KE: electronic circuits example (steps)

5) Encode the specific problem instance

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

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

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

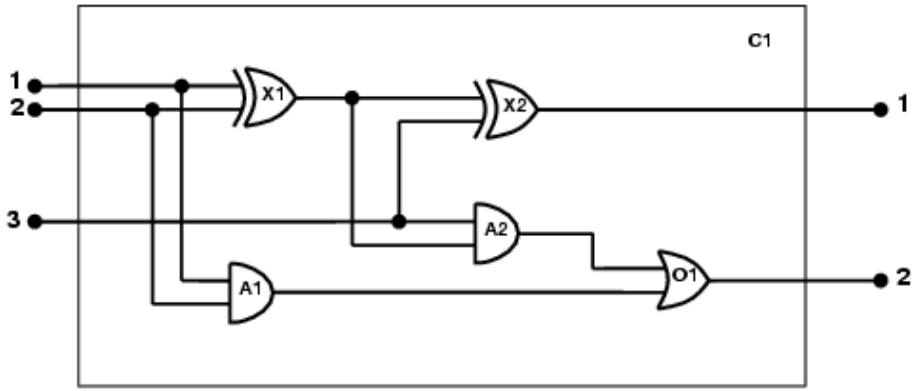


KE: electronic circuits example (steps)

6) Pose queries to the inference procedure

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

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



KE: electronic circuits example (steps)

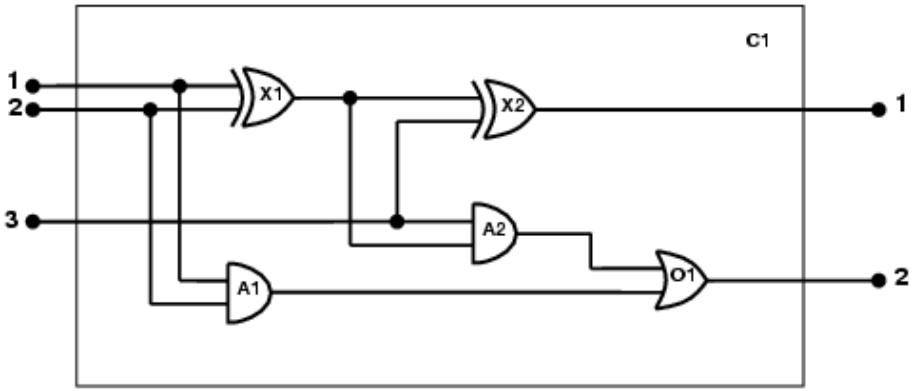
6) Pose queries to the inference procedure

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

Bindings = whole input-output table

7) Debug the knowledge base

Has $1 \neq 0$ been asserted?



Summary

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