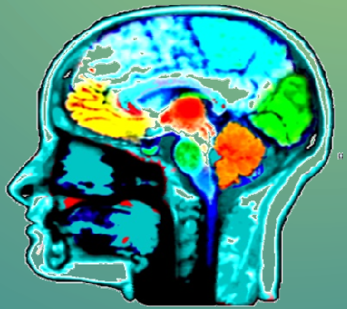
Introduction To Artificial Intelligence

Isfahan University of Technology (IUT)
1402



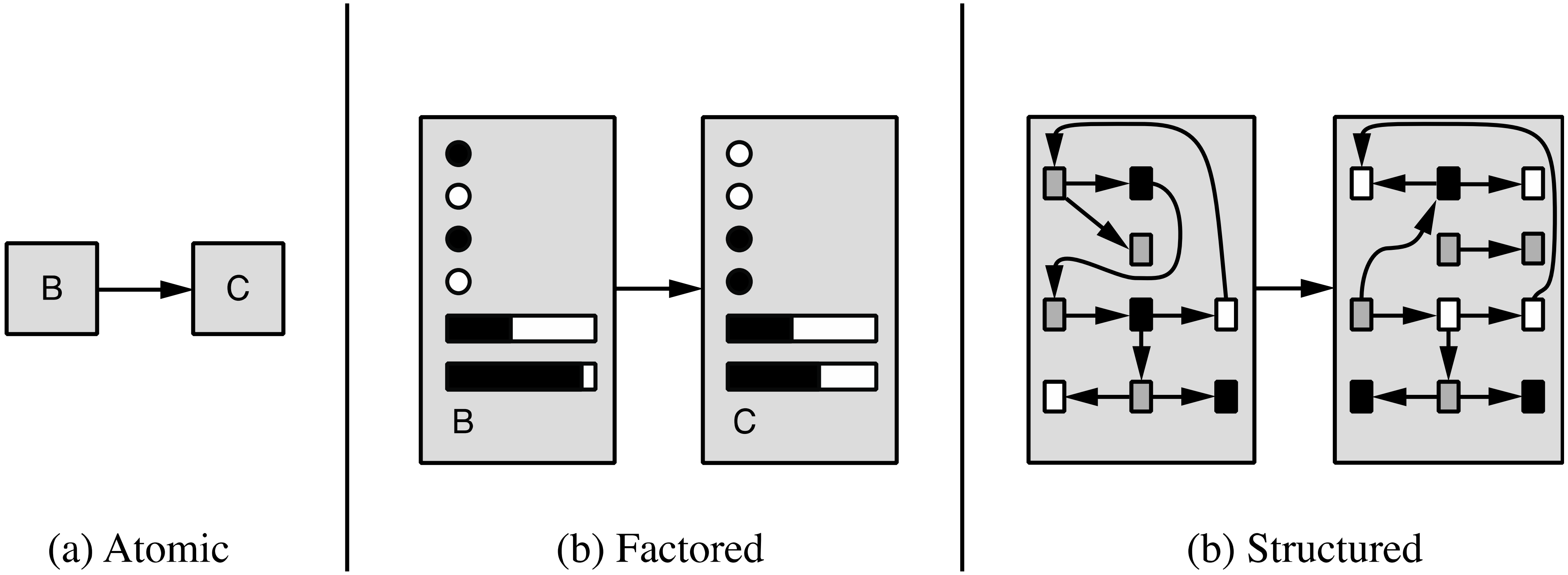
First Order Logic (FOL)

Dr. Hamidreza Hakim
hakim@iut.ac.ir

[These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley.]

بسم الله الرحمن الرحیم

Spectrum of representations



Search, game-playing

CSPs, planning, propositional logic, Bayes nets, neural nets

First-order logic, databases, logic programs, probabilistic programs

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

Expressive power

- Rules of chess:
  - 100,000 pages in propositional logic
  - 1 page in first-order logic

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)

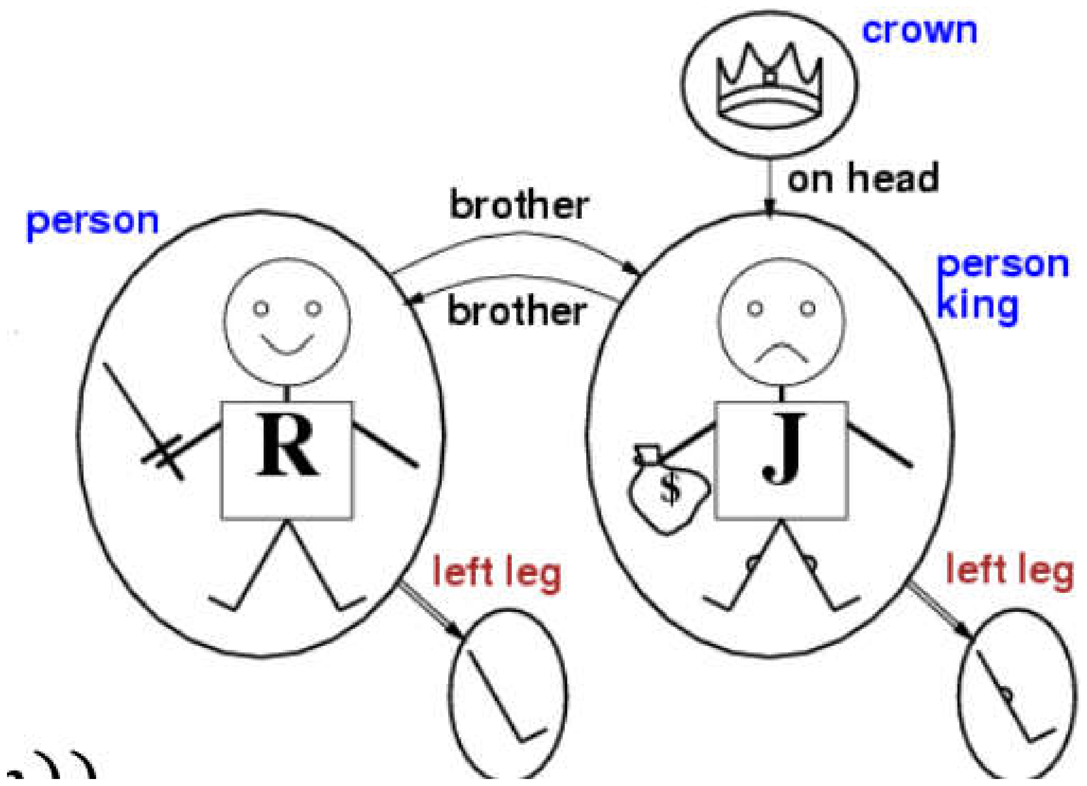
- Examples:
  - Objects: people, houses, numbers, baseball games, …
  - Relations
    - Unary relation or property: red, round, prime, …
    - *n*-ary: brother of, bigger than, inside, part of, owns, comes between, …
  - Functions: father of, best friend, one more than, beginning of, …

- 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


crown
on head
person
brother
brother
person
king
R
J
$
left leg
left leg

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 $\rightarrow$ *Predicate* | *Predicate* (*Term*,..., *Term*) | *Term*=*Term*

Relation

Term $\rightarrow$ *Constant*
| *variable*
| *Function*(*Term*,...)

Object

1) $Brother(Richard, John)$

2) $Father(John) = Henry$

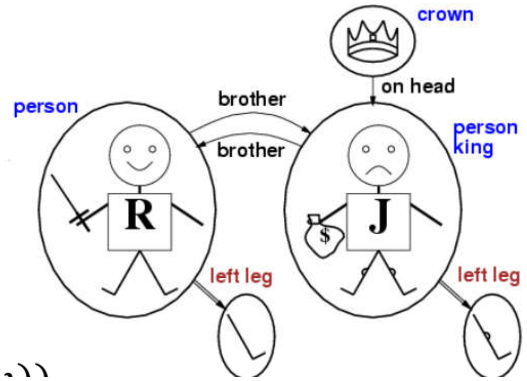
3) $Married(Father(Richard), Mother(John))$

Constant $\rightarrow$ *Richard* | *A* | X_1 | ...

Variable $\rightarrow$ *a* | *x* | *s* | ...

Function $\rightarrow$ *Mother* | *LeftLeg* | ...

Predicate $\rightarrow$ *True* | *False* | *After* | *Loves* | *Hascolor* | ....



Syntax of FOL (BNF Grammar)

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{Atomic Sentence} \mid \textit{ComplexSentence} \\
\textit{Atomic Sentence} &\rightarrow \textit{Predicate} \mid \textit{Predicate}\,(\textit{Term},\ldots,\textit{Term}) \mid \textit{Term}=\textit{Term} \\
\textit{ComplexSentence} &\rightarrow (\ \textit{Sentence}\ ) \mid \neg\ \textit{Sentence} \\
&\quad \mid \textit{Sentence} \wedge \textit{Sentence} \mid \textit{Sentence} \vee \textit{Sentence} \\
&\quad \mid \textit{Sentence} \Rightarrow \textit{Sentence} \mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
&\quad \mid \textit{Quantifier variable},\ \ldots\textit{Sentence} \\
\\
\textit{Term} &\rightarrow \textit{Function}(\textit{Term},\ldots) \\
&\quad \mid \textit{Constant} \\
&\quad \mid \textit{variable} \\
\\
\textit{Quantifier} &\rightarrow \forall \mid \exists \\
\textit{Constant} &\rightarrow A \mid X_1 \ldots \\
\textit{Variable} &\rightarrow a \mid x \mid s \mid \ldots \\
\textit{Predicate} &\rightarrow \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Hascolor} \mid \ldots. \\
\textit{Function} &\rightarrow \textit{Mother} \mid \textit{LeftLeg} \mid \ldots
\end{aligned}
$$

$$
\begin{aligned}
&\neg Brother(LeftLeg(Richard), John) \\
&Brother(Richard, John) \wedge Brother(John, Richard) \\
&King(Richard) \vee King(John) \\
&\neg King(Richard) \Rightarrow King(John).
\end{aligned}
$$

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 $P(x)$

Everyone at IUT is smart:
$\forall$x At(x,IUT) $\Rightarrow$ Smart(x)

At(King_John,IUT) $\Rightarrow$ Smart(King_John)
$\wedge$ At(Richard,IUT) $\Rightarrow$ Smart(Richard)
$\wedge$ At(IUT,IUT) $\Rightarrow$ Smart(IUT)

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 $P(x)$

Someone at IUT is smart:

$\exists$x At(x,IUT) $\wedge$ Smart(x)

At(King_John,IUT) $\wedge$ Smart(King_John)
$\vee$ At(Richard,IUT) $\wedge$ Smart(Richard)
$\vee$ At(IUT,IUT) $\wedge$ Smart(IUT)

A Common Mistake to avoid

$\forall x\ At(x,IUT) \Rightarrow Smart(x)$

$\exists x\ At(x,IUT) \land Smart(x)$

اشتباه : استفاده از سور عمومي با تركيب عطفي

معناي " هر كسي در كلاس است و هر كسي باهوش است " مي باشد

$\forall x\ At(x,IUT) \land Smart(x)$

اشتباه : استفاده از سور وجودي با تركيب شرطي

$\exists x\ At(x,IUT) \Rightarrow Smart(x)$

درست است اگر شخصي وجود داشته باشد كه در IUT نباشد!!!

Properties of quantifiers(Nested 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"

Properties of quantifiers(Connections between ∀ and ∃)

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

Demorgan Why?

$\neg(\forall x P) \equiv \exists x \neg P$

$\neg(\exists x P) \equiv \forall x \neg P$

$\forall x P \equiv \neg\exists x \neg P$

$\exists x P \equiv \neg\forall x \neg P$

$\neg(p \wedge q)$	$\neg p \vee \neg q$
$\neg(p \vee q)$	$\neg p \wedge \neg q$
$(p \wedge q)$	$\neg(\neg p \vee \neg q)$
$(p \vee q)$	$\neg(\neg p \wedge \neg q)$

Example

∀x Likes(x,IceCream)	¬∃x ¬Likes(x,IceCream)
∃x Likes(x,Broccoli)	¬∀x ¬Likes(x,Broccoli)

Equality

- New form to make atomic sentences.
- use the equality symbol to signify that two terms refer to the same object.

$$Father(John) = Henry$$

$\exists x, y \; Brother(x, Richard) \land Brother(y, Richard) \land \neg(x = y)$. (Richard has at least two brothers)

$$\exists x, y \; Brother(x, Richard) \land Brother(y, Richard)$$

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

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
  - $ASK\big(KB,\ King(John)\big)$
  - $ASK\big(KB,\ \exists x\ \ King(x)\big)$

Assertions & queries in FOL KBs

- **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⸮
- Functions ⸮
- Predicates⸮

FOL: Set domain example

- Objects: sets, elements
- Functions: $s_1 \cap s_2, s_1 \cup s_2, \{x|s\}$

$\{x|s_2\}=Add(x,s2)$

- Predicates:
  - Unary: Set
  - Binary: $x \in s, s_1 \subseteq s_2$

1. $\forall s\ Set(s) \Leftrightarrow (s = \{\}\ ) \vee (\exists x,s_2\ Set(s_2) \wedge s = \{x|s_2\})$
2. $\neg\exists x,s\ \{x|s\} = \{\}$ (The empty set has no elements added into it)
3. $\forall x,s\ x \in s \Leftrightarrow s = \{x|s\}$ (Adding an element already in the set has no effect)
4. $\forall x,s\ x \in s \Leftrightarrow [\ \exists y,s_2\ (s = \{y|s_2\} \wedge (x = y \vee x \in s_2))]$
5. $\forall s_1,s_2\ s_1 \subseteq s_2 \Leftrightarrow (\forall x\ x \in s_1 \Rightarrow x \in s_2)$
6. $\forall s_1,s_2\ (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$
7. $\forall x,s_1,s_2\ x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$
8. $\forall x,s_1,s_2\ x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$

FOL: Wumpus world example

- the wumpus agent receives a percept vector with five elements
- the knowledge base must include both the percept and the time at which it occurred

$$Percept([Stench, Breeze, Glitter, None, None], 5)\,.$$

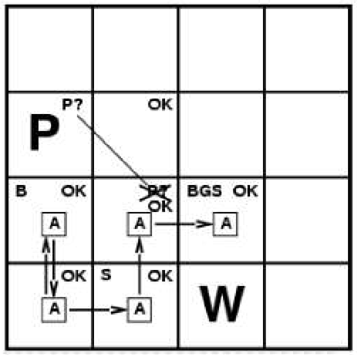
- Environment
  - Objects: pairs identifying squares $[i, j], Agent, Wumpus$
  - Relations: $Pit, Adjacent, Breezy, Stenchy,$
    $Percept, Action, At, HaveArrow, \ldots$

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

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



FOL: Wumpus world example

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

- 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)gather 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

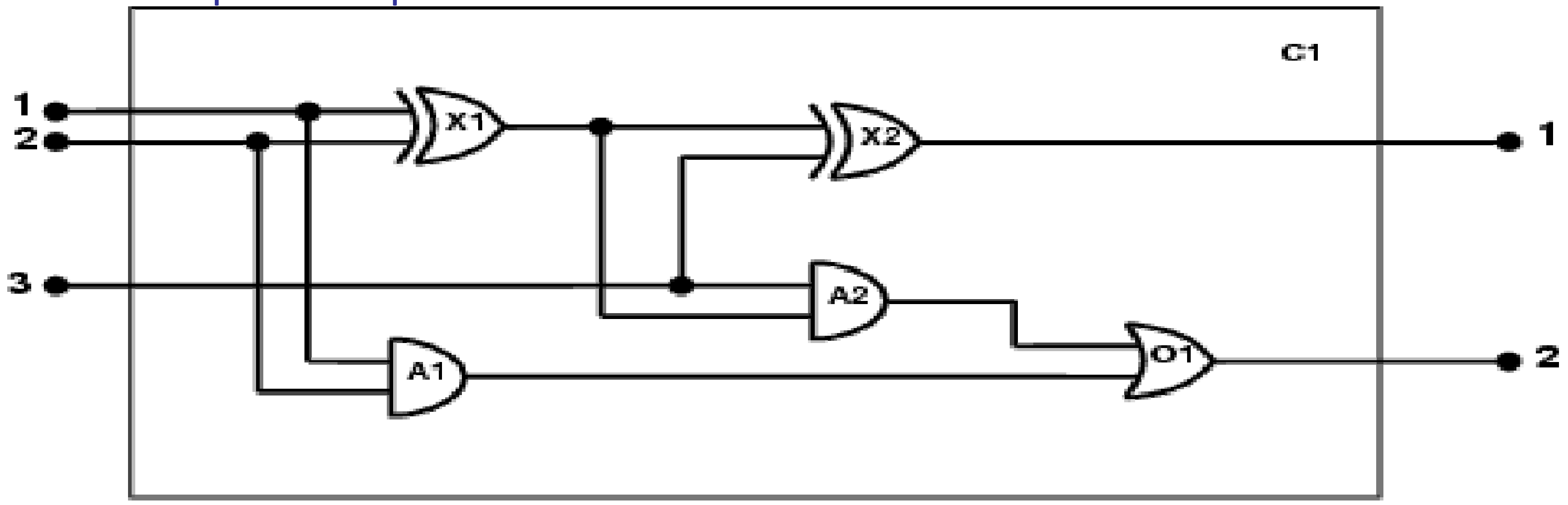
6)Pose queries to the inference procedure and get answers

7)Debug the knowledge base

KE: electronic circuits example

- One-bit full adder

Three input 2 output



KE: electronic circuits example (steps)

1)Identify the task

Does the circuit add properly? (circuit verification)

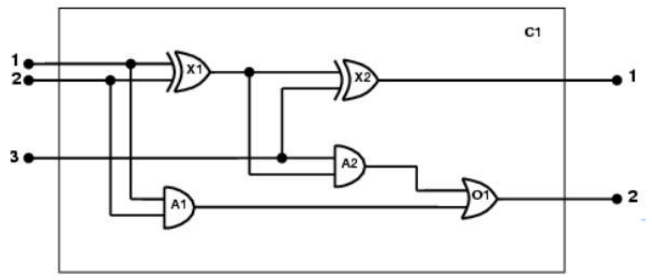
2) Assemble the relevant knowledge

Signals flow along wires to the input terminals of gates

Each gate produce a signal on the output

Signals have two state (0/1),

input and output terminals, gates, wires connecting gates, types of gates (AND, OR,XOR,NOT)



KE: electronic circuits example (steps)

3) Decide on a vocabulary (ontology)

Each gate instance represented as constant (ex X1)

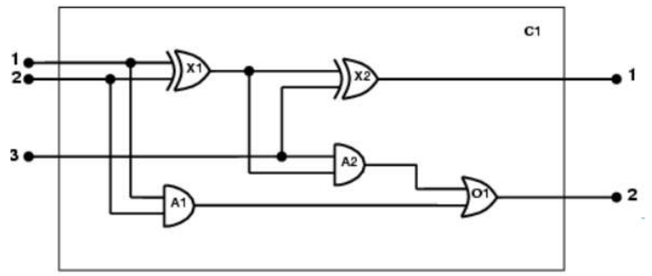
Types of gates: Each gate type represented as constant (ex “AND”)

e.g., alternatives for determining the type of a gate:

$Type(X1) = XOR$ (function type)

$XOR(X1)$ (predicate)

$Type(X1, XOR)$ (predicate)



Gate terminals represented as integer constant,(ex “t1”)

Two function In, Out (ex In(1,X1) :(first terminal of X1))

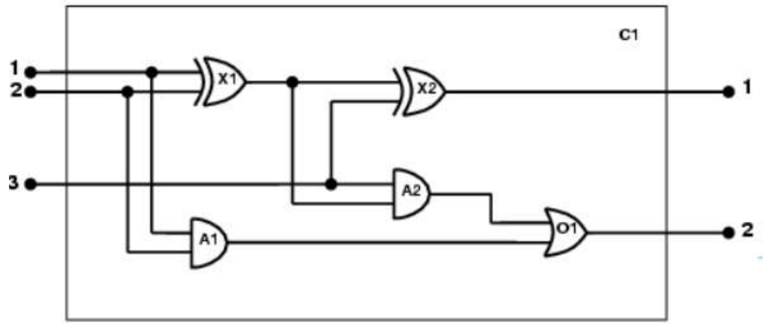
terminals of gates :Terminal represented by t

connections between gates, signal on terminals of gates by two value 0,1

….

KE: electronic circuits example (steps)

4) Encode general knowledge of the domain



- $\forall\ t_1,t_2\ \text{Connected}(t_1,t_2) \Rightarrow \text{Signal}(t_1) = \text{Signal}(t_2)$ (two connected terminals have the same signal)
- $\forall t\ \ (\text{Signal}(t) = 1)\ \ \vee\ \ (\text{Signal}(t) = 0)$ (The signal at every terminal is either 1 or 0:)
- $1 \neq 0$
- $\forall t_1,t_2\ (\text{Connected}(t_1, t_2)) \Rightarrow (\text{Connected}(t_2, t_1))$ (Connected is commutative)
- $\forall g\ \text{Type}(g) = \text{OR}\ \ \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \exists n\ \text{Signal}(\text{In}(n,g)) = 1$
- $\forall g\ \text{Type}(g) = \text{AND} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 0 \Leftrightarrow \exists n\ \text{Signal}(\text{In}(n,g)) = 0$
- $\forall g\ \text{Type}(g) = \text{XOR} \Rightarrow \text{Signal}(\text{Out}(1,g)) = 1 \Leftrightarrow \text{Signal}(\text{In}(1,g)) \neq \text{Signal}(\text{In}(2,g))$
- $\forall g\ \text{Type}(g) = \text{NOT} \Rightarrow \text{Signal}(\text{Out}(1,g)) \neq \text{Signal}(\text{In}(1,g))$

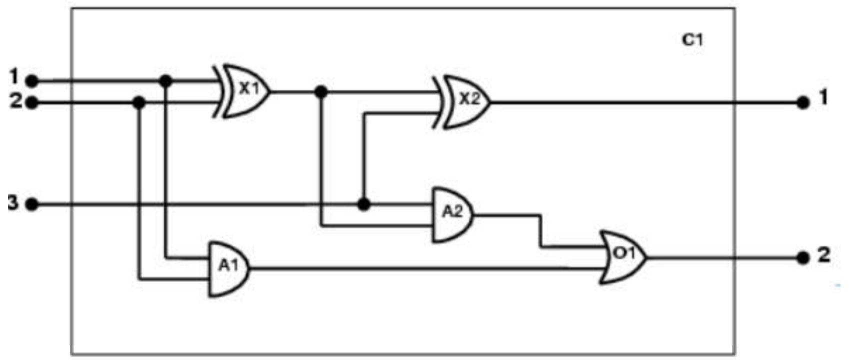
KE: electronic circuits example (steps)

5) Encode the specific problem instance

Type(X1) = XOR	Type(X2) = XOR
Type(A1) = AND	Type(A_2) = AND
Type(O_1) = OR	Circuit(C_1)

Connected(Out(1,X_1),In(1,X_2))
Connected(Out(1,X1),In(2,A_2))
Connected(Out(1,A_2),In(1,O_1))
Connected(Out(1,A_1),In(2,O_1))
Connected(Out(1,X_2),Out(1,C_1))
Connected(Out(1,O_1),Out(2,C_1))

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)

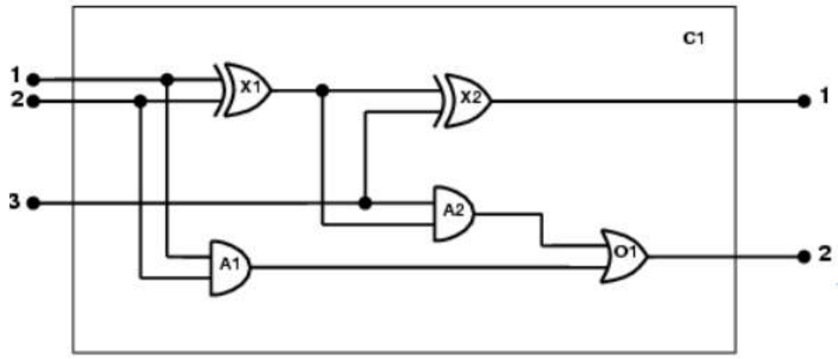
6)Pose queries to the inference procedure

- Which input 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?

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

Bindings = whole input-output table

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



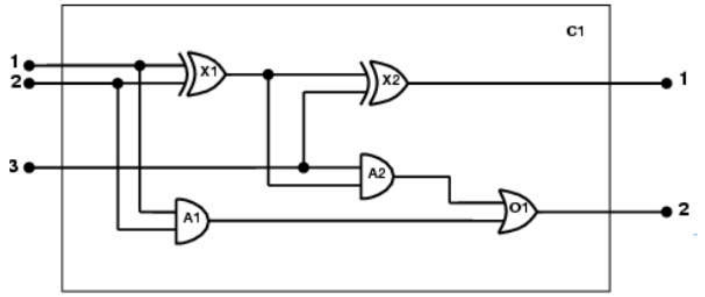
- Excercize:What are the possible value sets of all terminals?

KE: electronic circuits example (steps)

7) Debug the knowledge base

if we don’t have $1 \neq 0$ in KB ?

System unable to give output in no signals



$\exists i_1,i_2,o\ (Signal(In(1,C_1)) = i_1) \wedge (Signal(In(2,C_1)) = i_2) \wedge (Signal(Out(1,X_1)) = o)$

no outputs are known at X1 for the input cases 10 and 01.

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

$Signal(Out(1, X1)) = 1 \Leftrightarrow 1 \neq 0$

Summary, pointers

- FOL is a very expressive formal language
- It adds terms to represent objects, and has universal and existential quantifiers to construct assertions about all or some of the possible values of the quantified variables
- Many domains of common-sense and technical knowledge can be written in FOL (see AIMA Ch. 10)
  - circuits, software, planning, law, taxes, network and security protocols, product descriptions, ecommerce transactions, geographical information systems, Google Knowledge Graph, Semantic Web, etc.
- Inference technology for logic programming is especially efficient (see AIMA Ch. 9)