First-Order Logic

Chapter 8

Last chapter

Logical agents apply inference to a knowledge base
to derive new information and make decisions

Basic concepts of logic:

— **syntax（语法）: formal structure of sentences**
— **semantics（语义）: truth of sentences wrt models**
— **entailment（蕴涵）: necessary truth of one sentence given another**
— **inference（推理）: deriving sentences from other sentences**
— **soundness（可靠性）: derivations produce only entailed sentences**
— **completeness（完备性）: derivations can produce all entailed sentences**

Forward, backward chaining are linear-time, complete for Horn clauses
Resolution is complete for propositional logic

Propositional logic lacks expressive power

Outline

- Why FOL?
- Syntax and semantics of FOL
- Using FOL
- Knowledge engineering（知识工程） in FOL

Pros （优点） of propositional logic

- ☺ Propositional logic is declarative （陈述性的）
  - 知识和推理分开，而且推理完全不依赖于领域
  - 对比：程序设计语言——过程性语言
    - 缺乏从其它事实派生出事实的通用机制
    - 对数据结构的更新通过一个领域特定的过程来完成
- ☺ Propositional logic allows partial（不完全）/disjunctive（分离的）/negated information
  - (unlike most data structures and databases)
- ☺ Propositional logic is compositional（合成性的）：
  - meaning of $B_{1,1} \land P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$ （语句的含义是它的各部分含义的一个函数）
- ☺ Meaning in propositional logic is context-independent
  - (unlike natural language, where meaning depends on context)

Cons （缺点） of propositional logic

- ☹ Propositional logic has very limited expressive power
  - (unlike natural language)
  - E.g., cannot say "pits cause breezes in adjacent squares“
    except by writing one sentence for each square

    $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

Cons （缺点） of propositional logic

- All students know arithmetic.
  - AliceIsStudent → AliceKnowsArithmetic
  - BobIsStudent → BobKnowsArithmetic

  …
- Propositional logic is very clunky. What's missing?
  - Objects and relations: propositions (e.g., AliceKnowsArithmetic) have more internal structure (alice, Knows, arithmetic)
  - Quantifiers and variables: all is a quantifier which applies to each person, don't want to enumerate them all...

First-order logic

采用命题逻辑的基础—陈述式、上下文无关和合成语义，并借用自然语言的思想。

Whereas propositional logic assumes the world contains facts,

first-order logic (like natural language) assumes the world contains

- Objects（对象）: people, houses, numbers, colors, baseball games, wars, …
- Relations（关系）: red, round, prime…,

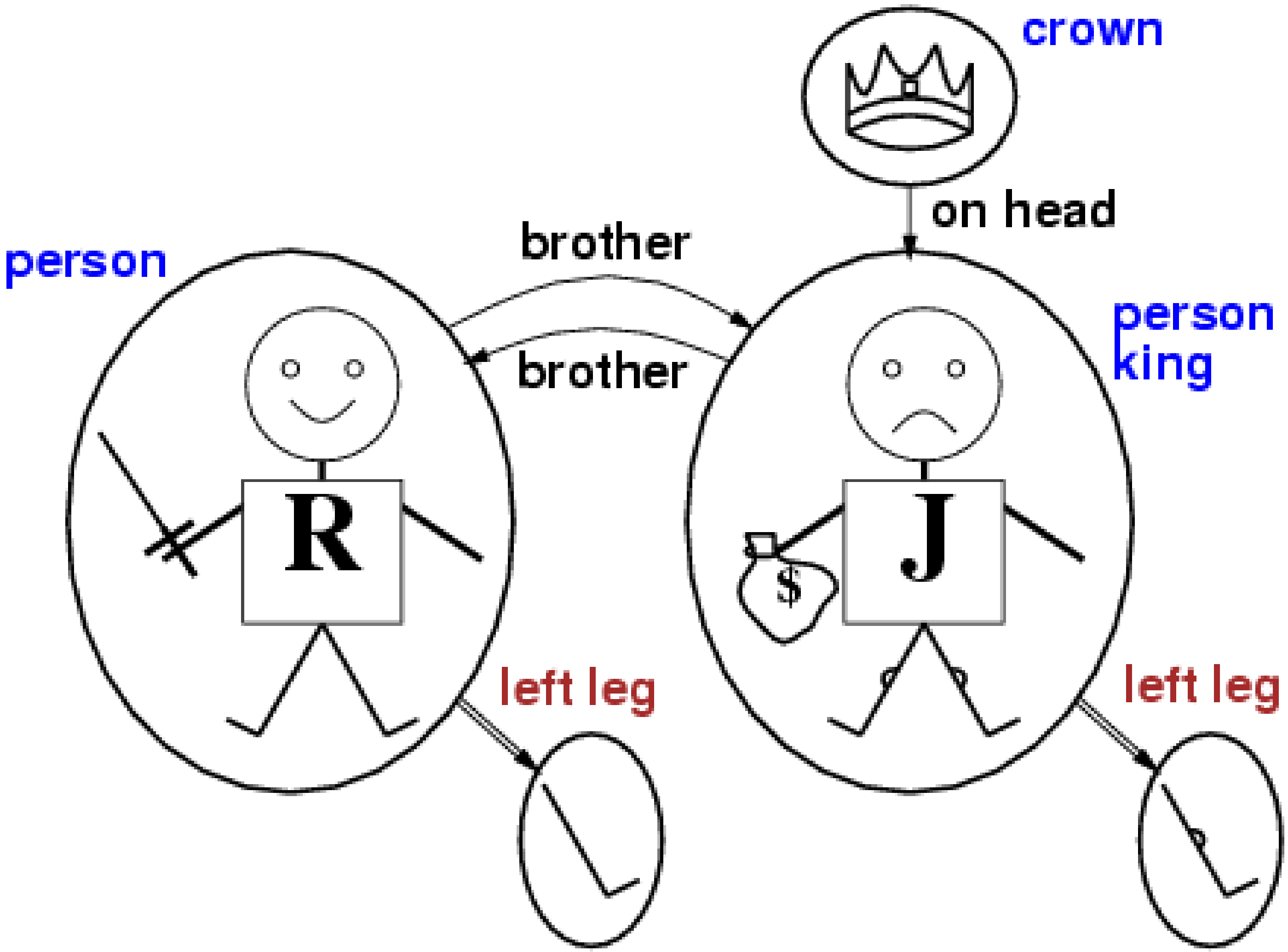
  brother of, bigger than, part of, comes between, …
- Functions（函数）: father of, best friend, one more than, plus, …

谓词用来描述个体（可以独立存在的事物）之间的关系或属性

Logics in general

语言	本体论约定（世界中存在的）	认识论约定 （智能体对事实所相信的内容）
命题逻辑 Propositional logic	事实	真/假/未知
一阶逻辑 First-order logic	事实、对象、关系	真/假/未知
时序逻辑 Temporal logic	事实、对象、关系、时间	真/假/未知
概率逻辑 Probability theory	事实	信度∈[0,1]

一阶逻辑的模型: Example



Syntax of FOL: Basic elements

- Constants/常量 KingJohn, 2, USTC,...
- Predicates/谓词 Brother, >,...
- Functions/函数 Sqrt, LeftLegOf,...
- Variables/变量 x, y, a, b,...
- Connectives/连接词 $\neg, \Rightarrow, \wedge, \vee, \Leftrightarrow$
- Equality/等词 $=$
- Quantifiers/量词 $\forall, \exists$

Atomic sentences（原子语句）

Term = *function* ($term_1$,...,$term_n$)
or *constant* or *variable*

Atomic sentence = *predicate* ($term_1$,...,$term_n$)
or $term_1 = term_2$

- E.g., *Brother(KingJohn,RichardTheLionheart)*
  > *(Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))*

Complex sentences（复合语句）

Complex sentences are made from atomic sentences using connectives

$\neg S$, $S_1 \wedge S_2$, $S_1 \vee S_2$, $S_1 \Rightarrow S_2$, $S_1 \Leftrightarrow S_2$,

E.g. *Sibling(KingJohn,Richard)* $\Rightarrow$ *Sibling(Richard,KingJohn)*

$>(1,2) \vee \leq (1,2)$

$>(1,2) \wedge \neg >(1,2)$

Truth in first-order logic

- 语句的真值由一个模型和对句子符号的解释来判定。
  Sentences are true with respect to a model and an interpretation

- Model contains objects (domain elements域元素) and relations among them

- 我们需要一个对分别被常量、谓词和函数符号指代的对象、关系和函数进行详细说明的解释
  Interpretation specifies referents（指代） 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*

Truth example

Consider the interpretation in which

Richard → Richard the Lionheart

John → the evil King John

Brother → the brotherhood relation

Under this interpretation, Brother(Richard, John) is true just in case Richard the Lionheart and the evil King John are in the brotherhood relation in the model

Models for FOL: Lots!

Entailment（蕴涵） in propositional logic（命题逻辑） can be computed by enumerating（枚举） models

We can enumerate the FOL models for a given KB vocabulary:

For each number of domain elements n from 1 to ∞

For each k-ary predicate（k元谓词） P_k in the vocabulary

For each possible k-ary relation on n objects

For each constant symbol C in the vocabulary

For each choice of referent for C from n objects …

Computing entailment by enumerating FOL models is not easy!

通过枚举所有可能模型以检验“语义后承”在一阶逻辑中不可行

Universal quantification（全称量词）

∀<*variables*> <*sentence*>
"对于所有的……"

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

$\forall x \quad P$ 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（实例的合取式） of P

$$
\begin{aligned}
& At(KingJohn,USTC) \Rightarrow Smart(KingJohn) \\
\wedge\ & At(Richard,USTC) \Rightarrow Smart(Richard) \\
\wedge\ & At(USTC,USTC) \Rightarrow Smart(USTC) \\
\wedge\ & \ldots
\end{aligned}
$$

A common mistake to avoid

Typically, $\Rightarrow$ is the main connective with $\forall$

在需要用全称量词书写一般规则的时候， $\Rightarrow$的真值表项是一个理想的选择

Common mistake: using $\wedge$ as the main connective with $\forall$:

$\forall x\ At(x,USTC) \wedge Smart(x)$

means “Everyone is at USTC and everyone is smart”

Existential quantification（存在量词）

∃<*variables*> <*sentence*>
"存在一个……，这样以致"或"对于某个……"

Someone at USTC is smart:
$\exists x$ At(x,USTC) $\wedge$ 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（实例的析取式） of *P*

At(KingJohn,USTC) $\wedge$ Smart(KingJohn)
$\vee$ At(Richard,USTC) $\wedge$ Smart(Richard)
$\vee$ At(USTC,USTC) $\wedge$ Smart(USTC)
$\vee$ ...

Another common mistake to avoid

Typically, $\wedge$ is the main connective with $\exists$

Common mistake: using $\Rightarrow$ as the main connective with $\exists$:

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

is true if there is anyone who is not at USTC!

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（量词的二义性）: each can be expressed using the other

$\forall x$ Likes(x,IceCream) $\qquad \neg\exists x\ \neg$Likes(x,IceCream)

$\exists x$ Likes(x,Broccoli) $\qquad \neg\forall x\ \neg$Likes(x,Broccoli)

Equality（等式）

$term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object（指代的对象是相同的）

E.g., definition of *Sibling* in terms of *Parent*:

$\forall x,y\ Sibling(x,y) \Leftrightarrow [\neg(x = y) \land \exists m,f\ \neg (m = f) \land Parent(m,x) \land Parent(f,x) \land Parent(m,y) \land Parent(f,y)]$

Outline

- Why FOL?
- Syntax and semantics of FOL
- Using FOL
- Knowledge engineering（知识工程） in FOL

Using FOL

The kinship（亲属关系） domain:

Brothers are siblings

$\forall$ x, y Brother(x,y) $\Rightarrow$ Sibling(x, y).

"Sibling" is symmetric

$\forall$ x, y Sibling(x, y) $\Leftrightarrow$ Sibling(y, x).

One's mother is one's female parent

$\forall$ x, y Mother(x, y) $\Leftrightarrow$ (Female(x) $\wedge$ Parent(x, y)).

A cousin is a child of a parent's sibling

$\forall$ x, y Cousin(x,y) $\Leftrightarrow$ $\exists$ p, ps Parent(p, x) $\wedge$ Sibling(ps, p) $\wedge$Parent(ps, y)

Using FOL

The set（集合） domain:

集合就是空集或通过将一些元素添加到一个集合而构成

$\forall s \quad Set(s) \Leftrightarrow (s = \{\}) \lor (\exists x, s_2 \quad Set(s_2) \land s = \{x | s_2\})$

空集没有任何元素，也就是说，空集无法再分解为更小的集合和元素

$\neg\exists x, s \quad \{x | s\} = \{\}$

将已经存在于集合中的元素添加到该集合，无任何变化

$\forall x, s \quad x \in s \Leftrightarrow s = \{x | s\}$

集合的元素仅是那些被添加到集合中的元素

$\forall x, s \quad x \in s \Leftrightarrow [\exists y, s_2 (s = \{y | s_2\} \land (x = y \lor x \in s_2))]$

Using FOL

The set（集合） domain:

一个集合是另一个集合的子集，当且仅当第一个集合的所有元素都是第二个集合的元素

$\forall s_1,s_2 \quad s_1 \subseteq s_2 \Leftrightarrow (\forall x \quad x \in s_1 \Rightarrow x \in s_2)$

两个集合是相同的，当且仅当它们互为子集

$\forall s_1,s_2 \quad (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1)$

一个对象是两个集合的交集的元素，当且仅当它同时是这两个集合的元素

$\forall x,s_1,s_2 \quad x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2)$

一个对象是两个集合的并集的元素，当且仅当它是其中某一集合的元素

$\forall x,s_1,s_2 \quad x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2)$

Interacting with FOL KBs

Suppose a wumpus-world agent is using an FOL KB and perceives a smell and a breeze (but no glitter) at *t=5*:

Tell(KB,Percept([Smell,Breeze,None],5))

Ask(KB,∃a BestAction(a,5))

I.e., does the KB entail some best action at *t=5*?

Answer: *Yes*, {*a*/*Shoot*} ← substitution (binding list绑定表)

Given a sentence *S* and a substitution σ,

*S*σ denotes the result of plugging σ into *S*; e.g.,

S = Smarter(x,y)

σ = {x/Hillary,y/Bill}

*S*σ = Smarter(Hillary,Bill)

Ask(KB,S) returns some/all σ such that KB ╞ *S*σ

Knowledge base for the wumpus world

Perception （感知）

- $\forall t,s,b$ Percept([s,b,Glitter],t) $\Rightarrow$ Glitter(t)

Reflex

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

Reflex with internal state: do we have the gold already?

$\forall$ t AtGold(t) $\wedge \neg$ Holding(Gold, t) $\Rightarrow$ BestAction(Grab, t)

Holding(Gold, t) cannot be observed $\Rightarrow$ keeping track of change is essential

Deducing hidden properties

Definition of adjacent squares

$\forall x,y,a,b \quad Adjacent([x,y],[a,b]) \Leftrightarrow [a,b] \in \{[x+1,y], [x-1,y],[x,y+1],[x,y-1]\}$

Properties of squares:

$\forall s,t \quad At(Agent,s,t) \wedge Breeze(t) \Rightarrow Breezy(s)$

Squares are breezy near a pit:

Diagnostic rule（诊断规则）— infer cause from effect

$\forall s \quad Breezy(s) \Rightarrow \exists r \quad Adjacent(r,s) \wedge Pit(r)$

Causal rule（因果规则）— infer effect from cause

$\forall r,s \quad Adjacent(r,s) \wedge Pit(r) \Rightarrow Breezy(s)$

Neither of these is complete — e.g., the causal rule doesn't say whether squares far away from pits can be breezy

Definition （定义） for the Breezy predicate:

$\forall s \quad Breezy(s) \Leftrightarrow \exists r \quad Adjacent(r,s) \wedge Pit(r)$

Outline

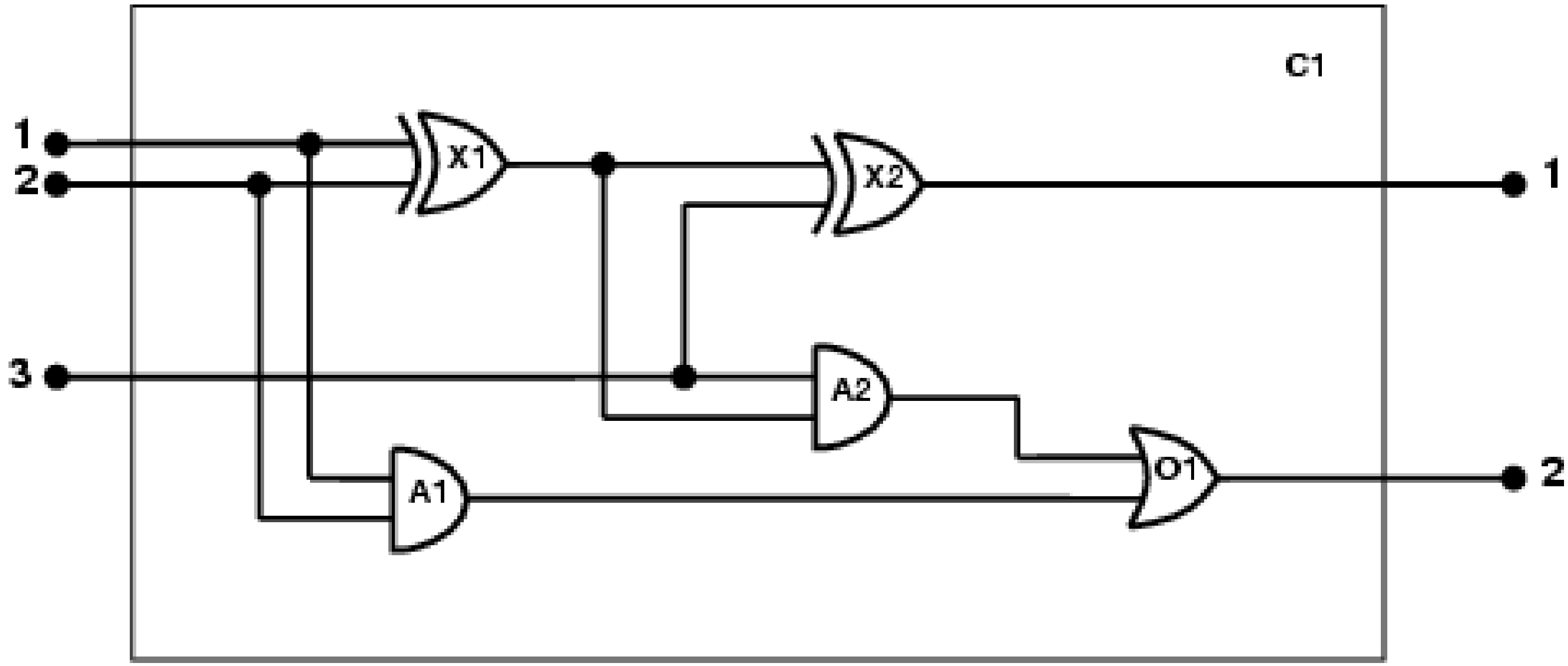
- Why FOL?
- Syntax and semantics of FOL
- Using FOL
- Knowledge engineering（知识工程） in FOL

Knowledge engineering（知识工程） in FOL

1. Identify the task
   确定任务
2. Assemble the relevant knowledge
   搜集相关知识
3. Decide on a vocabulary of predicates, functions, and constants
   确定谓词、函数和常量的词汇表
4. Encode general knowledge about the domain
   对域的通用知识进行编码
5. Encode a description of the specific problem instance
   对特定问题实例的描述进行编码
6. Pose queries to the inference procedure and get answers
   把查询提交给推理过程并获取答案
7. Debug the knowledge base
   调试知识库

The electronic circuits（电路） domain

One-bit full adder（一位全加器）



最初的两个输入是需要相加的两位，第三个输入是一个进位。
第一个输出是和，第二个输出是下一个加法器的进位。

The electronic circuits domain

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

2. Assemble the relevant knowledge
   - Composed of wires（导线） and gates（门）; Types of gates (AND, OR, XOR, NOT)
   - Irrelevant: size, shape, color, cost of gates

3. Decide on a vocabulary（词汇表）
   - Alternatives:

     $Type(X_1) = XOR$

     $Type(X_1, XOR)$

     $XOR(X_1)$

The electronic circuits domain

4. Encode（编码） general knowledge of the domain

(1) 如果两个接线端是相连的，那么它们具有相同的信号

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

(2) 每个接线端的信号不是1就是0（不可能两者都是）

$\forall t \quad Signal(t) = 1 \lor Signal(t) = 0$

$1 \neq 0$

(3) Connected是一个可交换谓词

$\forall t_1,t_2 \quad Connected(t_1, t_2) \Rightarrow Connected(t_2, t_1)$

(4) 或门的输出为1，当且仅当它的某一个输入为1

$\forall g \quad Type(g) = OR \Rightarrow$

$Signal(Out(1,g)) = 1 \Leftrightarrow \exists n \quad Signal(In(n,g)) = 1$

(5) 与门的输出为0，当且仅当它的某一个输入为0

$\forall g \quad Type(g) = AND$

$\Rightarrow Signal(Out(1,g)) = 0 \Leftrightarrow \exists n \quad Signal(In(n,g)) = 0$

(6) 异或门的输出为1，当且仅当它的输入是不相同的

$\forall g \quad Type(g) = XOR$

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

(7) 非门的输出与它的输入相反

$\forall g \quad Type(g) = NOT \Rightarrow Signal(Out(1,g)) \neq Signal(In(1,g))$

The electronic circuits domain

5. Encode the specific problem instance

首先对门加以分类

Type(X_1) = XOR Type(X_2) = XOR

Type(A_1) = AND Type(A_2) = AND

Type(O_1) = OR

其次说明门与门之间的连接

Connected(Out(1,X_1),In(1,X_2))

Connected(Out(1,X_1),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))

The electronic circuits domain

6. Pose queries to the inference procedure—把查询提交给推理过程

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

对于1位全加器有哪些可能的输入与输出组合?

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

7. Debug the knowledge base

May have omitted assertions like $1 \neq 0$

对异或门(XOR)尤其重要:

$Signal(Out(1,X_1))=1 \Leftrightarrow Signal(In(1, X_1)) \neq Signal(In(2,X_1))$

Summary

命题逻辑只是对事物的存在进行限定，而一阶逻辑对于对象和关系的存在进行限定，因而获得更强的表达能力。

First-order logic:

- objects and relations are semantic primitives（基本）
- syntax: constants, functions, predicates, equality, quantifiers
  - 语句的真值由一个模型和对句子符号的解释来判定。

Increased expressive power: sufficient to define wumpus world

在一阶逻辑中开发知识库是一个细致的过程，包括对域进行分析、选择词汇表、对支持所需推理必不可少的公理进行编码。

作业

- 8.6，8.15（第二版）=8.24(a-k)，8.17（第三版）