First-Order Logic and Inference in FOL

吉建民

USTC

jianmin@ustc.edu.cn

2024 年 4 月 2 日

Used Materials

Disclaimer: 本课件采用了 S. Russell and P. Norvig's Artificial Intelligence –A modern approach slides, 徐林莉老师课件和其他网络课程课件，也采用了 GitHub 中开源代码，以及部分网络博客内容

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

Table of Contents

Pros 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
    - 缺乏从其他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 / disjunctive / negated information
  - unlike most data structures and databases
- Propositional logic is compositional:
  - meaning of $B_{1,1} \wedge P_{1,2}$ 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 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 of propositional logic

- All students know arithmetic.
  - AliceIsStudent $\rightarrow$ AliceKnowsArithmetic
  - BobIsStudent $\rightarrow$ 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 logic	事实	信度 $\in [0,1]$

一阶逻辑的模型: Example

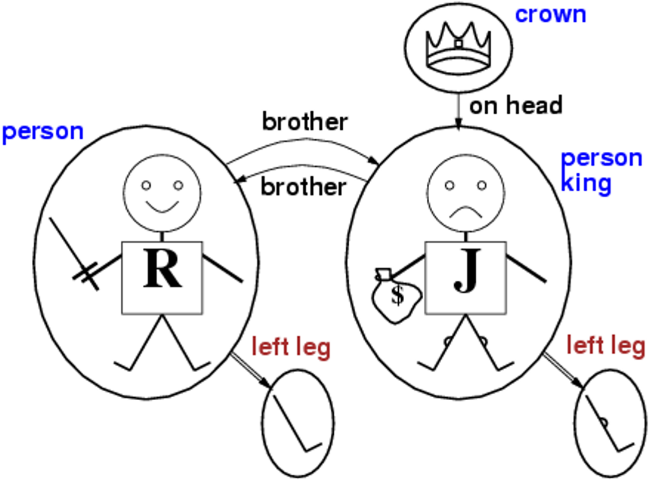


Table of Contents

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 $\rightarrow$ Richard the Lionheart
John $\rightarrow$ the evil King John
Brother $\rightarrow$ 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）

$\forall$<variables> <sentence>
"对于所有的..."

Everyone at USTC is smart:

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

$\forall x \, P$ 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

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

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 $\land$ as the main connective with $\forall$:

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

means "Everyone is at USTC and everyone is smart"

Existential quantification（存在量词）

$\exists$<variables> <sentence>
"存在一个⋯⋯，这样以致" 或 "对于某个⋯⋯"

Someone at USTC is smart:

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

$\exists x \, P$ 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

$$\begin{array}{ll} & At(KingJohn,USTC) \land Smart(KingJohn) \\ \lor & At(Richard,USTC) \land Smart(Richard) \\ \lor & At(USTC,USTC) \land Smart(USTC) \\ \lor & ... \end{array}$$

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 (**量词对偶**): each can be expressed using the other

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

  $\exists x$ Likes(x,Broccoli) $\quad \neg\forall x\ \neg$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:

$$\begin{aligned}&\forall x, y\ Sibling(x, y) \Leftrightarrow \\ &\qquad [\neg(x = y) \wedge \exists m, f\ \neg(m = f) \wedge Parent(m, x) \\ &\qquad\quad \wedge Parent(f, x) \wedge Parent(m, y) \wedge Parent(f, y)]\end{aligned}$$

Table of Contents

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) $\Rightarrow$ 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
$\forall s\ Set(s) \Leftrightarrow (s = \{\}\ ) \vee (\exists x,s_2\ Set(s_2) \wedge s = \{x|s_2\})$

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

将已经存在于集合中的元素添加到该集合，无任何变化
$\forall\ x,s\ x \in s \Leftrightarrow s = \{x|s\}$

集合的元素仅是那些被添加到集合中的元素
$\forall x,s\ x \in s \Leftrightarrow [\ \exists y,s_2\ (s = \{y|s_2\} \wedge (x = y \vee x \in s_2))]$

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

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

两个集合是相同的，当且仅当它们互为子集

$\forall s_1, s_2 \; (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \land s_2 \subseteq s_1)$

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

$\forall x, s_1, s_2 \; x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \land x \in s_2)$

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

$\forall x, s_1, s_2 \; x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \lor x \in s_2)$

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, $\{a/Shoot\} \leftarrow$ substitution （binding list 绑定表）

Given a sentence S and a substitution σ,
Sσ denotes the result of plugging σ into S; e.g.,
S = Smarter(x,y)
σ = x/Hillary, y/Bill
Sσ = Smarter(Hillary, Bill)

Ask(KB,S) returns some/all σ such that KB $\models$ Sσ

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 keeping track of change is essential

Deducing hidden properties

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

Properties of squares:
$\forall s,t$ 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

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

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

$\forall r,s$ 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:

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

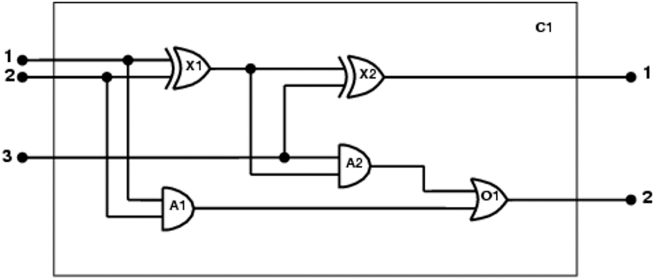
Table of Contents

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:
     $\text{Type}(X_1) = \text{XOR}$
     $\text{Type}(X_1, \text{XOR})$
     $\text{XOR}(X_1)$

The electronic circuits domain

4 Encode（编码）general knowledge of the domain

0.1 如果两个接线端是相连的，那么它们具有相同的信号

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

0.2 每个接线端的信号不是 1 就是 0（不可能两者都是）

$\forall$t Signal(t) = 1 $\vee$ Signal(t) = 0

$1 \neq 0$

0.3 Connected 是一个可交换谓词

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

0.4 或门的输出为 1，当且仅当它的某一个输入为 1

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

0.5 与门的输出为 0，当且仅当它的某一个输入为 0

$\forall$g Type(g) = AND

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

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 Type(g) = XOR

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

0.7 非门的输出与它的输入相反

$\forall$g 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

$\text{Type}(X_1) = \text{XOR}$ $\text{Type}(X_2) = \text{XOR}$
$\text{Type}(A_1) = \text{AND}$ $\text{Type}(A_2) = \text{AND}$
$\text{Type}(O_1) = \text{OR}$

其次说明门与门之间的连接

$\text{Connected}(\text{Out}(1,X_1),\text{In}(1,X_2))$ $\text{Connected}(\text{In}(1,C_1),\text{In}(1,X_1))$
$\text{Connected}(\text{Out}(1,X_1),\text{In}(2,A_2))$ $\text{Connected}(\text{In}(1,C_1),\text{In}(1,A_1))$
$\text{Connected}(\text{Out}(1,A_2),\text{In}(1,O_1))$ $\text{Connected}(\text{In}(2,C_1),\text{In}(2,X_1))$
$\text{Connected}(\text{Out}(1,A_1),\text{In}(2,O_1))$ $\text{Connected}(\text{In}(2,C_1),\text{In}(2,A_1))$
$\text{Connected}(\text{Out}(1,X_2),\text{Out}(1,C_1))$ $\text{Connected}(\text{In}(3,C_1),\text{In}(2,X_2))$
$\text{Connected}(\text{Out}(1,O_1),\text{Out}(2,C_1))$ $\text{Connected}(\text{In}(3,C_1),\text{In}(1,A_2))$

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
- 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。

Table of Contents

Universal instantiation (UI) 全称实例化

Every instantiation of a universally quantified sentence is entailed by it:
全称量化语句蕴含它的所有实例

$$\frac{\forall v \quad \alpha}{Subst(\{v/g\}, \alpha)}$$

for any variable（变量）v and ground term（基项）g

E.g., $\forall x$ King(x)$\wedge$greedy(x)$\Rightarrow$Evil(x) yields
King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)
King(Richard) $\wedge$ Greedy(Richard) $\Rightarrow$ Evil(Richard)
King(Father(John)) $\wedge$ Greedy(Father(John)) $\Rightarrow$ Evil(Father(John))
...

Existential instantiation (EI) 存在实例化

For any sentence , variable v, and new constant symbol k that does not appear elsewhere in the knowledge base:
全称量化语句蕴含它的所有实例

$$\frac{\exists v \quad \alpha}{Subst(\{v/k\}, \alpha)}$$

E.g., ∃xCrown(x)∧OnHead(x,John) yields

$$\text{Crown}(\text{C}_1)\wedge\text{OnHead}(\text{C}_1,\text{John})$$

provided C_1 is a new constant symbol, called a Skolem constant (斯科伦常数)
Another example: from $\exists$x $d(x^y)/dy = x^y$ we obtain

$$d(e^y/dy) = e^y$$

provided e is a new constant symbol

Existential instantiation contd.

UI can be applied several times to add new sentences; the new KB is logically equivalent to the old
全称实例化可以多次应用从而获得许多不同的结果

EI can be applied once to replace the existential sentence; the new KB is not equivalent to the old,
but is satisfiable iff the old KB was satisfiable
存在实例化可以应用一次，然后取代存在量化语句；
新知识库逻辑上并不等价于旧知识库，但只有在原始知识库可满足时，新的知识库才是可满足的。

Reduction to propositional inference 简化到命题逻辑推理

Suppose the KB contains just the following:
$\forall x$ King(x) $\wedge$ Greedy(x) $\Rightarrow$ Evil(x)
King(John)
Greedy(John)
Brother(Richard,John)

Instantiating the universal sentence in all possibleways, we have
King(John) $\wedge$ Greedy(John) $\Rightarrow$ Evil(John)
King(Richard) $\wedge$ Greedy(Richard) $\Rightarrow$ Evil(Richard)
King(John)
Greedy(John)
Brother(Richard,John)

The new KB is propositionalized（命题化）：proposition symbols are
King(John), Greedy(John), Evil(John), King(Richard) etc

Reduction contd.

Claim: Every FOL KB can be propositionalized so as to preserve entailment
每一个一阶逻辑知识库都可以命题化使得蕴含关系得以保持

Claim: A ground sentence is entailed by new KB iff entailed by original KB

Idea: propositionalize KB and query, apply resolution, return result

Problem: with function symbols, there are infinitely many（无限多个）ground terms（基项），
–e.g., Father(Father(Father(John)))

Reduction contd.

Theorem: Herbrand (1930). If a sentence α is entailed by an FOL KB, it is entailed by a finite subset of the propositionalized KB
定理：如果某个语句被原始的一阶知识库蕴含，则存在一个只涉及命题化知识库的有限子集的证明

Idea: For n $=0$ to ∞ do
create a propositional KB by instantiating with depth-n terms see if α is entailed by this KB

Problem: works if α is entailed, loops if α is not entailed

Theorem: Turing (1936), Church (1936) Entailment for FOL is semidecidable（半可判定的） (algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every nonentailed sentence.)

Problems with propositionalization

Propositionalization seems to generate lots of irrelevant/不相关的 sentences.

E.g., from: $\forall x$ King(x) $\wedge$Greedy(x) $\Rightarrow$ Evil(x)
King(John)
$\forall y$ Greedy(y)
Brother(Richard,John)

it seems obvious that *Evil(John)*, but propositionalization produces lots of facts such as *Greedy(Richard)* that are irrelevant

With p k-ary predicates/谓词 and n constants, there are $p \cdot n^k$ instantiations.

With function symbols, it gets much much worse!

Table of Contents

Unification（合一）

如果存在某个置换 θ 使蕴涵的前提和 KB 中已有的语句完全相同，那么应用 θ 后，就可以断言蕴涵的结论
We can get the inference immediately if we can find a substitution（置换）θ such that King(x) and Greedy(x) match King(John) and Greedy(y)

θ = x/John,y/John works

Unify(α,β)=θ if $\alpha\theta$=$\beta\theta$

p	q	θ
Knows(John,x)	Knows(John,Jane)	
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

Unification（合一）

如果存在某个置换 θ 使蕴涵的前提和 KB 中已有的语句完全相同，那么应用 θ 后，就可以断言蕴涵的结论

We can get the inference immediately if we can find a substitution（置换）θ such that King(x) and Greedy(x) match King(John) and Greedy(y)

$\theta =$ x/John,y/John works

Unify(α,β)=θ if $\alpha\theta$=$\beta\theta$

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

Unification（合一）

如果存在某个置换 θ 使蕴涵的前提和 KB 中已有的语句完全相同，那么应用 θ 后，就可以断言蕴涵的结论

We can get the inference immediately if we can find a substitution (置换) θ such that King(x) and Greedy(x) match King(John) and Greedy(y)

θ = x/John,y/John works

Unify(α,β)=θ if $\alpha\theta$=$\beta\theta$

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}
Knows(John,x)	Knows(y,Mother(y))	
Knows(John,x)	Knows(x,OJ)	

Unification（合一）

如果存在某个置换 θ 使蕴涵的前提和 KB 中已有的语句完全相同，那么应用 θ 后，就可以断言蕴涵的结论

We can get the inference immediately if we can find a substitution (置换) θ such that King(x) and Greedy(x) match King(John) and Greedy(y)

θ = x/John,y/John works

Unify(α,β)=θ if $\alpha\theta$=$\beta\theta$

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}
Knows(John,x)	Knows(x,OJ)	

Unification（合一）

如果存在某个置换 θ 使蕴涵的前提和 KB 中已有的语句完全相同，那么应用 θ 后，就可以断言蕴涵的结论

We can get the inference immediately if we can find a substitution（置换）θ such that King(x) and Greedy(x) match King(John) and Greedy(y)

θ = x/John,y/John works

Unify(α,β)=θ if $\alpha\theta=\beta\theta$

p	q	θ
Knows(John,x)	Knows(John,Jane)	{x/Jane}
Knows(John,x)	Knows(y,OJ)	{x/OJ,y/John}
Knows(John,x)	Knows(y,Mother(y))	{y/John,x/Mother(John)}
Knows(John,x)	Knows(x,OJ)	{fail}

Standardizing apart（标准化分离） eliminates overlap of variables, e.g., Knows(z_{17},OJ)

Unification（合一）

To unify Knows(John,x) and Knows(y,z),

$\theta = \{$y/John, x/z $\}$ or $\theta = \{$y/John, x/John, z/John$\}$

The first unifier is more general（更加一般） than the second.

–对变量的取值限制比较少

There is a single most general unifier (MGU) that is unique up to renaming of variables.
对每个表达式的合一对，存在一个唯一的最一般合一者，不考虑变量的重新命名它是唯一的。

MGU $= \{$ y/John, x/z $\}$

Table of Contents

Generalized Modus Ponens (GMP)

Modus Ponens（演绎推理，分离规则）(for Horn Form): complete for Horn KBs

$$\frac{\alpha_1, \ldots, \alpha_n, \quad \alpha_1 \wedge \ldots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

GMP（一般化分离规则）：

$$\frac{p_1', p_2', \ldots, p_n', (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta}$$

where $p_i'\theta = p_i\theta$ for all i

p_1' is King(John)	p_1 is King(x)
p_2' is Greedy(y)	p_2 is Greedy(x)
θ is {x/John,y/John}	q is Evil(x)
$q\theta$ is Evil(John)	

GMP used with KB of definite clauses 确定子句 (exactly one positive literal)

All variables assumed universally quantified

Semi-decidability（半可判定）

First-order logic (even restricted to only Horn clauses) is semi-decidable.

–If KB entails f, algorithms exist to prove f in finite time.

–If KB does not entail f, no algorithm can show this in finite time.

Soundness of GMP

Need to show that

$p_1',\ \ldots,\ p_n',\ (p_1 \wedge \cdots \wedge p_n \Rightarrow q) \models q\theta$

provided that $p_i'\theta = p_i\theta$ for all i

Lemma: For any sentence p, we have $p \models p\theta$

1. $(p_1 \wedge \cdots \wedge p_n \Rightarrow q) \models (p_1 \wedge \cdots \wedge p_n \Rightarrow q)\theta = (p_1\theta \wedge \cdots \wedge p_n\theta \Rightarrow q\theta)$
2. $p_1', \ldots, p_n' \models p_1' \wedge \cdots \wedge p_n' \models p_1'\theta \wedge \cdots \wedge p_n'\theta$
3. From 1 and 2, $q\theta$ follows by ordinary Modus Ponens

Completeness of GMP

- GMP: incomplete for FOL
  –Not every sentence can be converted to Horn form
- GMP: complete for FOL KB of definite clauses

Example knowledge base

The law says that it is a crime for an American to sell weapons to hostile nations.
The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.
Prove that Col. West is a criminal

Example knowledge base

...it is a crime for an American to sell weapons to hostile nations:

Example knowledge base

...it is a crime for an American to sell weapons to hostile nations:

$$\mathit{American}(x) \land \mathit{Weapon}(y) \land \mathit{Sells}(x,y,z) \land \mathit{Hostile}(z) \Rightarrow \mathit{Criminal}(x)$$

Nono ...has some missiles

Example knowledge base

...it is a crime for an American to sell weapons to hostile nations:

$$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$$

Nono ...has some missiles, i.e., $\exists x$ Owns(Nono,x) $\wedge$Missile(x):

$$Owns(Nono,M_1) \text{ and } Missile(M_1)$$

...all of its missiles were sold to it by Colonel West

Example knowledge base

...it is a crime for an American to sell weapons to hostile nations:

$$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$$

Nono ...has some missiles, i.e., $\exists x$ Owns(Nono,x) $\wedge$Missile(x):

$$Owns(Nono,M_1) \text{ and } Missile(M_1)$$

...all of its missiles were sold to it by Colonel West

$$Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$$

Missiles are weapons:

Example knowledge base

...it is a crime for an American to sell weapons to hostile nations:

$$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$$

Nono ...has some missiles, i.e., $\exists x\ Owns(Nono,x) \wedge Missile(x)$:

$$Owns(Nono,M_1) \text{ and } Missile(M_1)$$

...all of its missiles were sold to it by Colonel West

$$Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$$

Missiles are weapons:

$$Missile(x) \Rightarrow Weapon(x)$$

An enemy of America counts as "hostile":

Example knowledge base

...it is a crime for an American to sell weapons to hostile nations:

$$American(x) \wedge Weapon(y) \wedge Sells(x,y,z) \wedge Hostile(z) \Rightarrow Criminal(x)$$

Nono ...has some missiles, i.e., $\exists x\ Owns(Nono,x) \wedge Missile(x)$:

$$Owns(Nono, M_1) \text{ and } Missile(M_1)$$

...all of its missiles were sold to it by Colonel West

$$Missile(x) \wedge Owns(Nono,x) \Rightarrow Sells(West,x,Nono)$$

Missiles are weapons:

$$Missile(x) \Rightarrow Weapon(x)$$

An enemy of America counts as "hostile":

$$Enemy(x,America) \Rightarrow Hostile(x)$$

West, who is American ...

$$American(West)$$

The country Nono, an enemy of America ...

$$Enemy(Nono,America)$$

Table of Contents

Forward chaining algorithm

```
function FOL-FC-ASK(KB, α) returns a substitution or false

  repeat until new is empty
      new ← { }
      for each sentence r in KB do
          (p1 ∧ . . . ∧ pn ⇒ q) ← STANDARDIZE-APART(r)
          for each θ such that (p1 ∧ . . . ∧ pn)θ = (p'1 ∧ . . . ∧ p'n)θ
                      for some p'1, . . . , p'n in KB
              q' ← SUBST(θ, q)
              if q' is not a renaming of a sentence already in KB or new then do
                  add q' to new
                  φ ← UNIFY(q', α)
                  if φ is not fail then return φ
      add new to KB
  return false
```

Example knowledge base

...it is a crime for an American to sell weapons to hostile nations:

$$\mathit{American}(x) \wedge \mathit{Weapon}(y) \wedge \mathit{Sells}(x,y,z) \wedge \mathit{Hostile}(z) \Rightarrow \mathit{Criminal}(x)$$

Nono ...has some missiles, i.e., $\exists x$ Owns(Nono,x) $\wedge$Missile(x):

$$\mathit{Owns}(\mathit{Nono},M_1) \text{ and } \mathit{Missile}(M_1)$$

...all of its missiles were sold to it by Colonel West

$$\mathit{Missile}(x) \wedge \mathit{Owns}(\mathit{Nono},x) \Rightarrow \mathit{Sells}(\mathit{West},x,\mathit{Nono})$$

Missiles are weapons:

$$\mathit{Missile}(x) \Rightarrow \mathit{Weapon}(x)$$

An enemy of America counts as "hostile":

$$\mathit{Enemy}(x,\mathit{America}) \Rightarrow \mathit{Hostile}(x)$$

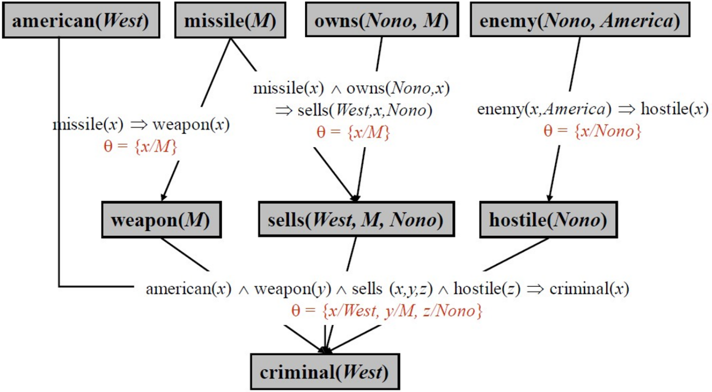
West, who is American ...

$$\mathit{American}(\mathit{West})$$

The country Nono, an enemy of America ...

$$\mathit{Enemy}(\mathit{Nono},\mathit{America})$$

Forward chaining proof



Properties of forward chaining

Sound and complete for first-order definite clauses
(proof similar to propositional proof)

Datalog $=$ first-order definite clauses $+$ no functions (e.g., crime KB)
FC terminates for Datalog in poly iterations: at most $p \cdot n^k$ literals

May not terminate in general if α is not entailed

This is unavoidable: entailment with definite clauses is semidecidable（半可判定的）

Efficiency of forward chaining

Simple observation: no need to match a rule on iteration *k*
if a premise wasn't added on iteration *k-1*
$\Rightarrow$ match each rule whose premise contains a newly added literal
Matching itself can be expensive
Database indexing allows *O(1)* retrieval of known facts
  e.g., query *Missile(x)* retrieves $Missile(M_1)$
Matching conjunctive premises against known facts is NP-hard
把确定子句与事实集相匹配是一个 NP 难题

Backward chaining algorithm

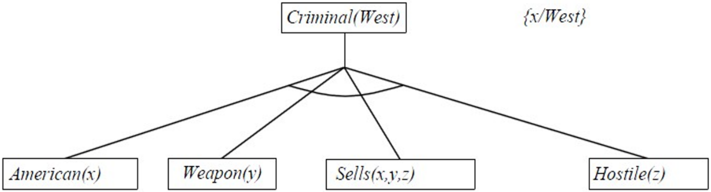
function FOL-BC-ASK(KB, $goals$, θ) **returns** a set of substitutions
  inputs: KB, a knowledge base
    $goals$, a list of conjuncts forming a query (θ already applied)
    θ, the current substitution, initially the empty substitution $\{\,\}$
  local variables: $answers$, a set of substitutions, initially empty

  if $goals$ is empty **then return** $\{\theta\}$
  $q' \leftarrow \textsc{Subst}(\theta, \textsc{First}(goals))$
  for each sentence r **in** KB
    where $\textsc{Standardize-Apart}(r) = (p_1 \land \ldots \land p_n \Rightarrow q)$
    and $\theta' \leftarrow \textsc{Unify}(q, q')$ succeeds
   $new\_goals \leftarrow [p_1, \ldots, p_n | \textsc{Rest}(goals)]$
   $answers \leftarrow \text{FOL-BC-ASK}(KB, new\_goals, \textsc{Compose}(\theta', \theta)) \cup answers$
  return $answers$

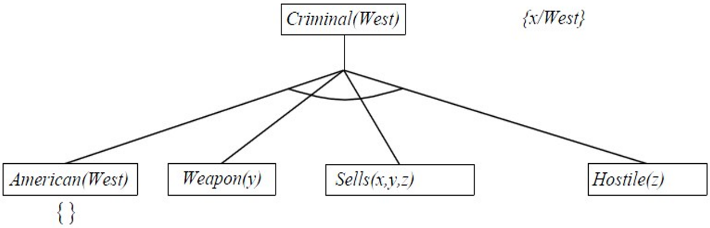
Backward chaining example

$\boxed{Criminal(West)}$

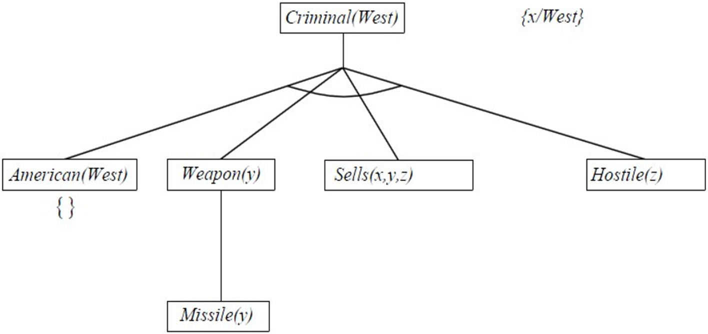
Backward chaining example



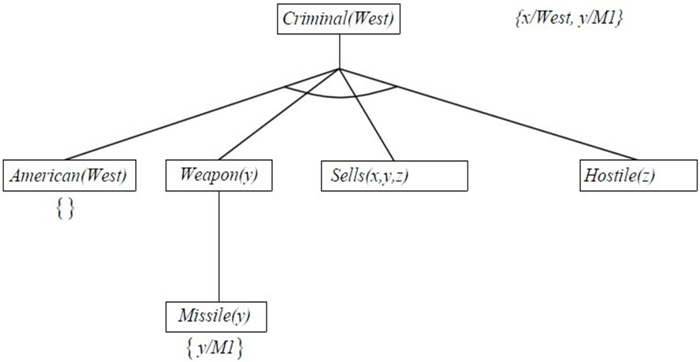
Backward chaining example



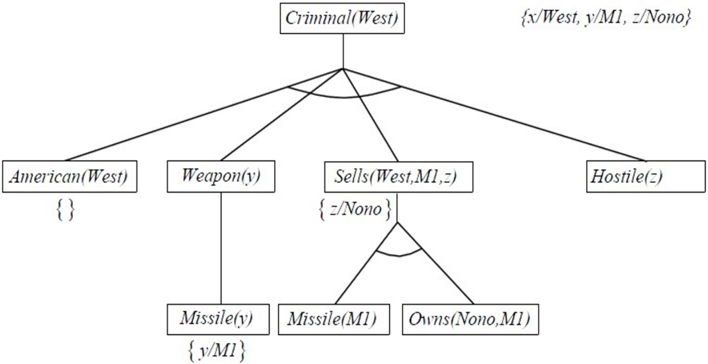
Backward chaining example



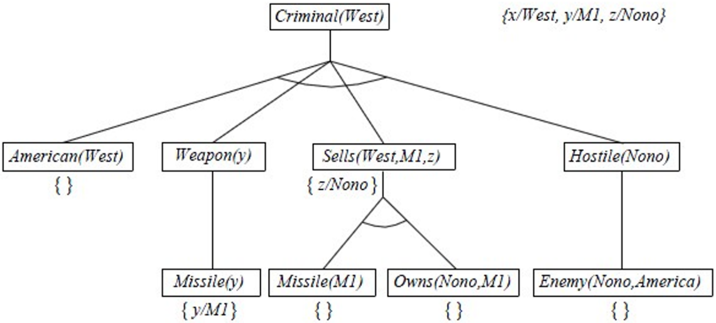
Backward chaining example



Backward chaining example



Backward chaining example



Properties of backward chaining

Depth-first recursive proof search: space is linear in size of proof

Incomplete due to infinite loops
$\Rightarrow$fix by checking current goal against every goal on stack

Inefficient due to repeated subgoals (both success and failure)
$\Rightarrow$fix using caching of previous results (extra space)

Widely used for logic programming（逻辑程序设计）

Completeness of FC/BC for General FOL

- FC and BC are complete for Horn KBs but are incomplete for general FOL KBs:

```
PhD(x)    ⇒ HighlyQualified(x)
¬PhD(x)   ⇒ EarlyEarnings(x)
HighlyQualified(x) ⇒ Rich(x)
EarlyEarnings(x)   ⇒ Rich(x)
Query: Rich(Me)
```

- Can't prove query with FC or BC. Why?
- Does a complete algorithm for FOL exist?

Table of Contents

Resolution algorithm

- Recall: KB operation boil down to satisfiability

$$KB \models \alpha \text{ if and only if } (KB \wedge \neg\alpha) \text{ is unsatisfiable}$$

- Algorithm: resolution-based inference
  - Convert all formulas to CNF
  - Repeatedly apply resolution rule
  - Return unsatisfiable iff derive false –empty clause

Resolution: brief summary

Full first-order version:

$$\frac{l_1 \vee \cdots \vee l_k, \quad m_1 \vee \cdots \vee m_n}{l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n}$$

where Unify$(l_i, \neg m_j) = \theta$.

The two clauses are assumed to be standardized apart so that they share no variables.——**假设两个子句已经标准化分离，没有共享变量。**

For example,

$$\frac{\neg Rich(x) \vee Unhappy(x) \qquad Rich(Ken)}{Unhappy(Ken)}$$

with $\theta = \{x/Ken\}$

Apply resolution steps to CNF(KB $\wedge \neg\alpha$); complete for FOL

Conversion to CNF

Everyone who loves all animals is loved by someone:

$\forall x\ [\forall y\ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y\ Loves(y,x)]$

1. Eliminate biconditionals and implications—消除蕴含

   $\forall x\ [\neg\forall y\ \neg Animal(y) \vee Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

2. Move $\neg$inwards —将 $\neg$ 内移: $\neg\forall x\ p \equiv \exists x\ \neg p$, $\neg x\ p \equiv \forall x\ \neg p$

   $\forall x\ [\exists y\ \neg(\neg Animal(y) \vee \neg Loves(x,y))] \vee [\exists y\ Loves(y,x)]$

   $\forall x\ [\exists y\ \neg\neg Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

   $\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x,y)] \vee [\exists y\ Loves(y,x)]$

Conversion to CNF contd.

Everyone who loves all animals is loved by someone:

$\forall x\ [\forall y\ Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists y\ Loves(y,x)]$

3 Standardize variables—变量标准化: each quantifier should use a different one

$$\forall x\ [\exists y\ Animal(y) \land \neg Loves(x,y)] \lor [\exists z\ Loves(z,x)]$$

4 Skolemize: a more general form of existential instantiation.

Each existential variable is replaced by a Skolem function（斯克伦函数）of the enclosing universally quantified variables:

$\forall x\ [Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$

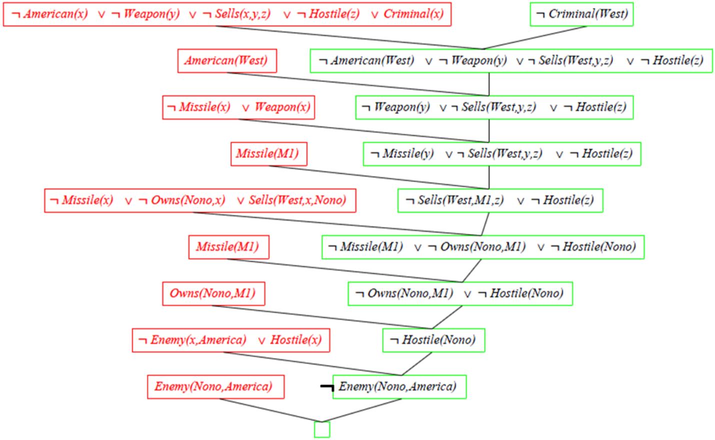
5 Drop universal quantifiers—(去除全称量词）:

$[Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x)$

6 Distribute $\lor$ over $\land$——将 $\lor$ 分配到 $\land$ 中

$[Animal(F(x)) \lor Loves(G(x),x)] \land [\neg Loves(x,F(x)) \lor Loves(G(x),x)]$

Resolution proof: definite clauses



A brief history of reasoning

450B.C.	Stoics	propositional logic, inference (maybe)
322B.C.	Aristotle	"syllogisms" (inference rules), quantifiers
1565	Cardano	probability theory (propositional logic + uncertainty)
1847	Boole	propositional logic (again)
1879	Frege	first-order logic
1922	Wittgenstein	proof by truth tables
1930	Gödel	$\exists$ complete algorithm for FOL
1930	Herbrand	complete algorithm for FOL (reduce to propositional)
1931	Gödel	$\neg\exists$ complete algorithm for arithmetic
1960	Davis/Putnam	"practical" algorithm for propositional logic
1965	Robinson	"practical" algorithm for FOL—resolution

Summary

- 一阶逻辑中的逻辑推理
  - 命题化推理问题/Reducing first-order inference to propositional inference
    - 效率较低
  - 合一/ Unification
    - 用于确定适当的变量置换
  - 一般化分离规则/ Generalized Modus Ponens
    - 确定子句/definite clauses
    - 可靠的，完备的
    - 应用于前向链接和反向链接算法
  - 前向链接，反向链接
  - 归结推理/Resolution

Summary

Propositional logic	First-order logic
•Model checking	•n/a
←propositionalization	
•Modus ponens (Horn clauses)	•Modus ponens++ (Horn clauses)
•Resolution (general)	•Resolution++ (general)

++: unification and substitution

Key idea: variables in first-order logic

Variables yield compact knowledge representations.

Homework

- 8.24 (a-k)，8.17　　（第三版）
- 9.3，9.4，9.6，9.13 (a,b,c)　　（第三版）