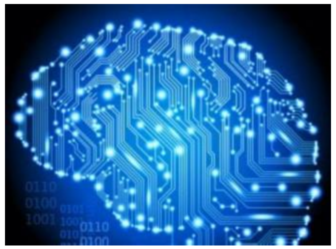
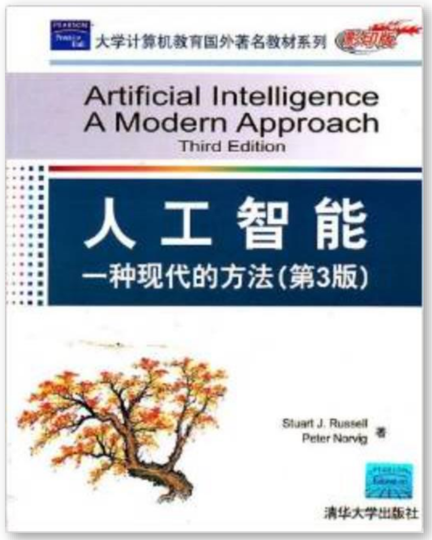
Knowledge 1

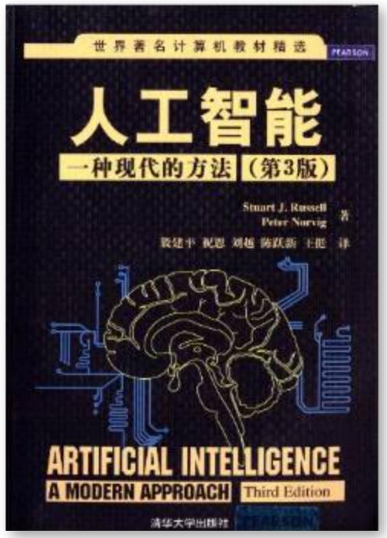


教材

http://www.ai-fundamental.com/

Materials on Knowledge





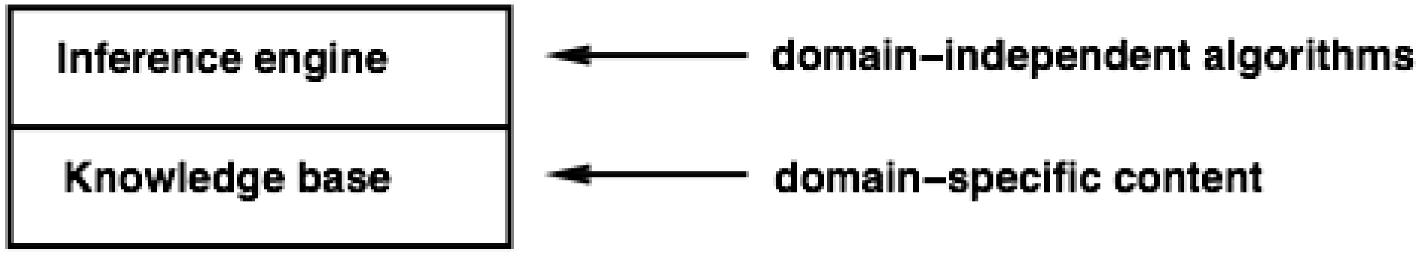
课程第二部分：
台湾大学 于天立教授 “人工智慧”课程

链接: http://pan.baidu.com/s/1miec9jM 密码: 4fd4

http://aima.cs.berkeley.edu/

中英文版本对照阅读

Knowledge bases



Knowledge base = set of sentences in a **formal** language

Declarative approach to building an agent (or other system):

  TELL it what it needs to know 只需要告知想知道什么

Then it can ASK itself what to do—answers should follow from the KB

Agents can be viewed at the knowledge level

  i.e., **what they know**, regardless of how implemented

Or at the implementation level

  i.e., data structures in KB and algorithms that manipulate them

数理逻辑

《计算机科学的数理逻辑》
陆钟万著

授课视频：
http://www.1ketang.com/course/2025.html

Wumpus World PEAS description

Performance measure

gold +1000, death -1000

-1 per step, -10 for using the arrow

Environment

Squares adjacent to wumpus are smelly

Squares adjacent to pit are breezy

Glitter iff gold is in the same square

Shooting kills wumpus if you are facing it

Shooting uses up the only arrow

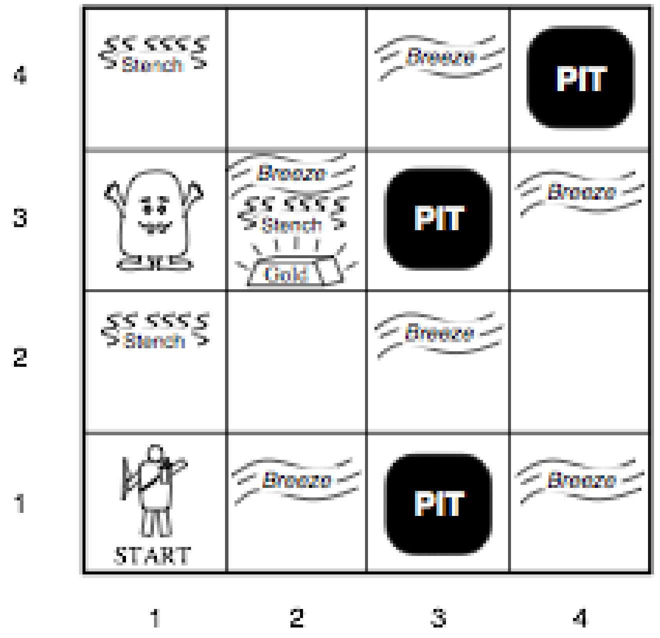
Grabbing picks up gold if in same square

Releasing drops the gold in same square

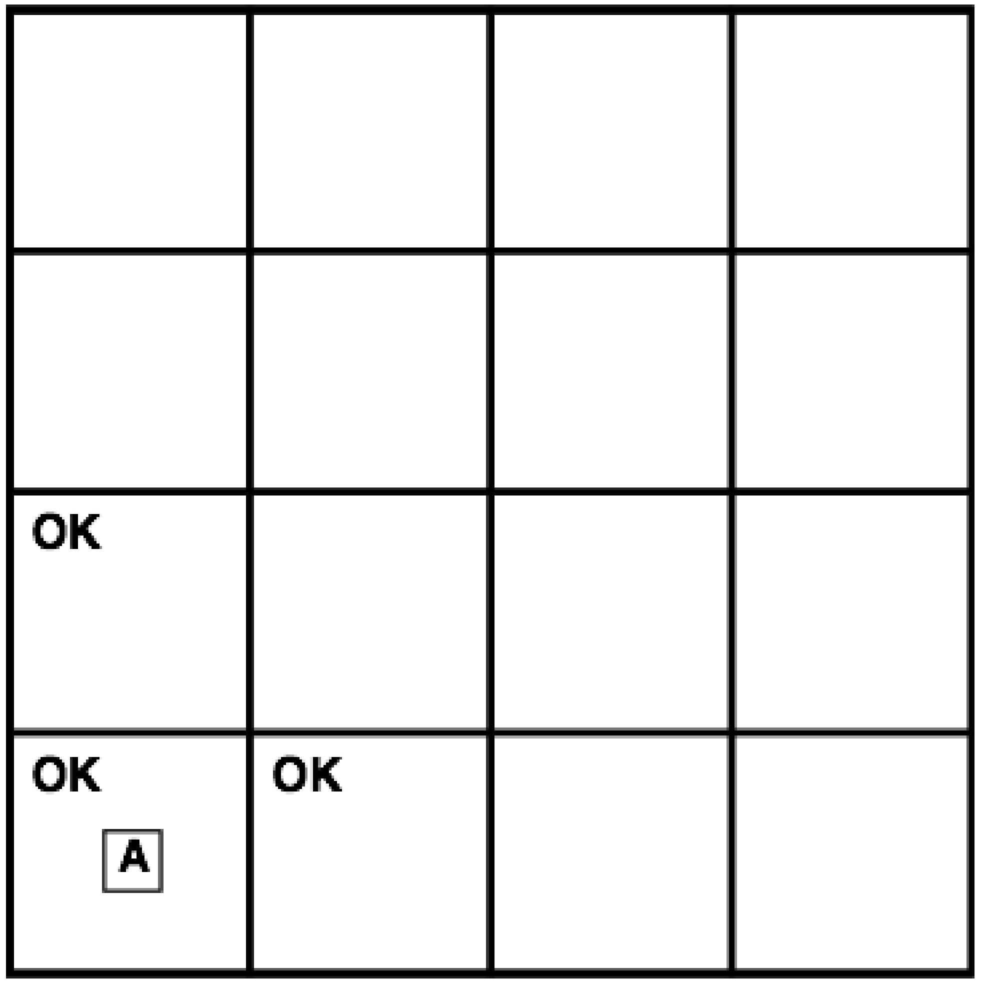
Actuators Left turn, Right turn,

Forward, Grab, Release, Shoot

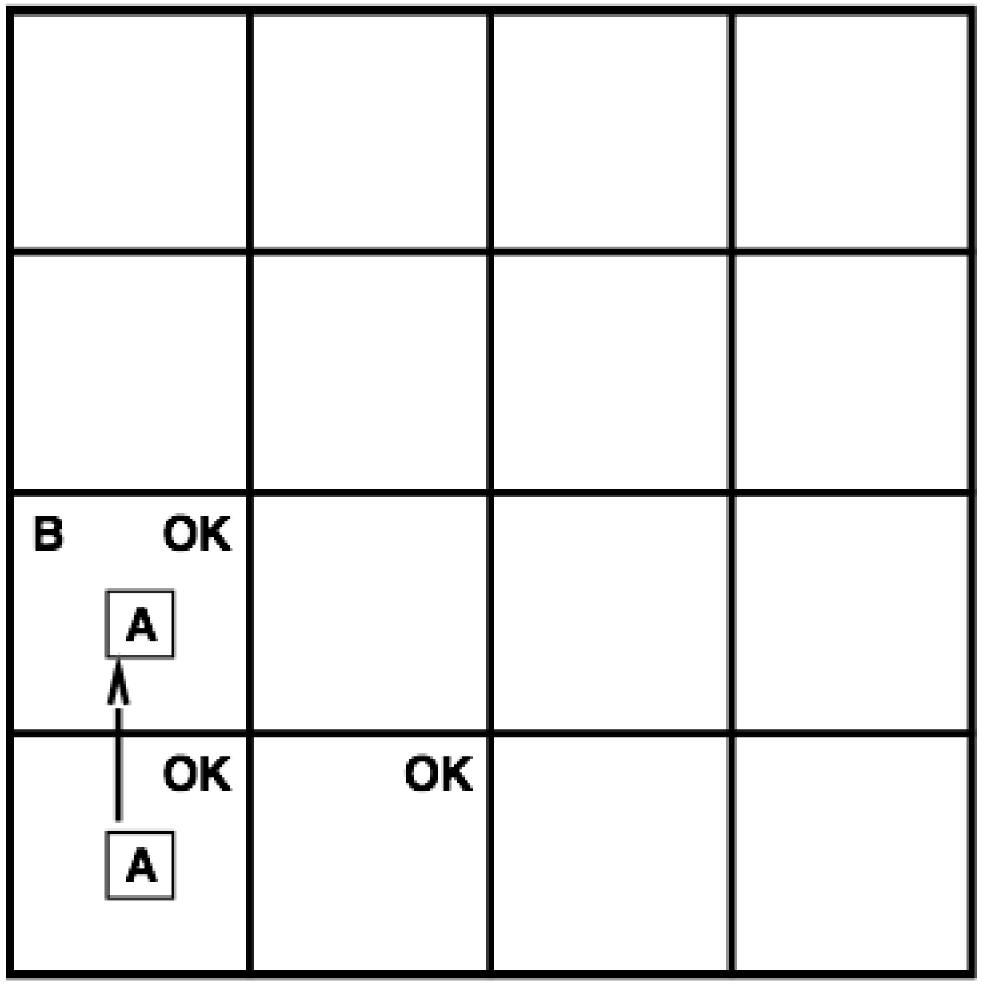
Sensors Breeze, Glitter, Smell



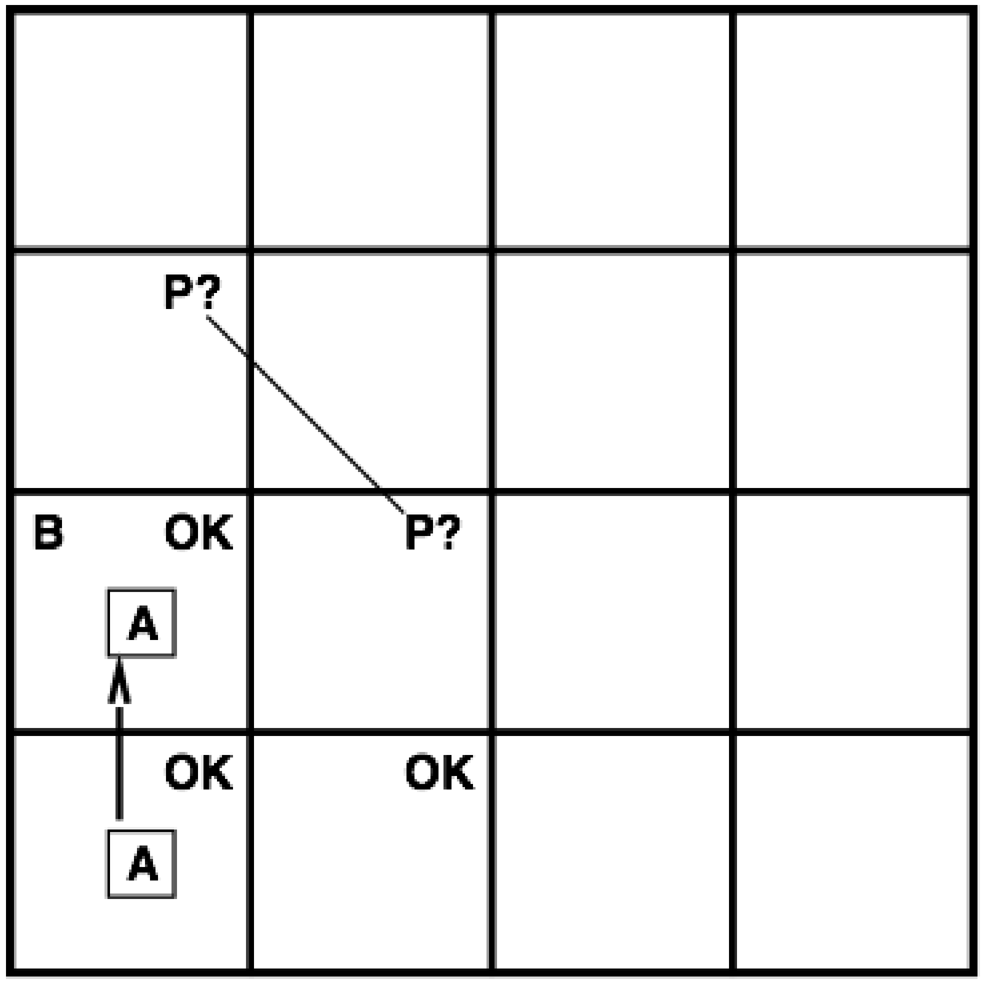
Exploring a wumpus world



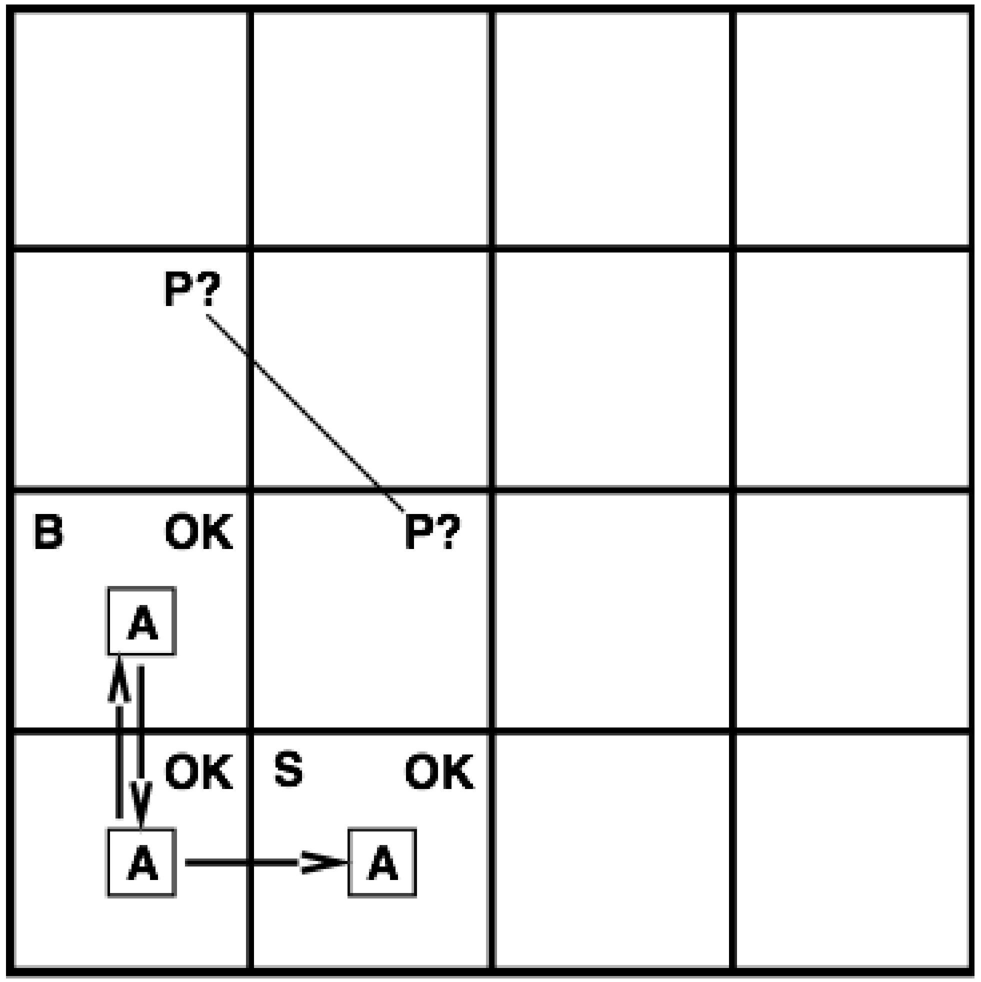
Exploring a wumpus world



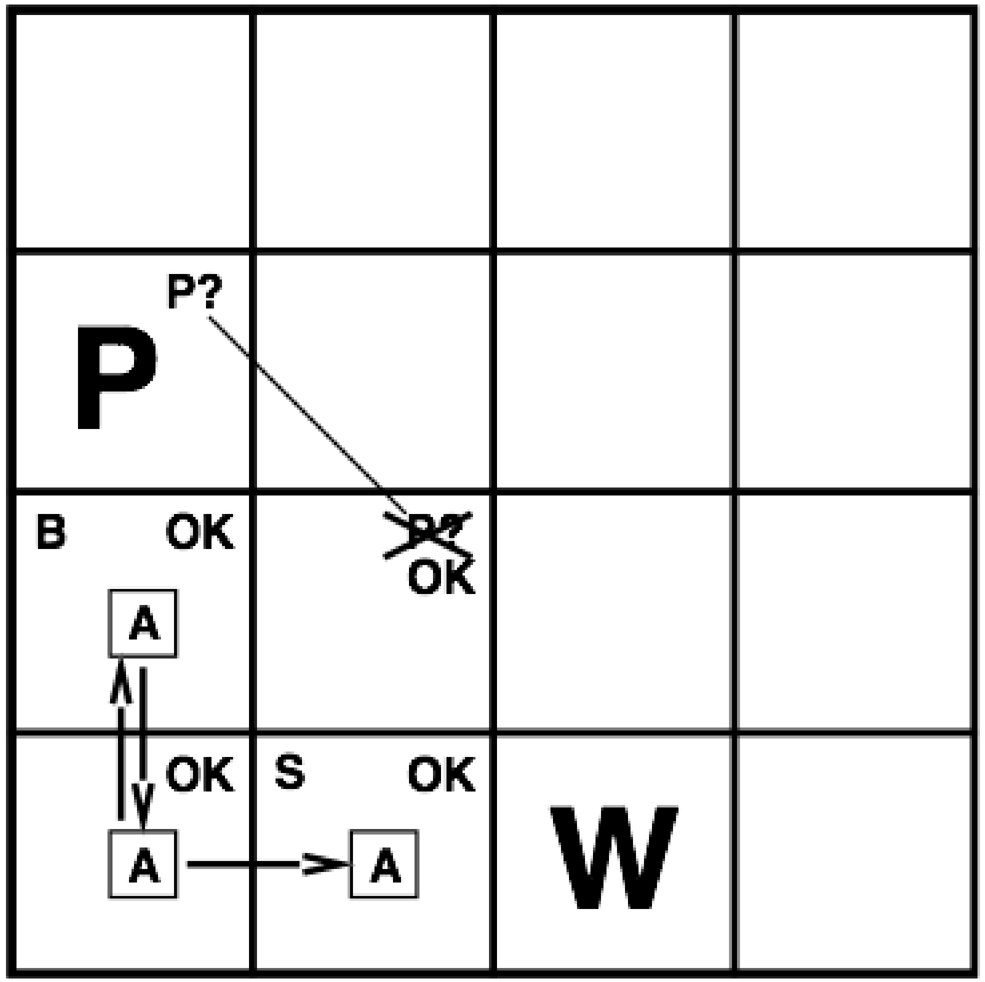
Exploring a wumpus world



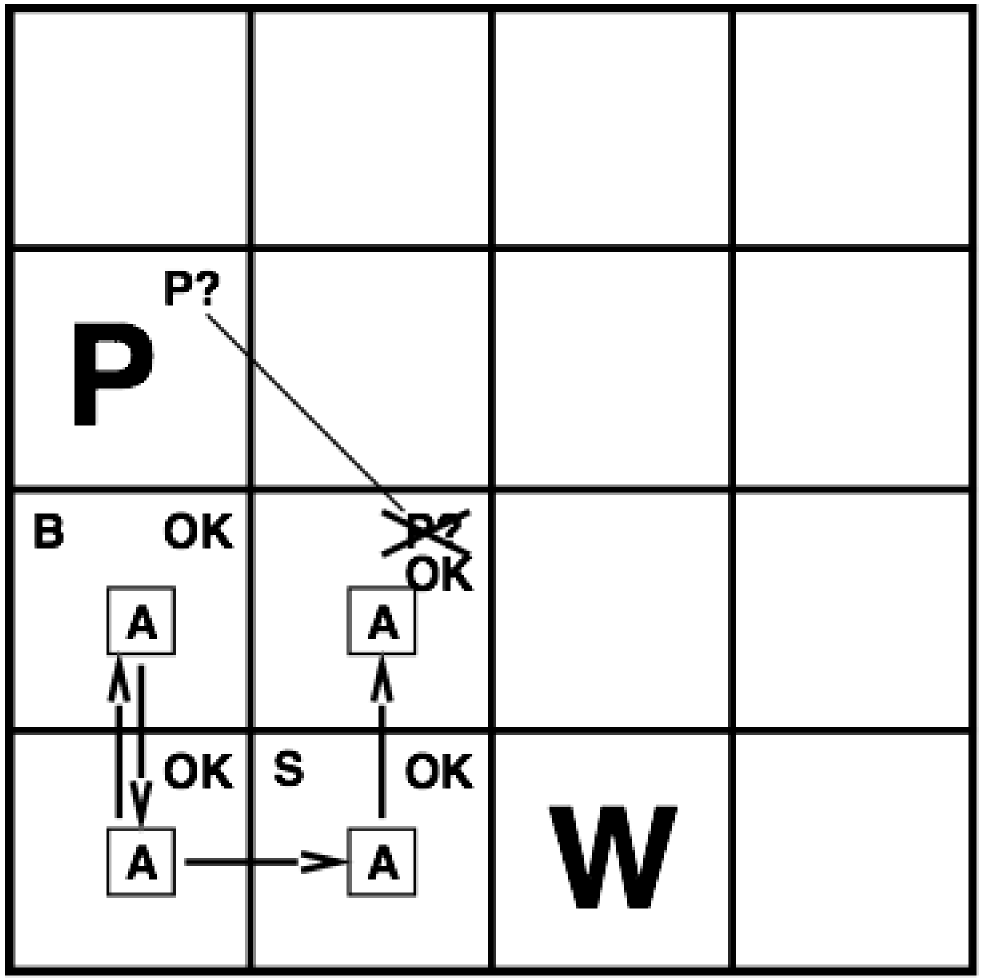
Exploring a wumpus world



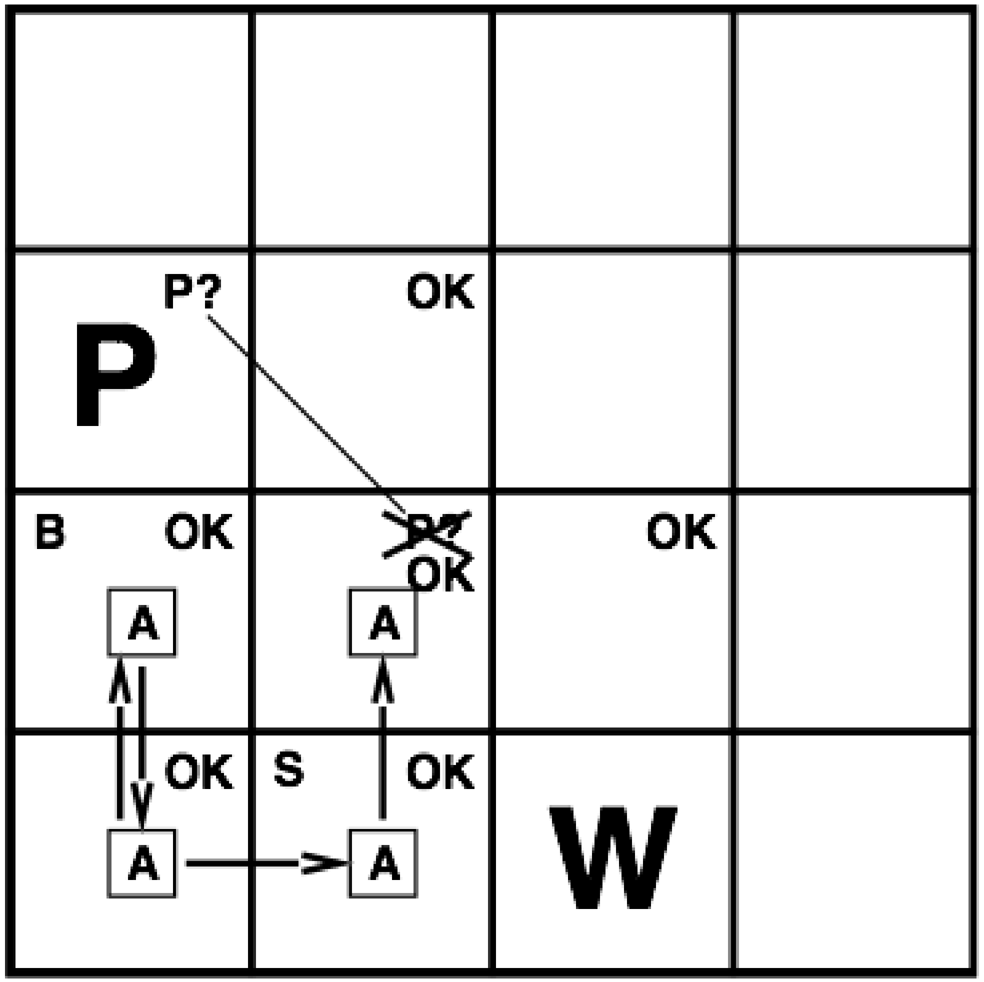
Exploring a wumpus world



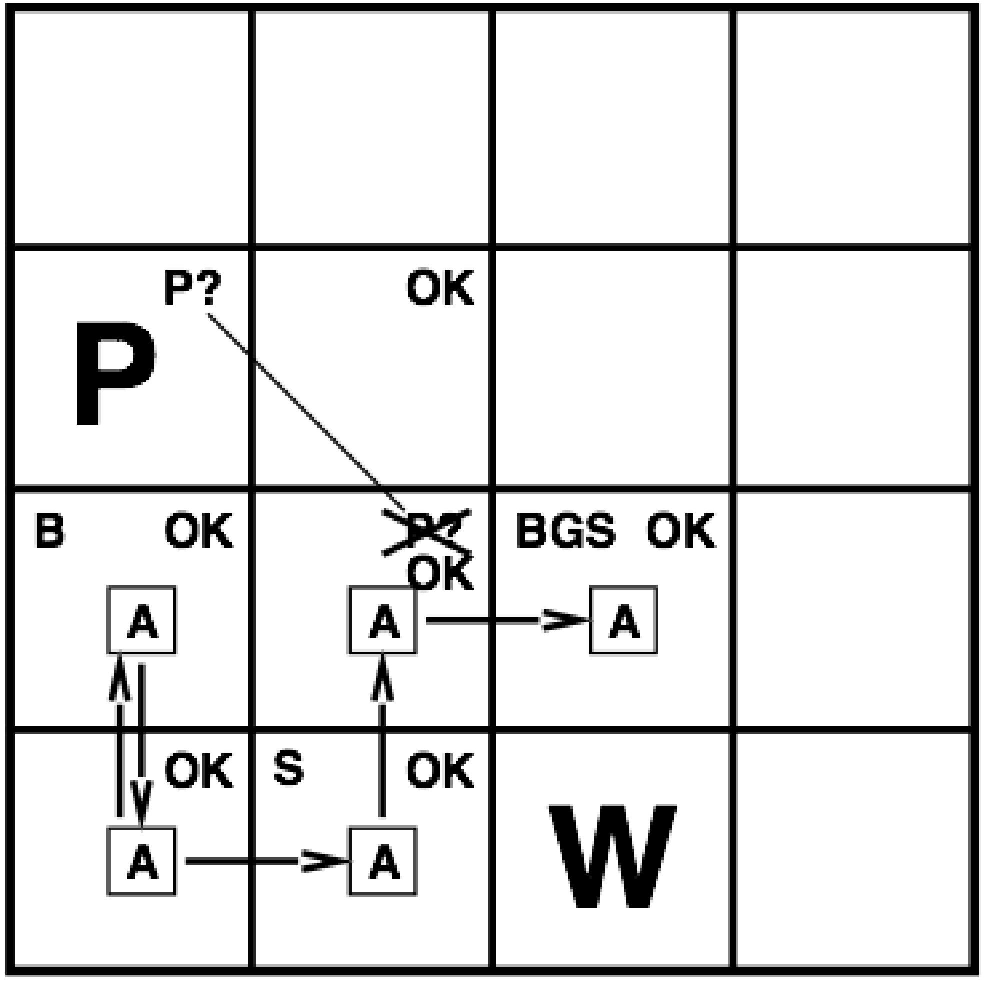
Exploring a wumpus world



Exploring a wumpus world



Exploring a wumpus world



Logic in general

Logics are formal languages for representing information
such that conclusions can be drawn

Syntax defines the sentences in the language 语法

Semantics define the "meaning" of sentences;
i.e., define truth of a sentence in a world 语义

E.g., the language of arithmetic

$x + 2 \geq y$ is a sentence; $x2 + y >$ is not a sentence

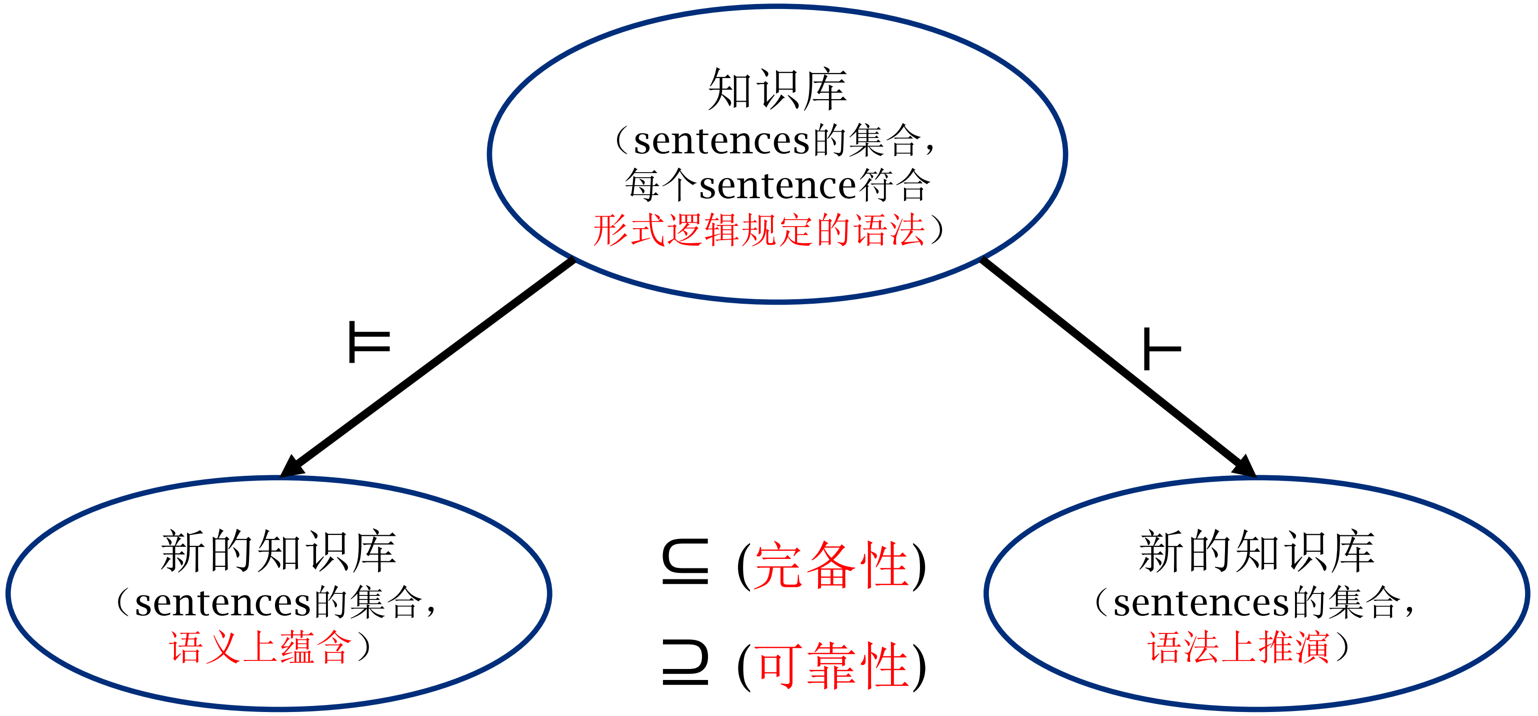
$x + 2 \geq y$ is true iff the number $x + 2$ is no less than the number y

$x + 2 \geq y$ is true in a world where $x = 7,\ y = 1$
$x + 2 \geq y$ is false in a world where $x = 0,\ y = 6$

逻辑研究的内容

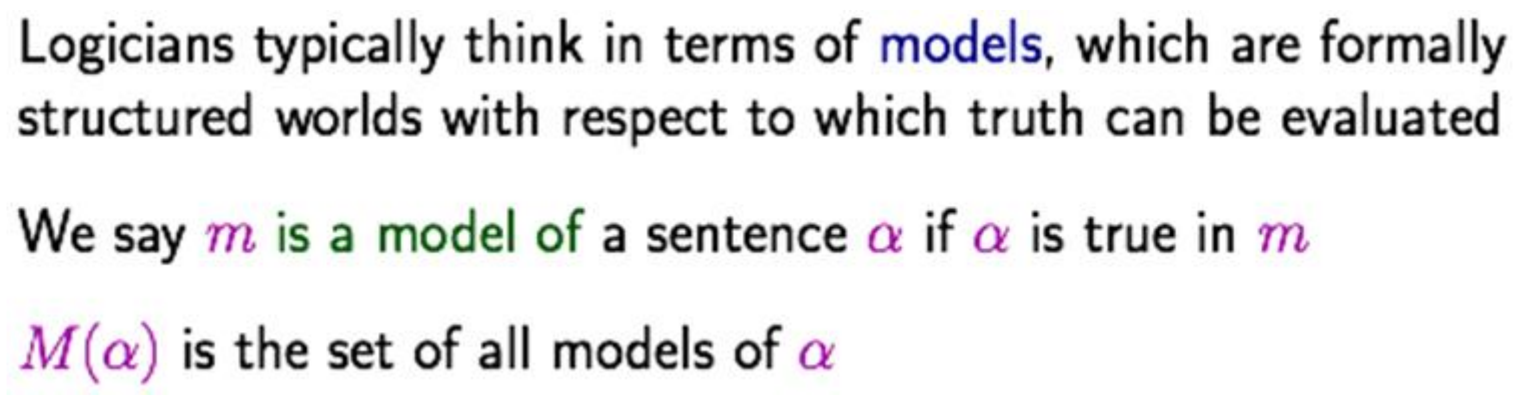
研究形式化定义的sentences之间的关系

两个角度：
语义：entailment 蕴含，逻辑推导
语法：deduction 演绎，形式推演



语义 Entailment

Model (模型), Truth Assignment (真值指派)



Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated

We say m is a model of a sentence α if α is true in m

$M(\alpha)$ is the set of all models of α

X+Y=4

X=0,Y=4是这个句子的model

Entailment (蕴涵 / 蕴含)

语义：逻辑推导

Entailment means that one thing **follows from** another:

$$KB \models \alpha$$

KB是True

α 一定是True

所以KB entail α

Knowledge base KB entails sentence α
if and only if
α is true in all worlds where KB is true

E.g., the KB containing "the Giants won" and "the Reds won" entails "Either the Giants won or the Reds won"

E.g., $x + y = 4$ entails $4 = x + y$

Entailment is a relationship between sentences (i.e., **syntax**) that is based on **semantics**

Note: brains process **syntax** (of some sort)

Model (模型)

Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated

We say m is a model of a sentence α if α is true in m $\forall$

$m \in M(KB)$

$m \in M(\alpha)$

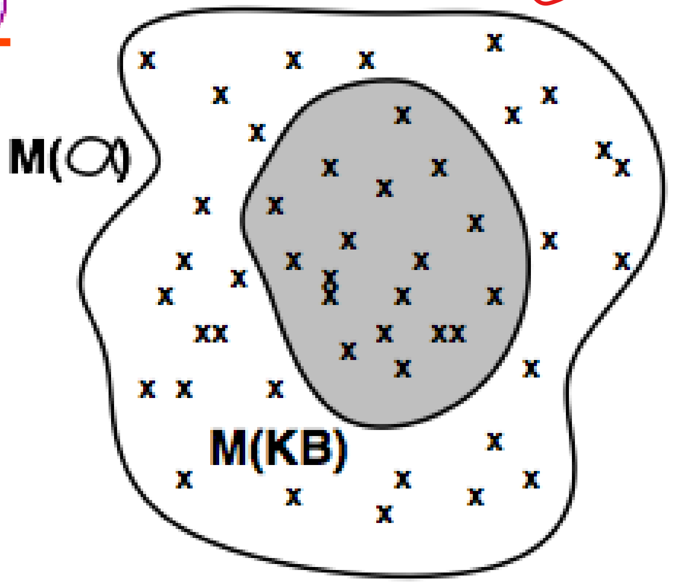
$M(\alpha)$ is the set of all models of α

$\forall$

Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$

E.g. KB = Giants won and Reds won

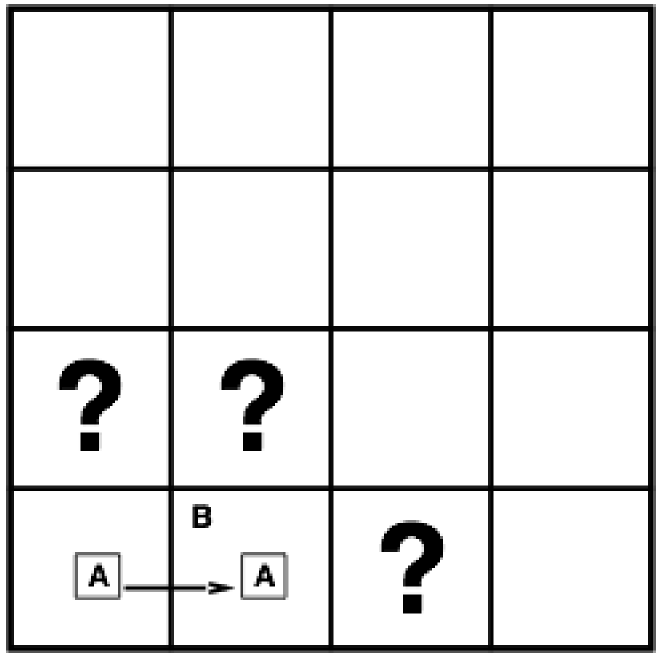
α = Giants won



Formal proof

Entailment in the wumpus world

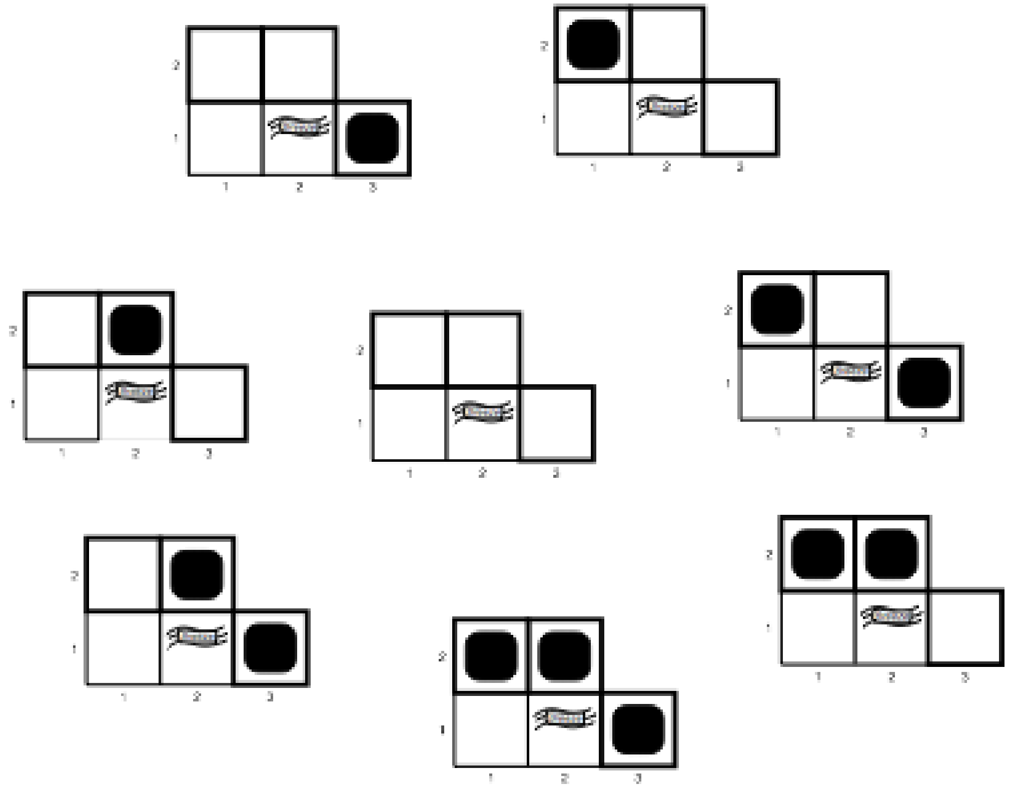
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]



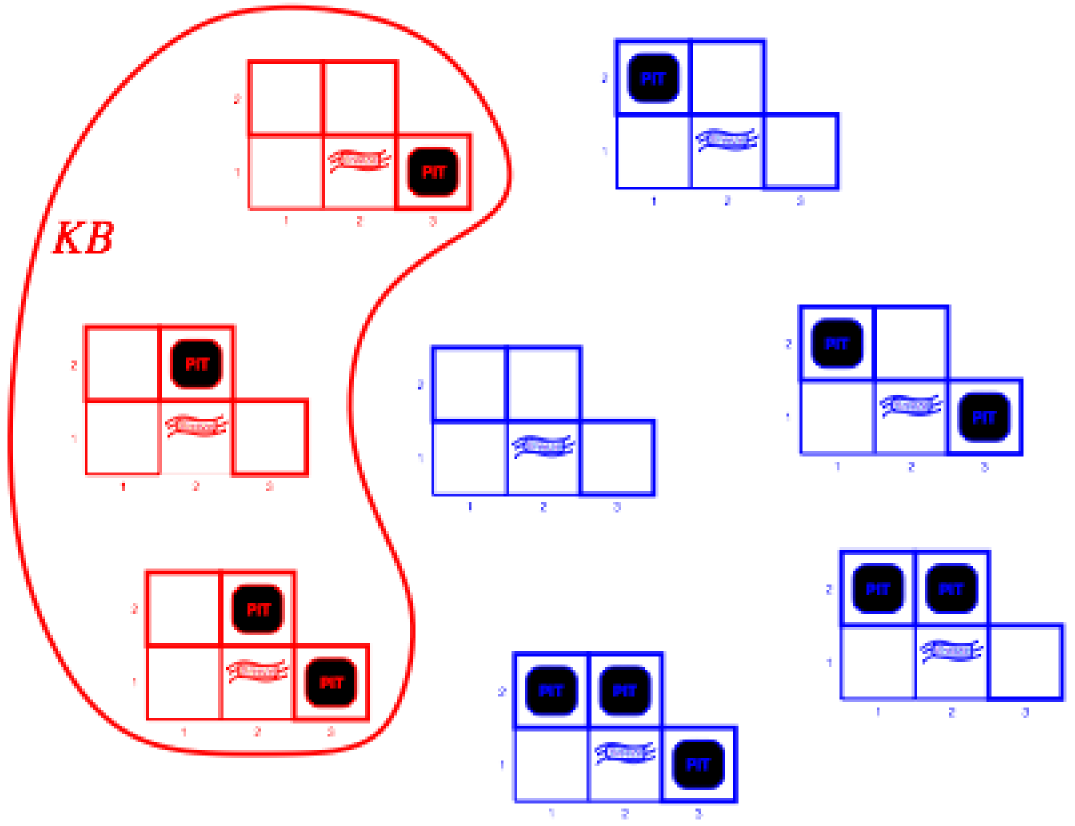
Consider possible models for ?s assuming only pits

3 Boolean choices $\Rightarrow$ 8 possible models

Wumpus models

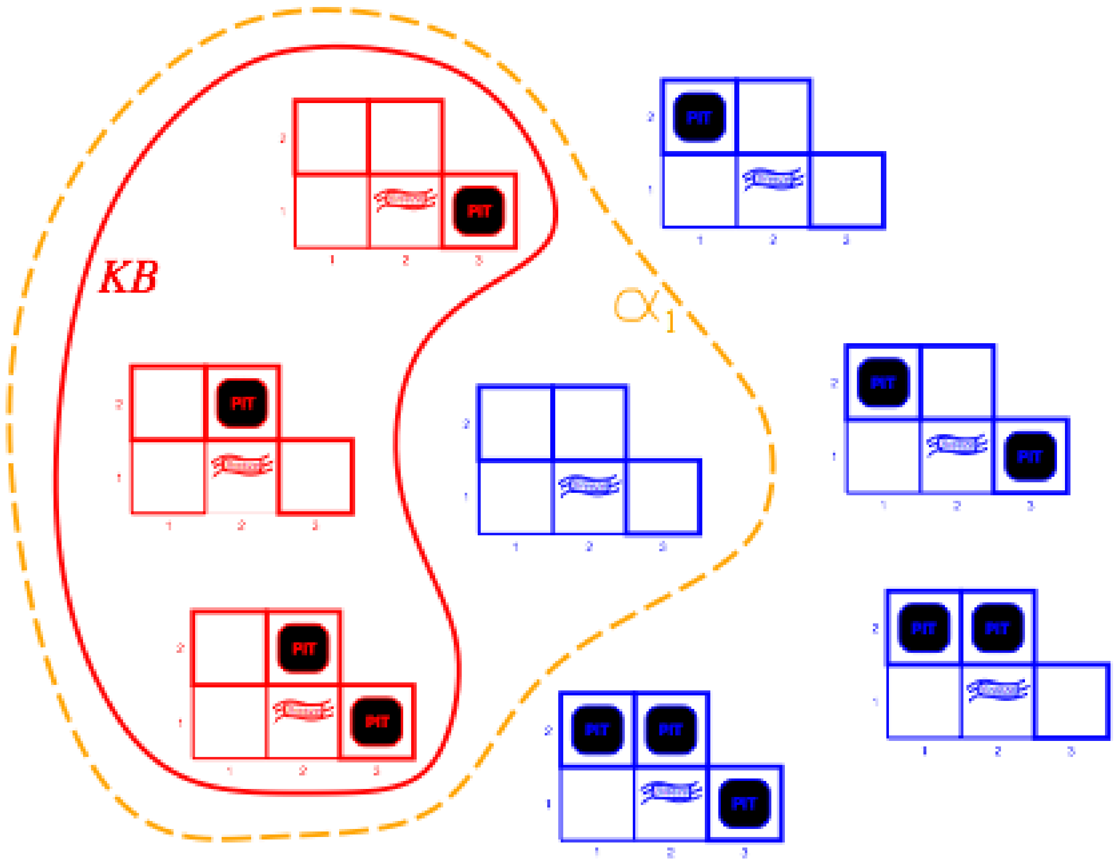


Wumpus models



KB = wumpus-world rules + observations

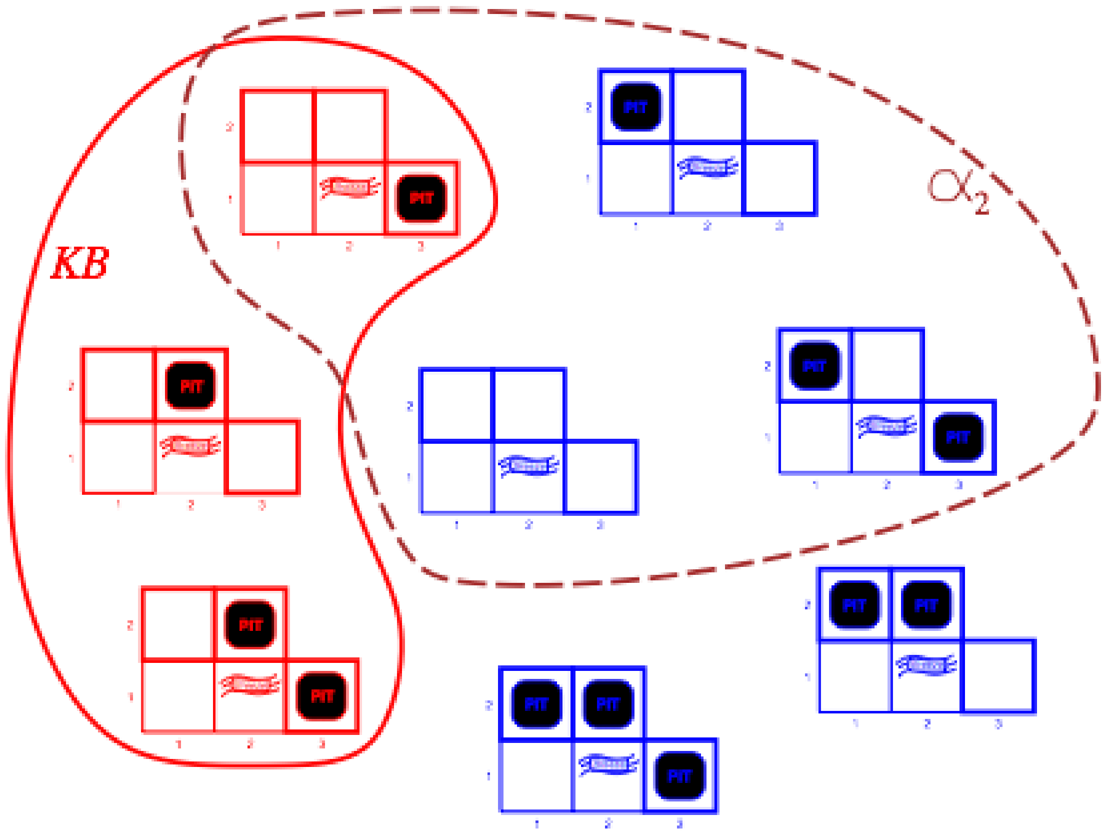
Wumpus models



KB = wumpus-world rules + observations

α_1 = "[1,2] is ~~safe~~", $KB \models \alpha_1$, proved by model checking
not pit

Wumpus models



KB = wumpus-world rules + observations

α_2 = "[2,2] is ~~safe~~", $KB \not\models \alpha_2$
not pit

Propositional logic: syntax and semantics

Propositional logic (命题逻辑): Syntax

- Proposition: a declarative sentence that is either true or false 陈述句
- Propositional logic usually does not consider time
- If the truth of a proposition varies over time, we call it fluent ("today is Monday")

- Atomic propositions (原子命题) are minimal propositions
- Literals (文字) are atomic propositions or their negations

Propositional logic (命题逻辑): Syntax

Propositional logic is the simplest logic—illustrates basic ideas

The proposition symbols P_1, P_2 etc are sentences

If S is a sentence, ❤ S is a sentence (negation)

If S_1 and S_2 are sentences, S_1 ☺ S_2 is a sentence (conjunction)

If S_1 and S_2 are sentences, S_1 ⭘ S_2 is a sentence (disjunction)

If S_1 and S_2 are sentences, S_1 ✹ S_2 is a sentence (implication)

If S_1 and S_2 are sentences, S_1 ☁ S_2 is a sentence (biconditional)

Propositional logic (命题逻辑): Syntax

Defined in Backus-Naur form (BNF)

Sentence	$\rightarrow$	*AtomicSentence* | *ComplexSentence*	
AtomicSentence	$\rightarrow$	*True* | *False* | *P* | *Q* | *R* | ...	
ComplexSentence	$\rightarrow$	(*Sentence*) | [*Sentence*]	
	|	♥ *Sentence*	$\neg$
	|	*Sentence* ☺ *Sentence*	$\wedge$
	|	*Sentence* ○ *Sentence*	
	|	*Sentence* ✹ *Sentence*	$\Rightarrow$
	|	*Sentence* ☁ *Sentence*	
OPERATOR PRECEDENCE	:	♥ ☺ ○ ✹ ☁	$\Leftrightarrow$

Propositional logic: Semantics

Each model specifies true/false for each proposition symbol

E.g.	$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
	~~$true$~~ false	$true$	$false$

(With these symbols, 8 possible models, can be enumerated automatically.)

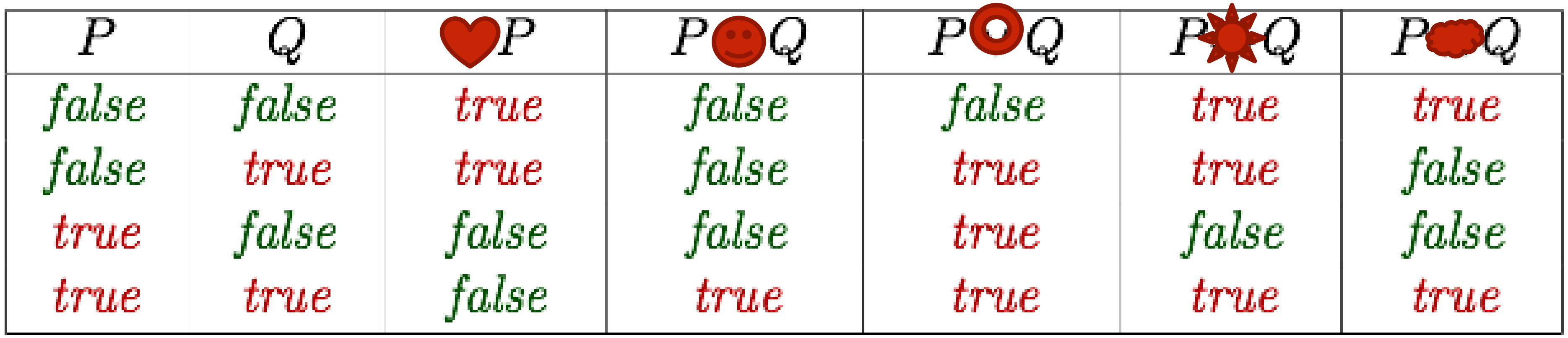
Rules for evaluating truth with respect to a model m:

❤S	is true iff	S	is false		
S_1 ☺ S_2	is true iff	S_1	is true **and**	S_2	is true
S_1 ○ S_2	is true iff	S_1	is true **or**	S_2	is true
S_1 ✸ S_2	is true iff	S_1	is false **or**	S_2	is true
i.e.,	is false iff	S_1	is true **and**	S_2	is false
S_1 ☁ S_2	is true iff	S_1 ✸ S_2	is true **and**	S_2 ✸ S_1	is true

Simple recursive process evaluates an arbitrary sentence, e.g.,

❤$P_{1,2}$ ☺ $(P_{2,2}$ ○ $P_{3,1}) = true$ ☺ $(false$ ○ $true) = true$ ☺ $true = true$

Truth tables for connectives （真值表）

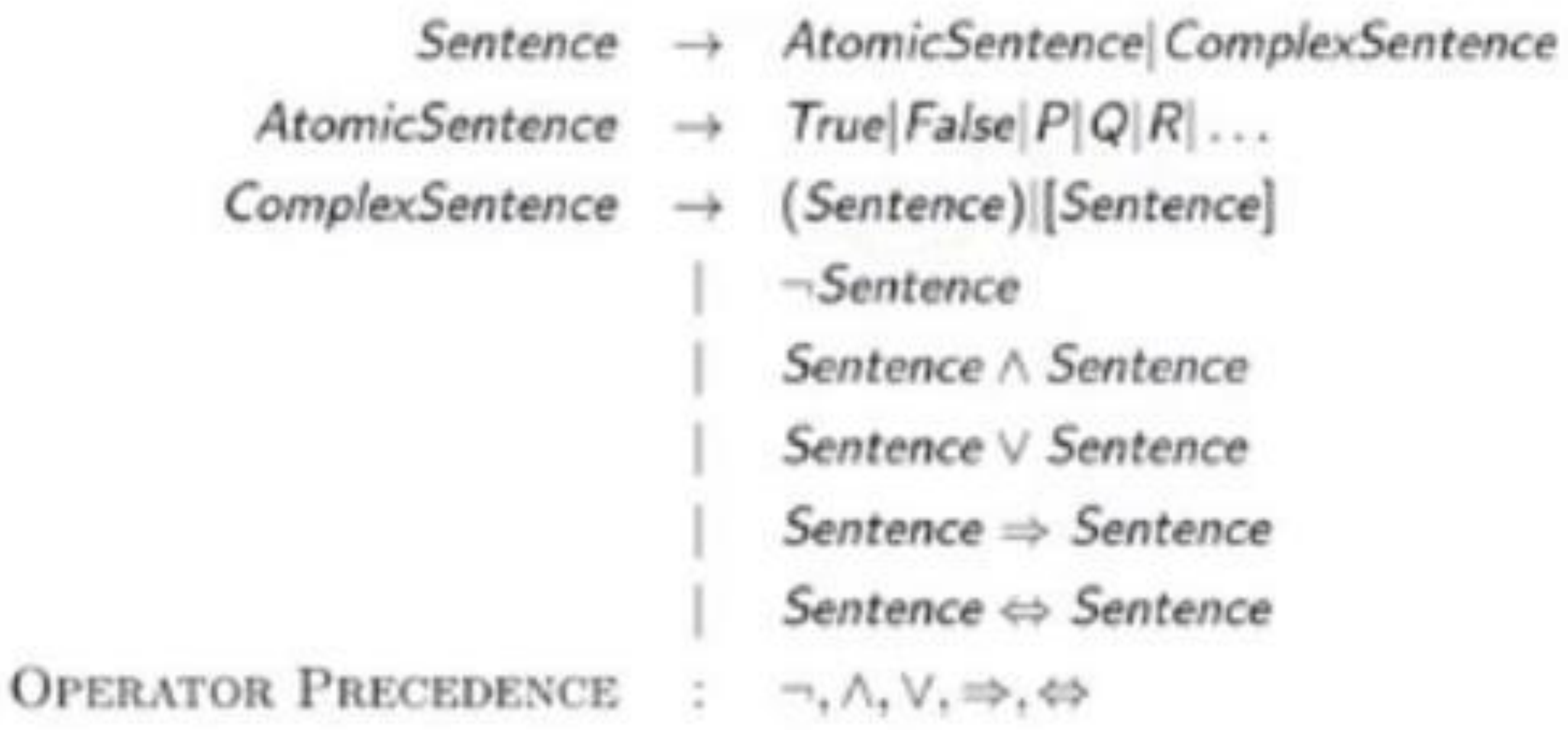


P	Q	♥P	P☺Q	P◎Q	P✸Q	P☁Q
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

♥	$\neg$
☺	$\wedge$
◎	$\vee$
✹	$\Rightarrow$
☁	$\Leftrightarrow$

Propositional logic (命题逻辑): Syntax

Defined in Backus-Naur form (BNF)



$$
\begin{array}{rcl}
\mathit{Sentence} & \rightarrow & \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} & \rightarrow & \mathit{True} \mid \mathit{False} \mid P \mid Q \mid R \mid \ldots \\
\mathit{ComplexSentence} & \rightarrow & (\,\mathit{Sentence}\,) \mid [\,\mathit{Sentence}\,] \\
 & \mid & \neg\,\mathit{Sentence} \\
 & \mid & \mathit{Sentence} \land \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \lor \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \Rightarrow \mathit{Sentence} \\
 & \mid & \mathit{Sentence} \Leftrightarrow \mathit{Sentence} \\
\textsc{Operator Precedence} & : & \neg, \land, \lor, \Rightarrow, \Leftrightarrow
\end{array}
$$

Propositional logic (命题逻辑): Syntax

Propositional logic is the simplest logic—illustrates basic ideas

The proposition symbols P_1, P_2 etc are sentences

If S is a sentence, $\neg S$ is a sentence (negation)

If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction) 合取

If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction) 析取

If S_1 and S_2 are sentences, $S_1 \;\Rightarrow\; S_2$ is a sentence (implication)

If S_1 and S_2 are sentences, $S_1 \;\Leftrightarrow\; S_2$ is a sentence (biconditional)

Propositional logic: Semantics

Each model specifies true/false for each proposition symbol

E.g.	$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
	~~$true$~~ false	$true$	$false$

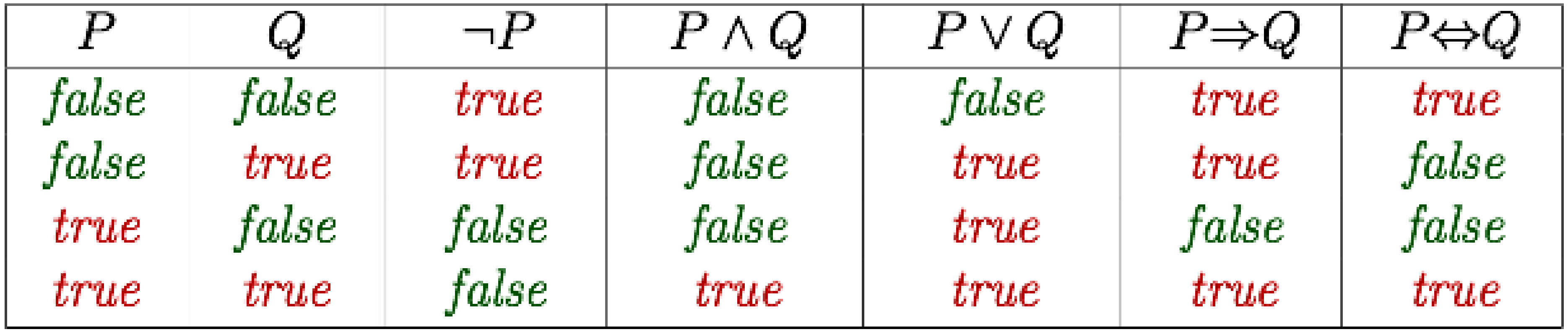
(With these symbols, 8 possible models, can be enumerated automatically.)

Rules for evaluating truth with respect to a model m:

$\neg S$	is true iff	S	is false		
$S_1 \wedge S_2$	is true iff	S_1	is true **and**	S_2	is true
$S_1 \vee S_2$	is true iff	S_1	is true **or**	S_2	is true
$S_1 \Rightarrow S_2$	is true iff	S_1	is false **or**	S_2	is true
i.e.,	is false iff	S_1	is true **and**	S_2	is false
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$	is true **and**	$S_2 \Rightarrow S_1$	is true

Simple recursive process evaluates an arbitrary sentence, e.g.,
$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = true \wedge (false \vee true) = true \wedge true = true$

Truth tables for connectives （真值表）



P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Inference by enumeration 枚举

$KB \models \alpha$ 是否成立

Depth-first enumeration of all models is sound and complete

n [illegible]

2^n 个数

```
function TT-Entails?($KB, \alpha$) returns true or false
    inputs: $KB$, the knowledge base, a sentence in propositional logic
            $\alpha$, the query, a sentence in propositional logic

    symbols ← a list of the proposition symbols in $KB$ and $\alpha$
    return TT-Check-All($KB, \alpha$, symbols, [])
```

```
function TT-Check-All($KB, \alpha$, symbols, model) returns true or false
    if Empty?(symbols) then
        if PL-True?($KB$, model) then return PL-True?($\alpha$, model)
        else return true
    else do
        $P$ ← First(symbols); rest ← Rest(symbols)
        return TT-Check-All($KB, \alpha$, rest, Extend($P$, true, model)) and
                   TT-Check-All($KB, \alpha$, rest, Extend($P$, false, model))
```

$O(2^n)$ for n symbols; problem is **co-NP-complete**

Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

$$\neg P_{1,1}$$
$$\neg B_{1,1}$$
$$B_{2,1}$$

"Pits cause breezes in adjacent squares"

$$B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$$
$$B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$$

"A square is breezy **if and only if** there is an adjacent pit"

Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Enumerate rows (different assignments to symbols),
if KB is true in row, check that α is too

Logical equivalence

语义

Two sentences are logically equivalent iff true in same models:

$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

$(\alpha \wedge \beta)$	$\equiv$	$(\beta \wedge \alpha)$	commutativity of $\wedge$
$(\alpha \vee \beta)$	$\equiv$	$(\beta \vee \alpha)$	commutativity of $\vee$
$((\alpha \wedge \beta) \wedge \gamma)$	$\equiv$	$(\alpha \wedge (\beta \wedge \gamma))$	associativity of $\wedge$
$((\alpha \vee \beta) \vee \gamma)$	$\equiv$	$(\alpha \vee (\beta \vee \gamma))$	associativity of $\vee$
$\neg(\neg\alpha)$	$\equiv$	α	double-negation elimination
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\alpha \vee \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	biconditional elimination
$\neg(\alpha \wedge \beta)$	$\equiv$	$(\neg\alpha \vee \neg\beta)$	De Morgan
$\neg(\alpha \vee \beta)$	$\equiv$	$(\neg\alpha \wedge \neg\beta)$	De Morgan
$(\alpha \wedge (\beta \vee \gamma))$	$\equiv$	$((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distributivity of $\wedge$ over $\vee$
$(\alpha \vee (\beta \wedge \gamma))$	$\equiv$	$((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	distributivity of $\vee$ over $\wedge$

作业

Entailment与implication的区别

Entailment: 逻辑上的概念，刻画两组sentence之间的关系

Implication: proposition之间的一种运算子，使用真值表刻画其语义

Validity and satisfiability

必考

A sentence is valid if it is true in **all** models,

e.g., $True$, $A \lor \neg A$, $A \Rightarrow A$, $(A \land (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the Deduction Theorem:

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid 必记考

A sentence is satisfiable if it is true in **some** model

e.g., $A \lor B$, C

A sentence is unsatisfiable if it is true in **no** models

e.g., $A \land \neg A$

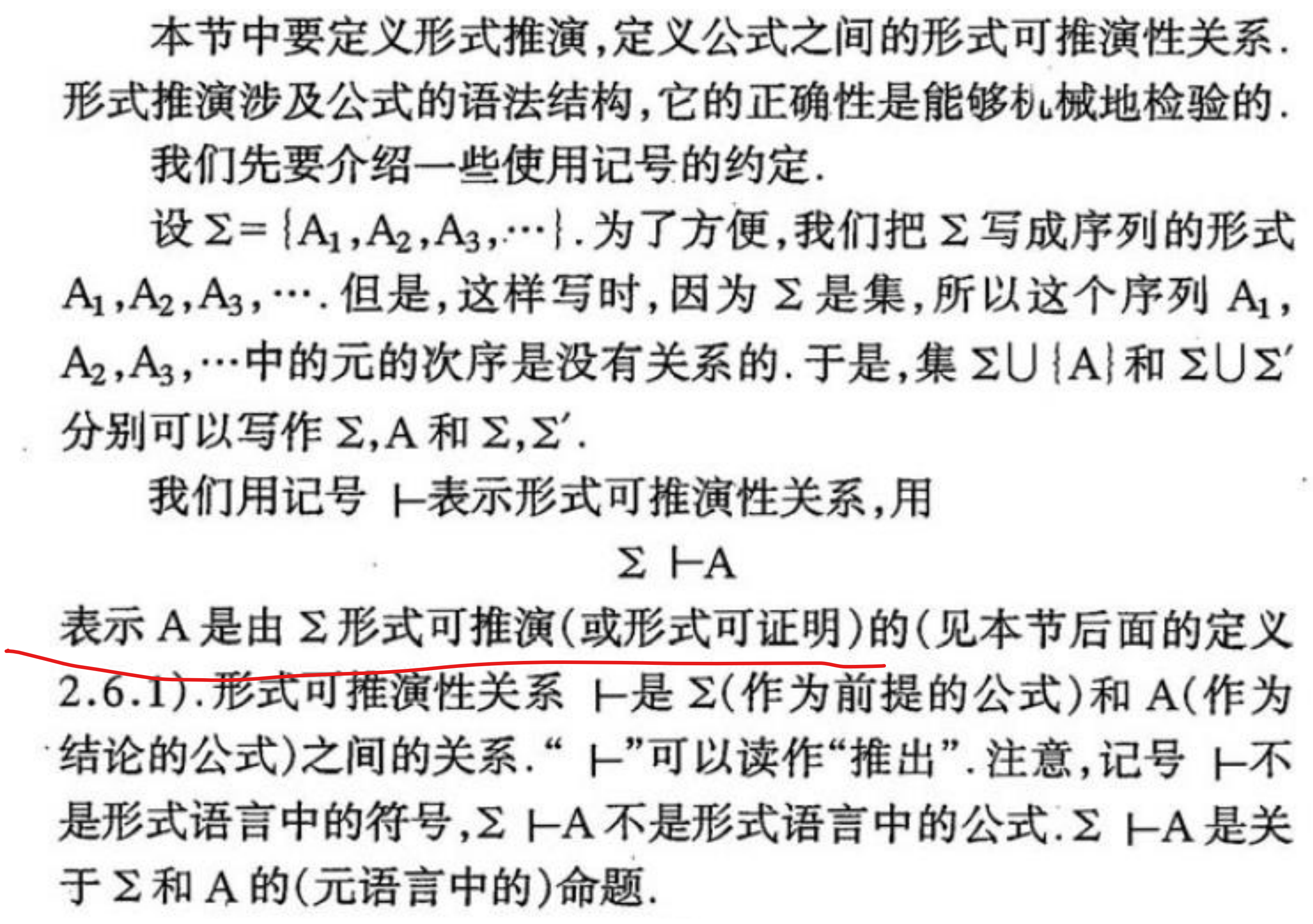
Satisfiability is connected to inference via the following:

$KB \models \alpha$ if and only if $(KB \land \neg\alpha)$ is unsatisfiable

i.e., prove α by *reductio ad absurdum*

形式推演 Deduction

形式推演 Deduction



本节中要定义形式推演，定义公式之间的形式可推演性关系．形式推演涉及公式的语法结构，它的正确性是能够机械地检验的．

我们先要介绍一些使用记号的约定．

设 $\Sigma = \{A_1, A_2, A_3, \cdots\}$．为了方便，我们把 Σ 写成序列的形式 $A_1, A_2, A_3, \cdots$．但是，这样写时，因为 Σ 是集，所以这个序列 $A_1, A_2, A_3, \cdots$ 中的元的次序是没有关系的．于是，集 $\Sigma \cup \{A\}$ 和 $\Sigma \cup \Sigma'$ 分别可以写作 Σ, A 和 Σ, Σ'．

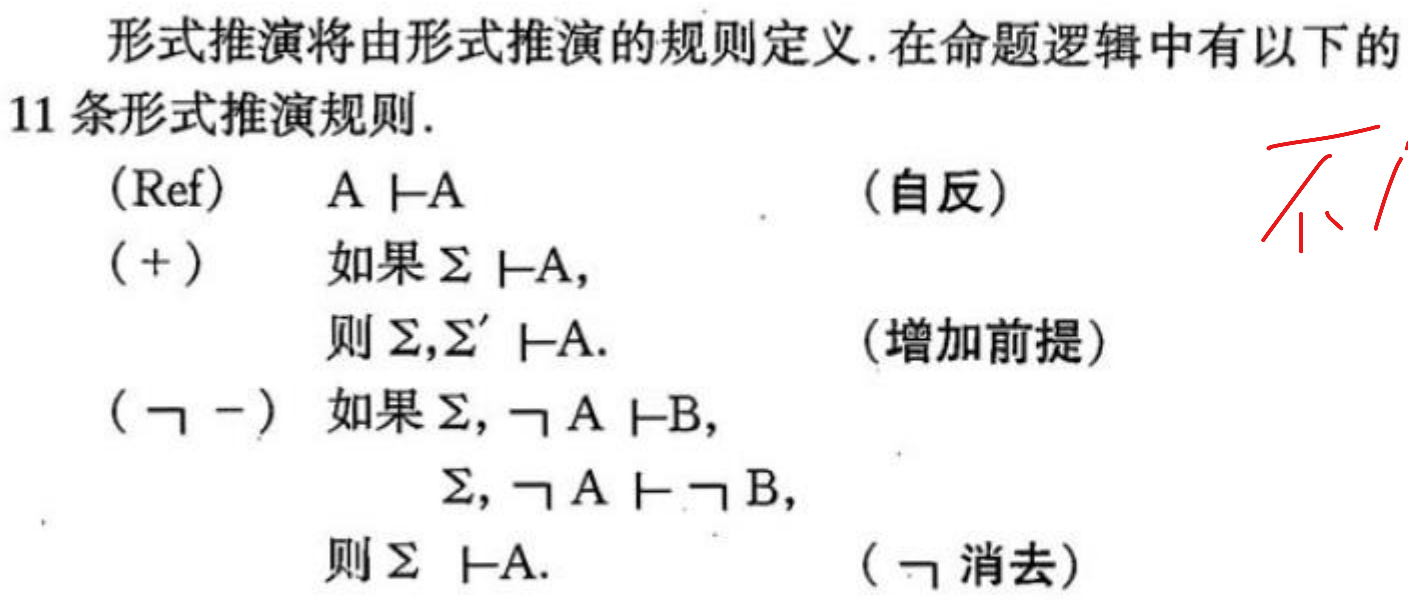
我们用记号 $\vdash$ 表示形式可推演性关系，用

$$\Sigma \vdash A$$

表示 A 是由 Σ 形式可推演（或形式可证明）的（见本节后面的定义 2.6.1）．形式可推演性关系 $\vdash$ 是 Σ（作为前提的公式）和 A（作为结论的公式）之间的关系．“$\vdash$”可以读作“推出”．注意，记号 $\vdash$ 不是形式语言中的符号，$\Sigma \vdash A$ 不是形式语言中的公式．$\Sigma \vdash A$ 是关于 Σ 和 A 的（元语言中的）命题．

《面向计算机科学的数理逻辑》49页

形式推演的11条规则（某一种推演系统）



形式推演将由形式推演的规则定义. 在命题逻辑中有以下的 11 条形式推演规则.

(Ref)　$A \vdash A$　(自反)

(+)　如果 $\Sigma \vdash A$,
则 $\Sigma, \Sigma' \vdash A$.　(增加前提)

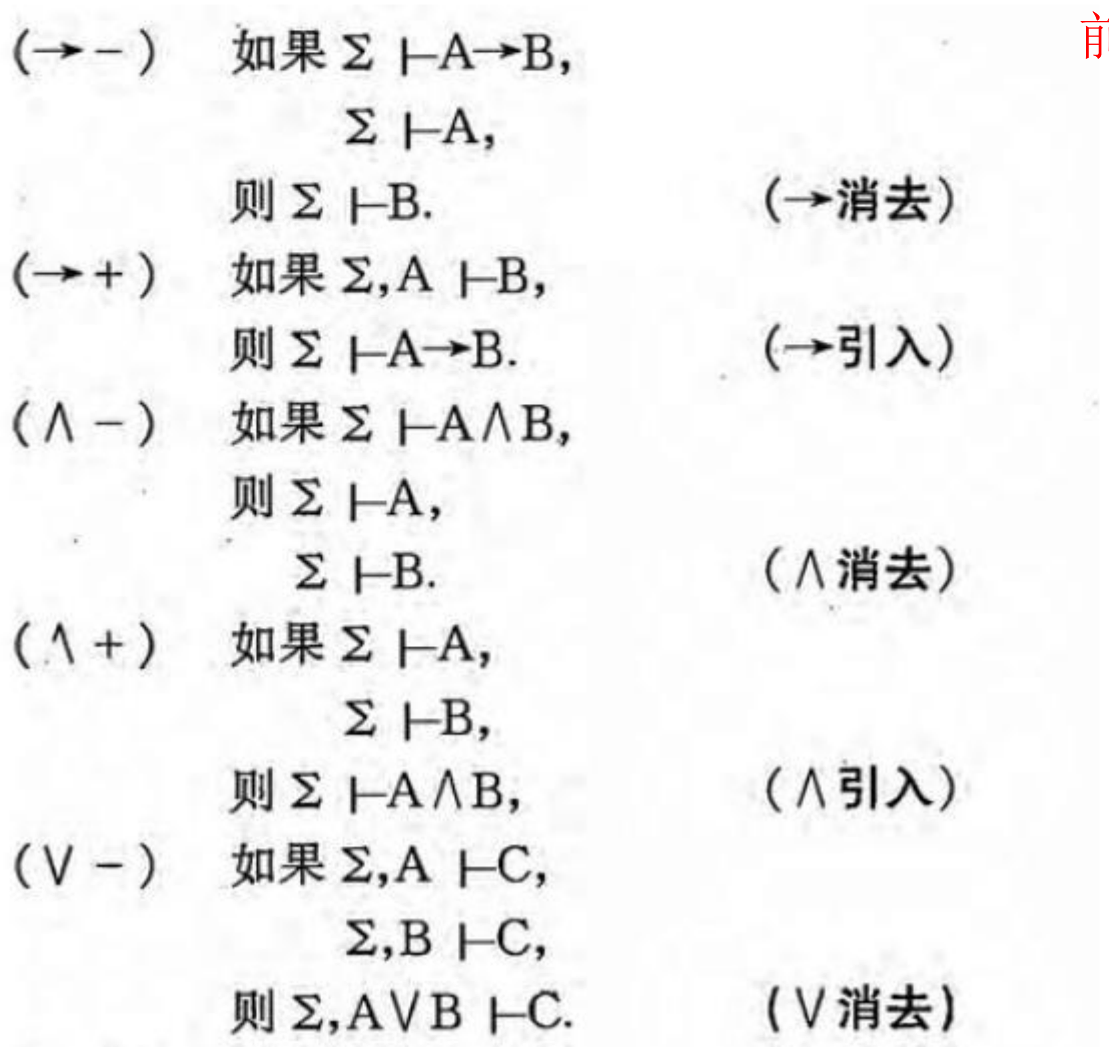
($\neg$ −)　如果 $\Sigma, \neg A \vdash B$,
$\Sigma, \neg A \vdash \neg B$,
则 $\Sigma \vdash A$.　($\neg$ 消去)

不用背

《面向计算机科学的数理逻辑》49页

形式推演的11条规则

符号提示：
这里的→ 就是
前面定义的⇒



(→ −)　如果 $\Sigma \vdash A \to B$,
　　　　$\Sigma \vdash A$,
　　　则 $\Sigma \vdash B$.　　(→消去)

(→ +)　如果 $\Sigma, A \vdash B$,
　　　则 $\Sigma \vdash A \to B$.　　(→引入)

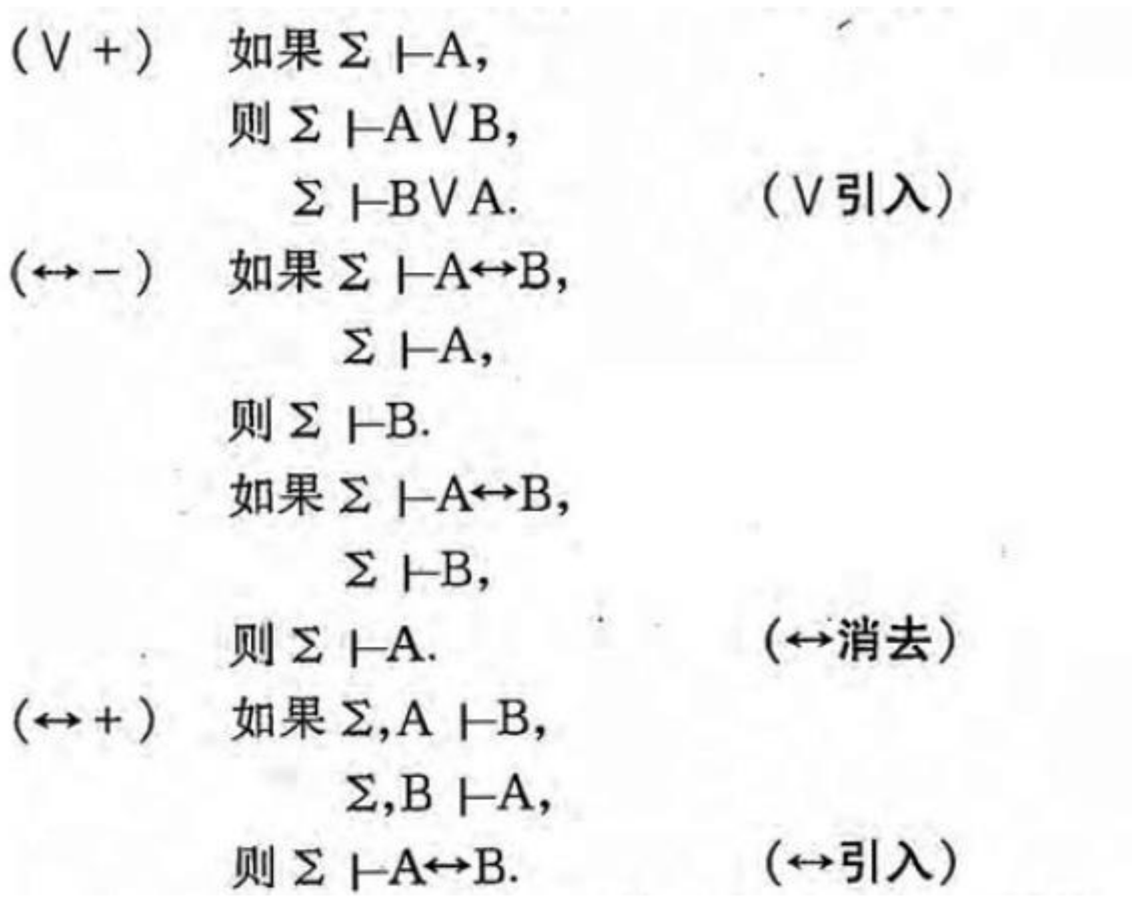
(∧ −)　如果 $\Sigma \vdash A \wedge B$,
　　　则 $\Sigma \vdash A$,
　　　　$\Sigma \vdash B$.　　(∧消去)

(∧ +)　如果 $\Sigma \vdash A$,
　　　　$\Sigma \vdash B$,
　　　则 $\Sigma \vdash A \wedge B$,　　(∧引入)

(∨ −)　如果 $\Sigma, A \vdash C$,
　　　　$\Sigma, B \vdash C$,
　　　则 $\Sigma, A \vee B \vdash C$.　　(∨消去)

形式推演的11条规则

符号提示：
这里的↔ 就是
前面定义的⇔

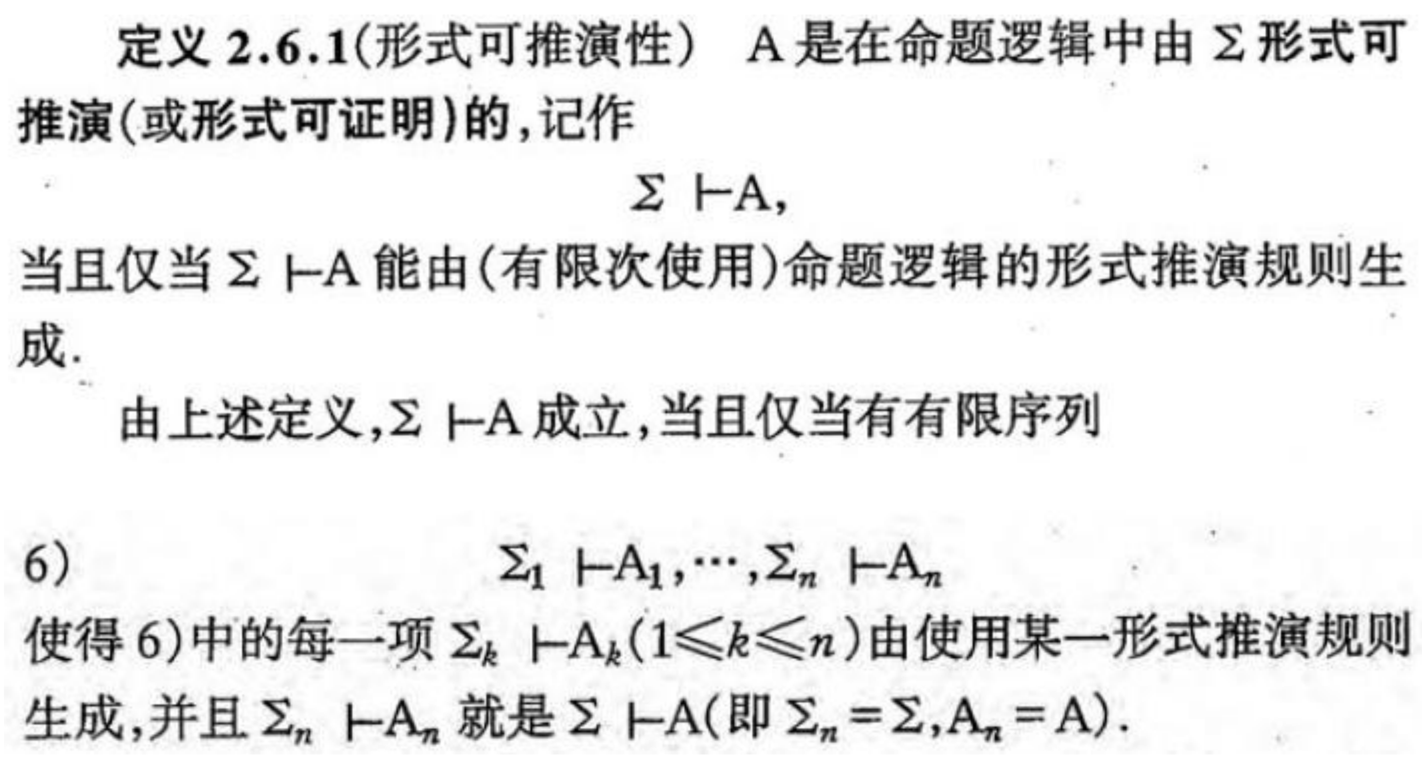


($\vee +$) 如果 $\Sigma \vdash A$,
则 $\Sigma \vdash A \vee B$,
$\Sigma \vdash B \vee A$. （$\vee$引入）

($\leftrightarrow -$) 如果 $\Sigma \vdash A \leftrightarrow B$,
$\Sigma \vdash A$,
则 $\Sigma \vdash B$.
如果 $\Sigma \vdash A \leftrightarrow B$,
$\Sigma \vdash B$,
则 $\Sigma \vdash A$. （$\leftrightarrow$消去）

($\leftrightarrow +$) 如果 $\Sigma, A \vdash B$,
$\Sigma, B \vdash A$,
则 $\Sigma \vdash A \leftrightarrow B$. （$\leftrightarrow$引入）

形式推演



定义 2.6.1(形式可推演性)　A 是在命题逻辑中由 Σ 形式可推演(或形式可证明)的，记作

$$\Sigma \vdash A,$$

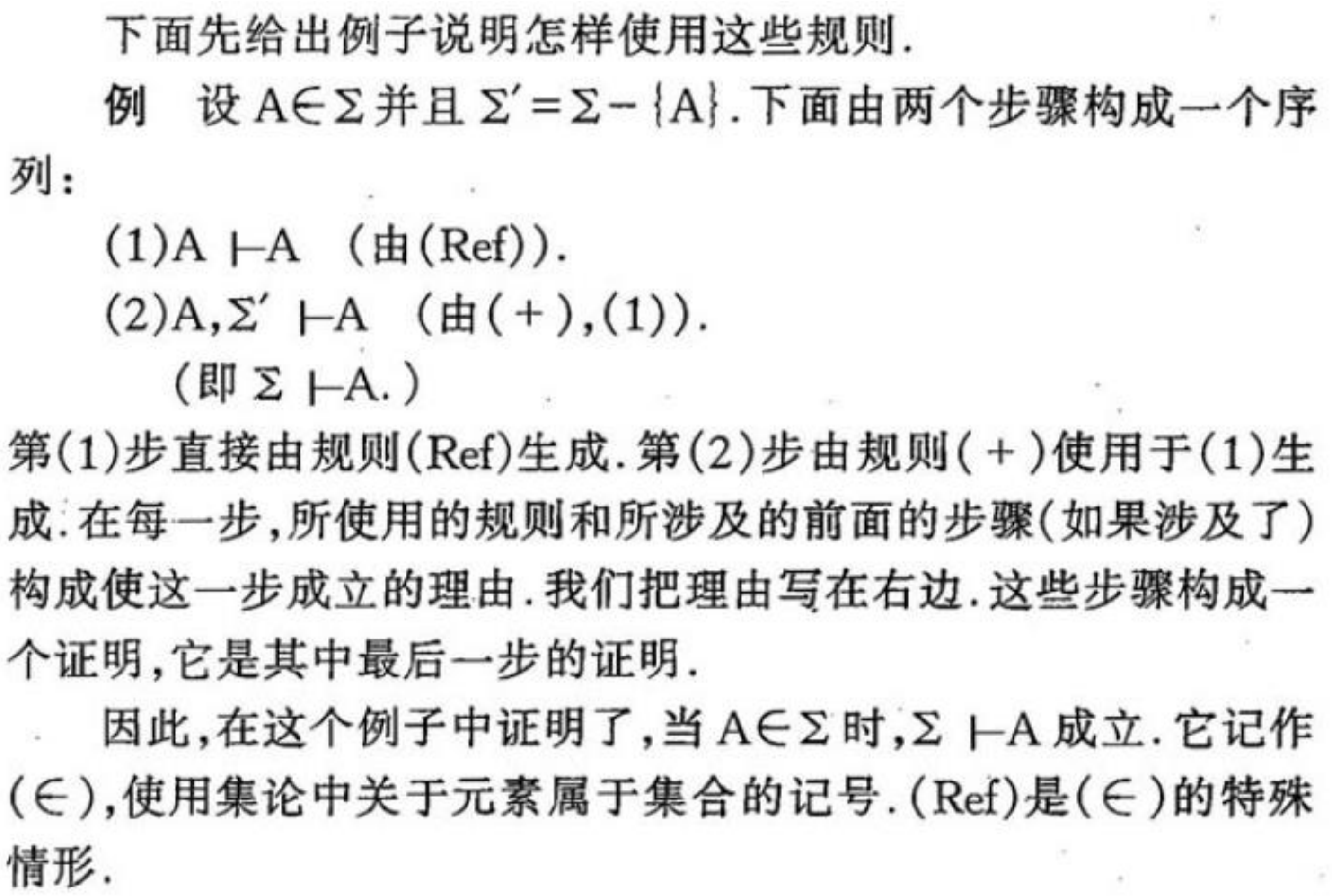
当且仅当 $\Sigma \vdash A$ 能由(有限次使用)命题逻辑的形式推演规则生成.

由上述定义，$\Sigma \vdash A$ 成立，当且仅当有有限序列

6)　$$\Sigma_1 \vdash A_1, \cdots, \Sigma_n \vdash A_n$$

使得 6)中的每一项 $\Sigma_k \vdash A_k (1 \leqslant k \leqslant n)$ 由使用某一形式推演规则生成，并且 $\Sigma_n \vdash A_n$ 就是 $\Sigma \vdash A$(即 $\Sigma_n = \Sigma, A_n = A$).

形式推演：例子



下面先给出例子说明怎样使用这些规则．

例 设 $A \in \Sigma$ 并且 $\Sigma' = \Sigma - \{A\}$．下面由两个步骤构成一个序列：

(1) $A \vdash A$ （由(Ref)）．

(2) $A, \Sigma' \vdash A$ （由(+)，(1)）．

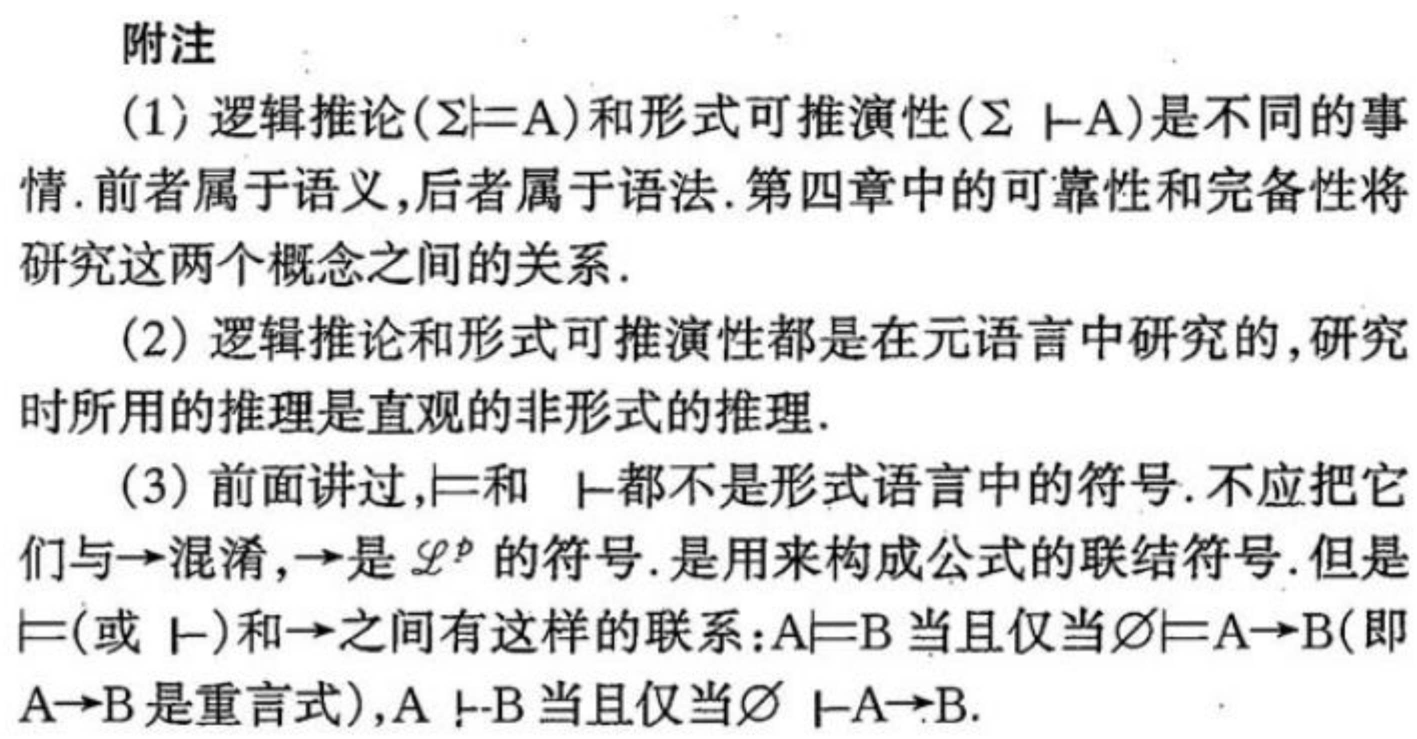
（即 $\Sigma \vdash A$．）

第(1)步直接由规则(Ref)生成．第(2)步由规则(+)使用于(1)生成．在每一步，所使用的规则和所涉及的前面的步骤(如果涉及了)构成使这一步成立的理由．我们把理由写在右边．这些步骤构成一个证明，它是其中最后一步的证明．

因此，在这个例子中证明了，当 $A \in \Sigma$ 时，$\Sigma \vdash A$ 成立．它记作(∈)，使用集论中关于元素属于集合的记号．(Ref)是(∈)的特殊情形．

《面向计算机科学的数理逻辑》51页

逻辑推导 vs. 形式推演



附注

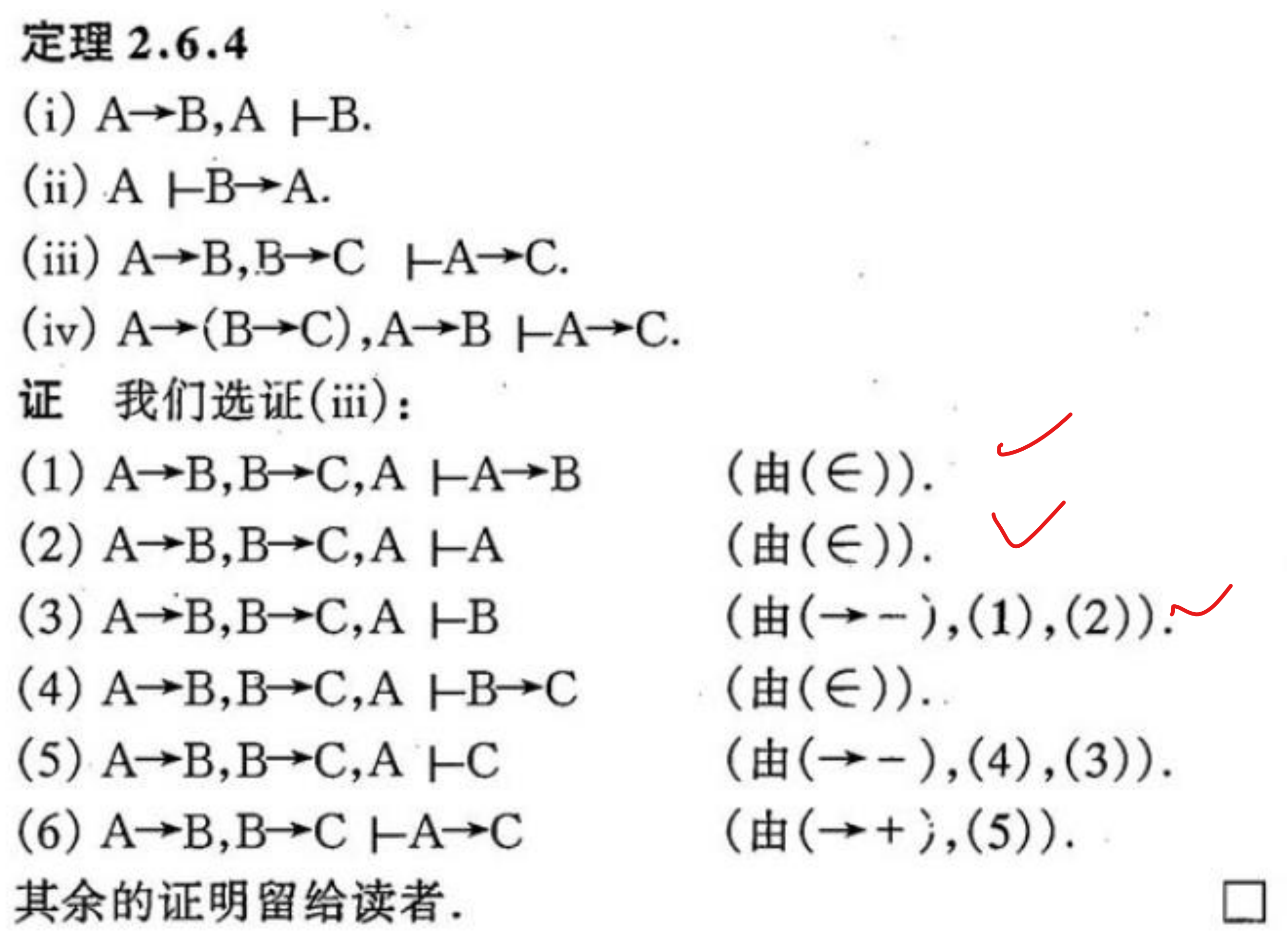
(1) 逻辑推论($\Sigma \models A$)和形式可推演性($\Sigma \vdash A$)是不同的事情．前者属于语义，后者属于语法．第四章中的可靠性和完备性将研究这两个概念之间的关系．

(2) 逻辑推论和形式可推演性都是在元语言中研究的，研究时所用的推理是直观的非形式的推理．

(3) 前面讲过，$\models$和 $\vdash$都不是形式语言中的符号．不应把它们与$\rightarrow$混淆，$\rightarrow$是 $\mathscr{L}^p$ 的符号．是用来构成公式的联结符号．但是$\models$(或 $\vdash$)和$\rightarrow$之间有这样的联系：$A \models B$ 当且仅当$\varnothing \models A \rightarrow B$(即$A \rightarrow B$是重言式)，$A \vdash B$ 当且仅当$\varnothing \vdash A \rightarrow B$.

《面向计算机科学的数理逻辑》55页

形式推演：例子



定理 2.6.4

(i) $A \to B, A \vdash B$.

(ii) $A \vdash B \to A$.

(iii) $A \to B, B \to C \vdash A \to C$.

(iv) $A \to (B \to C), A \to B \vdash A \to C$.

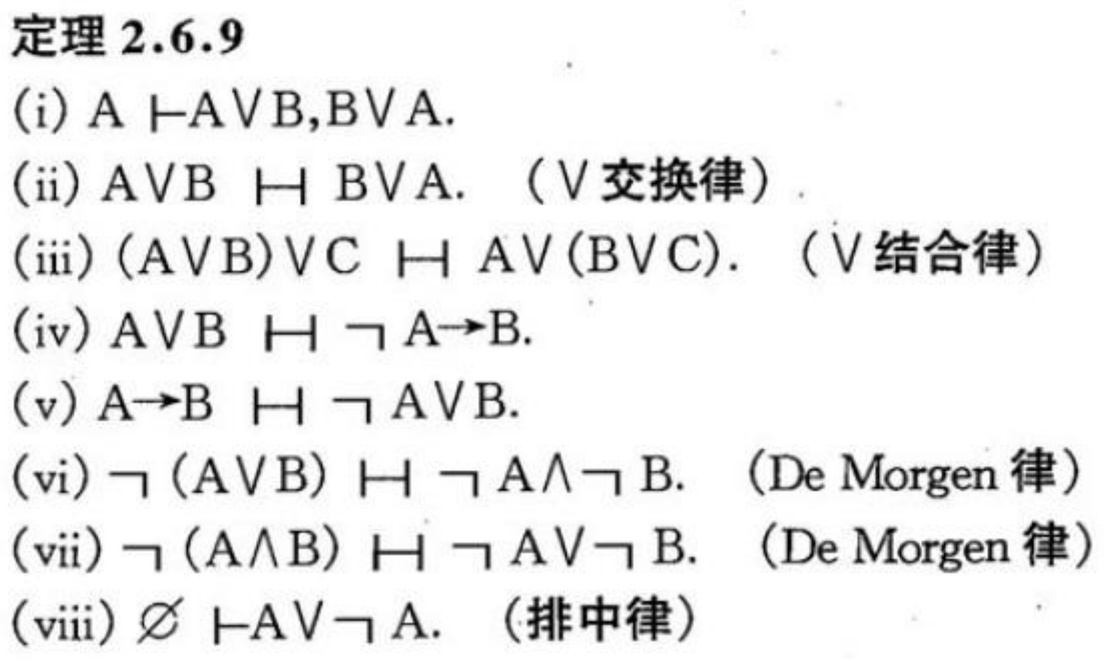
证 我们选证(iii)：

(1) $A \to B, B \to C, A \vdash A \to B$	(由($\in$)). ✓
(2) $A \to B, B \to C, A \vdash A$	(由($\in$)). ✓
(3) $A \to B, B \to C, A \vdash B$	(由($\to -$),(1),(2)). ✓
(4) $A \to B, B \to C, A \vdash B \to C$	(由($\in$)).
(5) $A \to B, B \to C, A \vdash C$	(由($\to -$),(4),(3)).
(6) $A \to B, B \to C \vdash A \to C$	(由($\to +$),(5)).

其余的证明留给读者. □

Homework：其余题目

形式推演：例子



定理 2.6.9

(i) $A \vdash A \vee B, B \vee A$.

(ii) $A \vee B \vdash\dashv B \vee A$.　（$\vee$ **交换律**）

(iii) $(A \vee B) \vee C \vdash\dashv A \vee (B \vee C)$.　（$\vee$ **结合律**）

(iv) $A \vee B \vdash\dashv \neg A \rightarrow B$.

(v) $A \rightarrow B \vdash\dashv \neg A \vee B$.

(vi) $\neg (A \vee B) \vdash\dashv \neg A \wedge \neg B$.　(De Morgen **律**)

(vii) $\neg (A \wedge B) \vdash\dashv \neg A \vee \neg B$.　(De Morgen **律**)

(viii) $\varnothing \vdash A \vee \neg A$.　（**排中律**）

Homework：证明这些性质

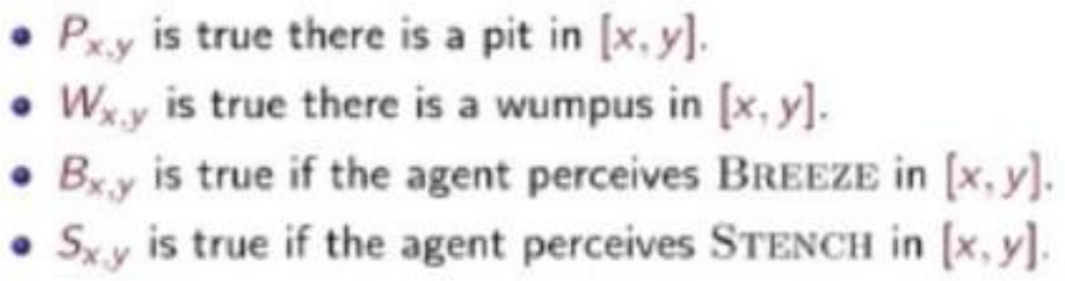
《面向计算机科学的数理逻辑》63页

形式推演：常用的定理

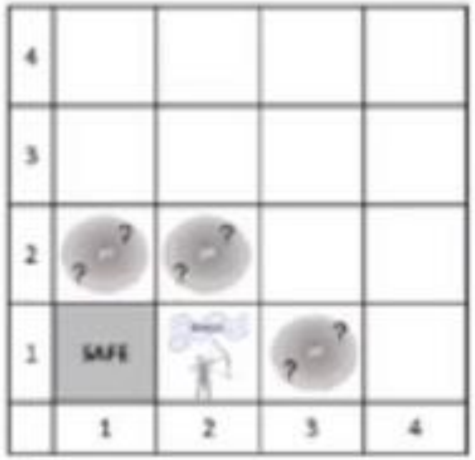
定理2.6.3, 2.6.4, 2.6.5, 2.6.6, 2.6.7, 2.6.8, 2.6.9

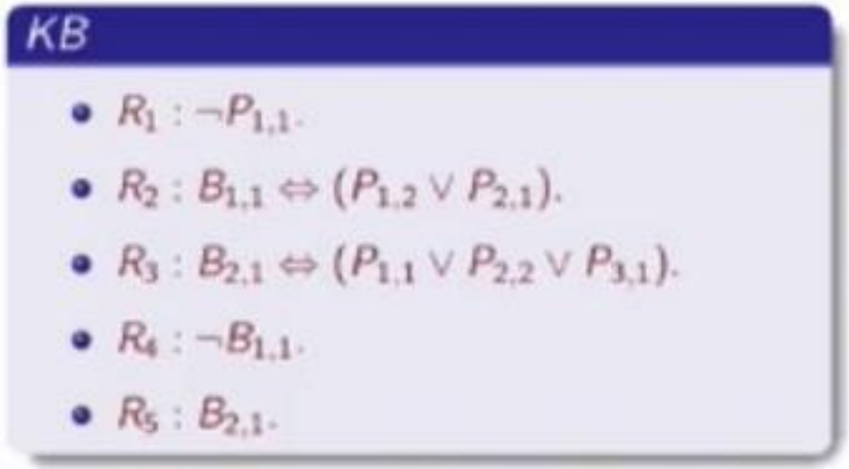
可以用基本的11条法则证明，它们则可以在其它证明中使用

形式推演：Wumpus



- $P_{x,y}$ is true there is a pit in $[x,y]$.
- $W_{x,y}$ is true there is a wumpus in $[x,y]$.
- $B_{x,y}$ is true if the agent perceives BREEZE in $[x,y]$.
- $S_{x,y}$ is true if the agent perceives STENCH in $[x,y]$.



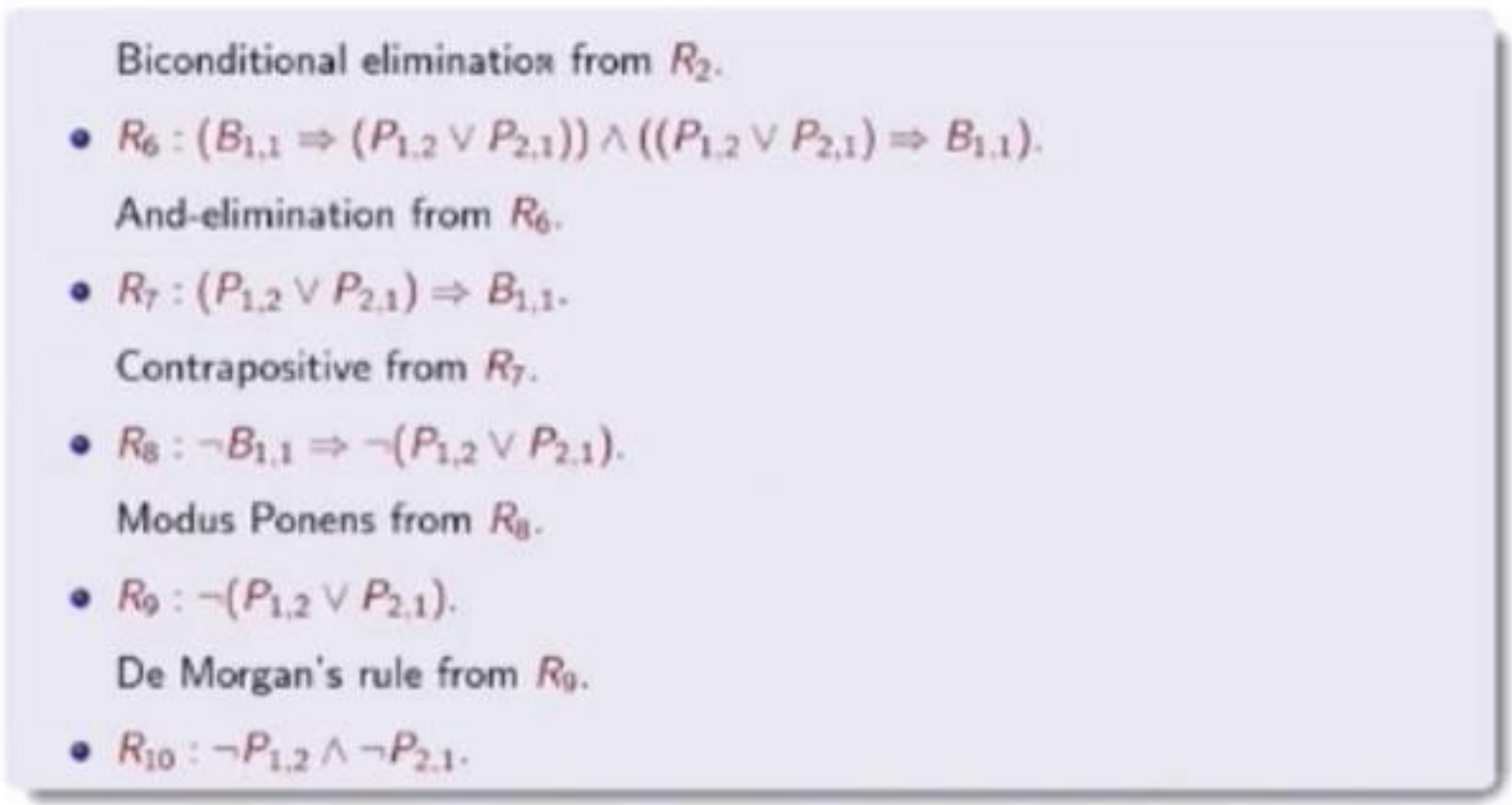


KB

- $R_1 : \neg P_{1,1}$.
- $R_2 : B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$.
- $R_3 : B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$.
- $R_4 : \neg B_{1,1}$.
- $R_5 : B_{2,1}$.

证明： $\mathrm{KB} \vdash \neg P_{1,2} \wedge \neg P_{2,1}$

形式推演：Wumpus



Biconditional elimination from R_2.

- $R_6 : (B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$.

And-elimination from R_6.

- $R_7 : (P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1}$.

Contrapositive from R_7.

- $R_8 : \neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1})$.

Modus Ponens from R_8.

- $R_9 : \neg(P_{1,2} \vee P_{2,1})$.

De Morgan's rule from R_9.

- $R_{10} : \neg P_{1,2} \wedge \neg P_{2,1}$.

证明： $\mathrm{KB} \vdash \neg P_{1,2} \wedge \neg P_{2,1}$

证明过程并不是一个机械化的方法

Summary

逻辑系统

- Syntax：formal structure of sentences
- Semantics: truth of sentences wrt models;
  Entailment: necessary truth of one sentence given another
- Deduction: formal deduction based on deduction rules

Inference

$KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i

Consequences of KB are a haystack; α is a needle.
Entailment = needle in haystack; inference = finding it

Soundness: i is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

Completeness: i is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

That is, the procedure will answer any question whose answer follows from what is known by the KB.

哥德尔不完全 定理：在一个大的范围内（证明法和问题与正整数存在一一对应关系），不存在既sound又complete的inference过程