知识表示与推理 I：逻辑智能体

University of Science and Technology of China

May 8, 2013

Outline

Outline

逻辑智能体

- 逻辑智能体（Logical Agents）：基于知识的智能体（Knowledge-Based Agents）
  - 知识库（Knowledge Base, KB）：关于世界（外部环境，智能体自身状态，行动能力等）知识的集合。一般每条知识对应一个语句（sentence），通过特定的知识表示语言（knowledge representation language）来表达。例如，**雪是白的**; **鸟通常会飞**; **杯子在桌子上**; **如果按红色按钮，则会发出警报**。
  - 推理机（Inference System）：根据 KB 推理出相应（隐含）知识。例如，由 $KB = \{$ **下雨**, **如果下雨则地湿** $\}$ 可以推理得出知识 **地湿**。
- 逻辑智能体操作过程：
  1. TELL KB 新观察或新知识；
  2. ASK KB 下一步采取什么行动；
  3. 执行行动，并 TELL KB.

逻辑智能体层次

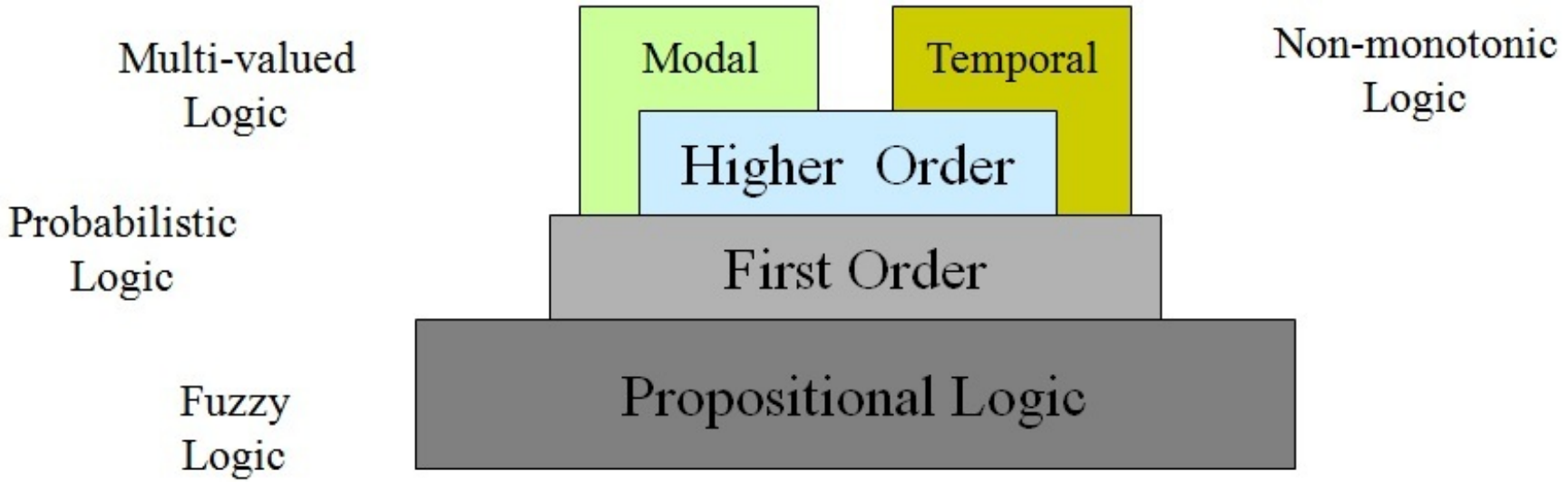
- 知识层次（Knowledge Level）：
  - 最抽象的层次，智能体所拥有的知识。
  - 例如，“可佳”机器人知道，通过操作微波炉可以加热食物，也知道按下微波炉的哪个按钮会有什么后果。
- **逻辑层次（Logical Level）：**
  - 语句层次，知识被编码为具体的语句。
  - 例如，在“可佳”机器人中，$process(micro, food) \Rightarrow het(food)$，$press(button, 2) \wedge startbutton(micro, button) \wedge in(micro, food) \Rightarrow process(micro, food)$.
- 执行层（Implementation Level）：
  - 具体的执行层次，涉及具体的算法和数据结构。
  - 例如，“可佳”中通过逻辑程序实现，
    $het(food, t) \leftarrow process(micro, food, t)$.
    $process(micro, food, t+2) \leftarrow$
    $press(button, 2, t), startbutton(micro, button), in(micro, food)$.

知识表示与推理

- 知识表示与推理（Knowledge Representation and Reasoning, KR）
  - The basic assumption underlying KR (and much of AI) is that thinking can be usefully understood as mechanical operations over symbolic representations. This hypothesis is, in fact, quite old, much older than computers, and seems to have originated with the philosopher Leibniz ... Just as there is a calculus of arithmetic, where numerical expressions are formally manipulated in a value-preserving way, so might there be a calculus of thought, where propositional expressions could be formally manipulated in a truth-preserving way. (Levesque, 1986)
  - KR 假设：（人工）智能可以通过符号推理形式化刻画。
- KR 的目的是将知识表达为计算机可处理（computer-tractable）的形式，使智能体可以使用。
- KR 语言包括：
  - 语法（syntax）：语言中合法的语句。例如，一阶逻辑中公式。
  - 语义（semantics）：语句在世界中的解释。例如，$het(food)$ 表示 *food* 处在被加热后的状态。

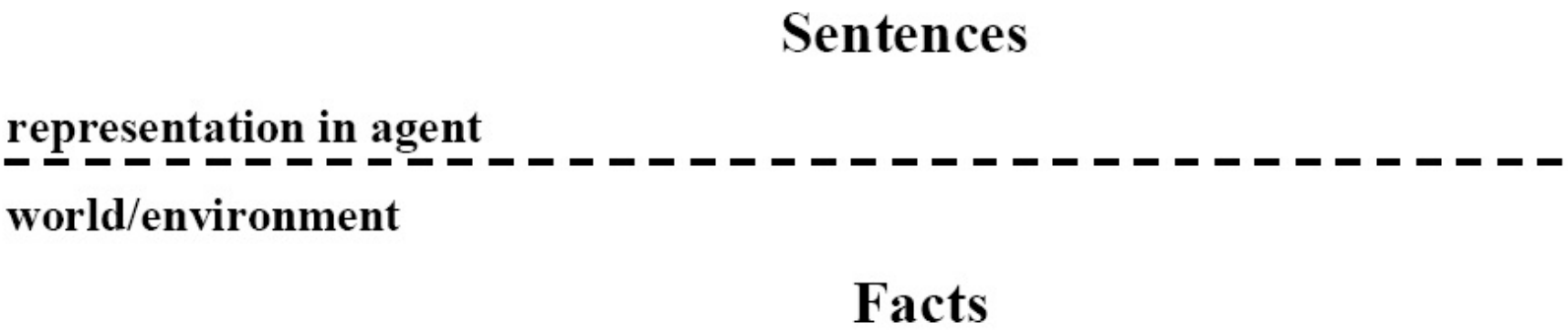
逻辑

- 逻辑（学科）：通过形式化语言刻画有效的推理（valid inferences）。
- 逻辑指刻画知识的形式化语言，并且可以进行推理。
- 逻辑作为一种 KR 语言：



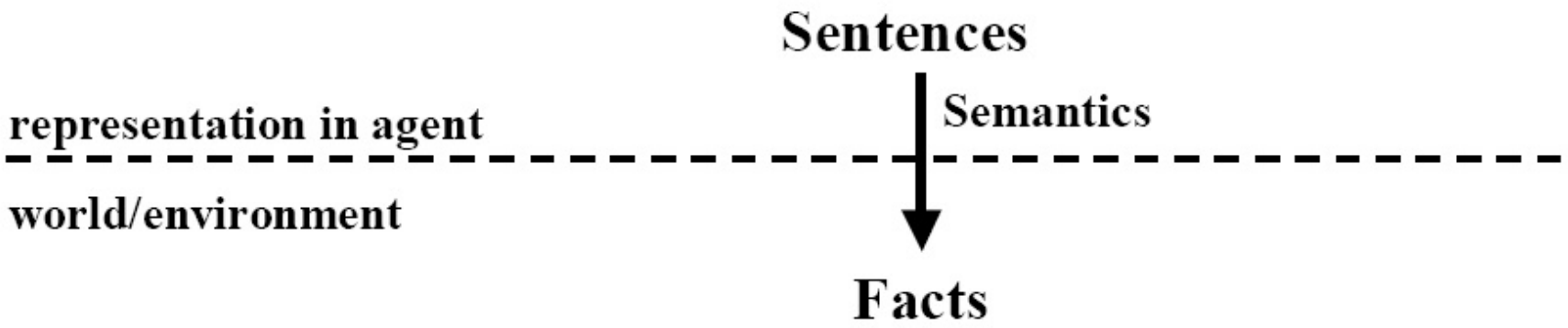
逻辑

- 智能体在 KR 中用语句（sentences）表达世界/环境
- 语句（Sentences）由某种形式化语言来描述
- 事实（Facts）表示世界/环境中真/假的断言



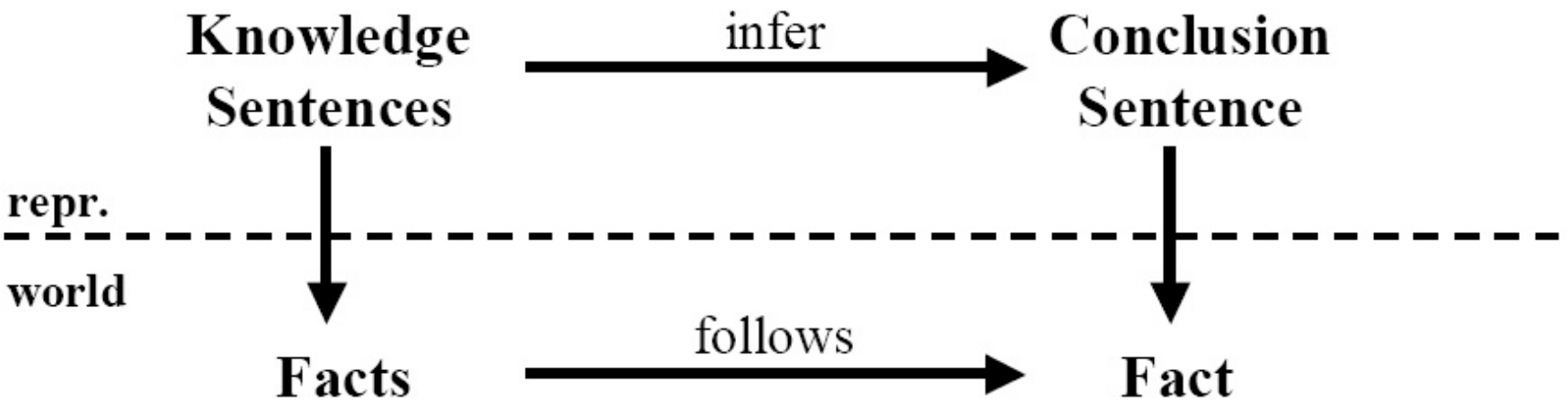
逻辑

- 语句表达世界中的事实
- 语义（semantics）将语句与事实连接起来
- 某语句为真，则它所代表的事实在相应（真实）世界中也为真



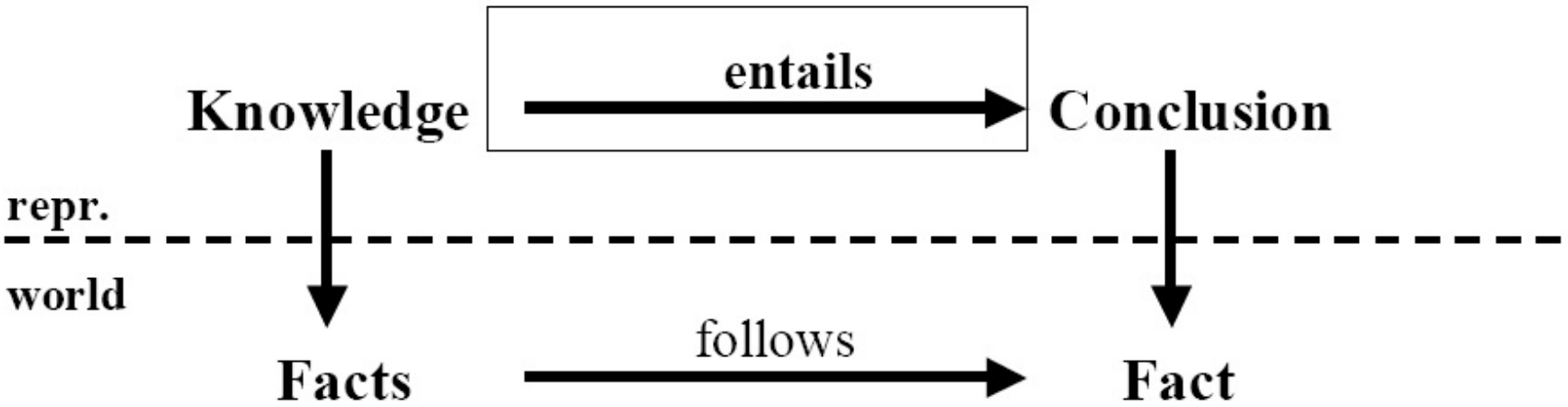
逻辑

- 通过合理的推理机，使得由 KR 推理得到的结论也符合事实
- 逻辑的可靠性（soundness）：推理机推出的语句都符合事实
- 逻辑的完全性（completeness）：凡是为真的事实，都可以由推理机推出



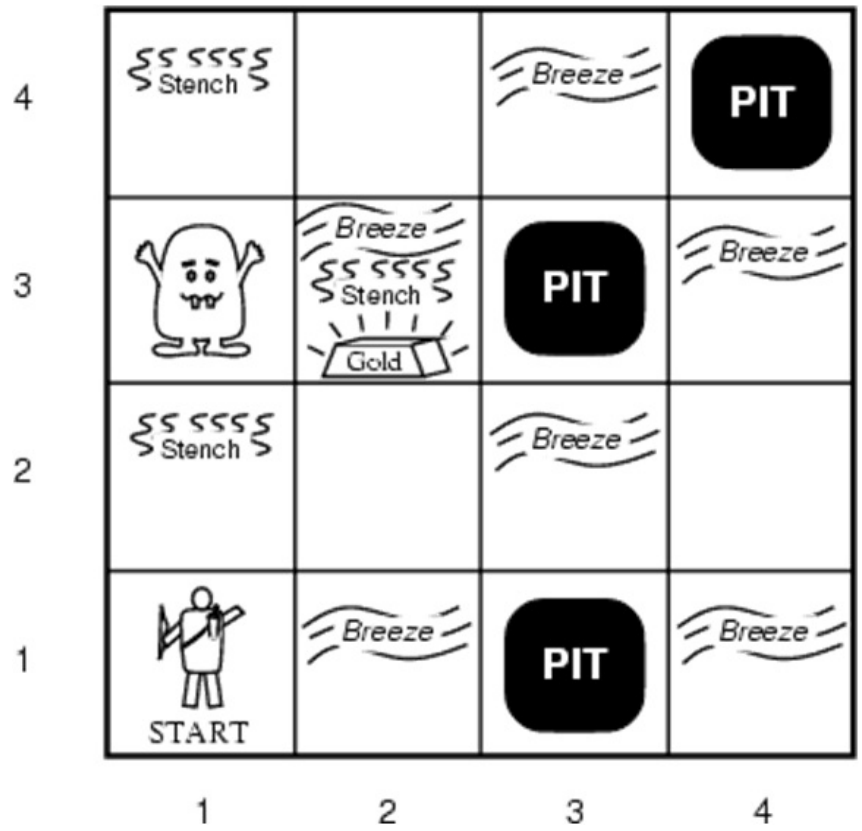
逻辑智能体

- 计算机不知道语义（semantics）/含义
- 只能在完全不知道语义的情况下机械的处理语句
- KR 的根本问题：在设计逻辑语言时，平衡其表达能力（expressiveness）与推理效率（tractability）

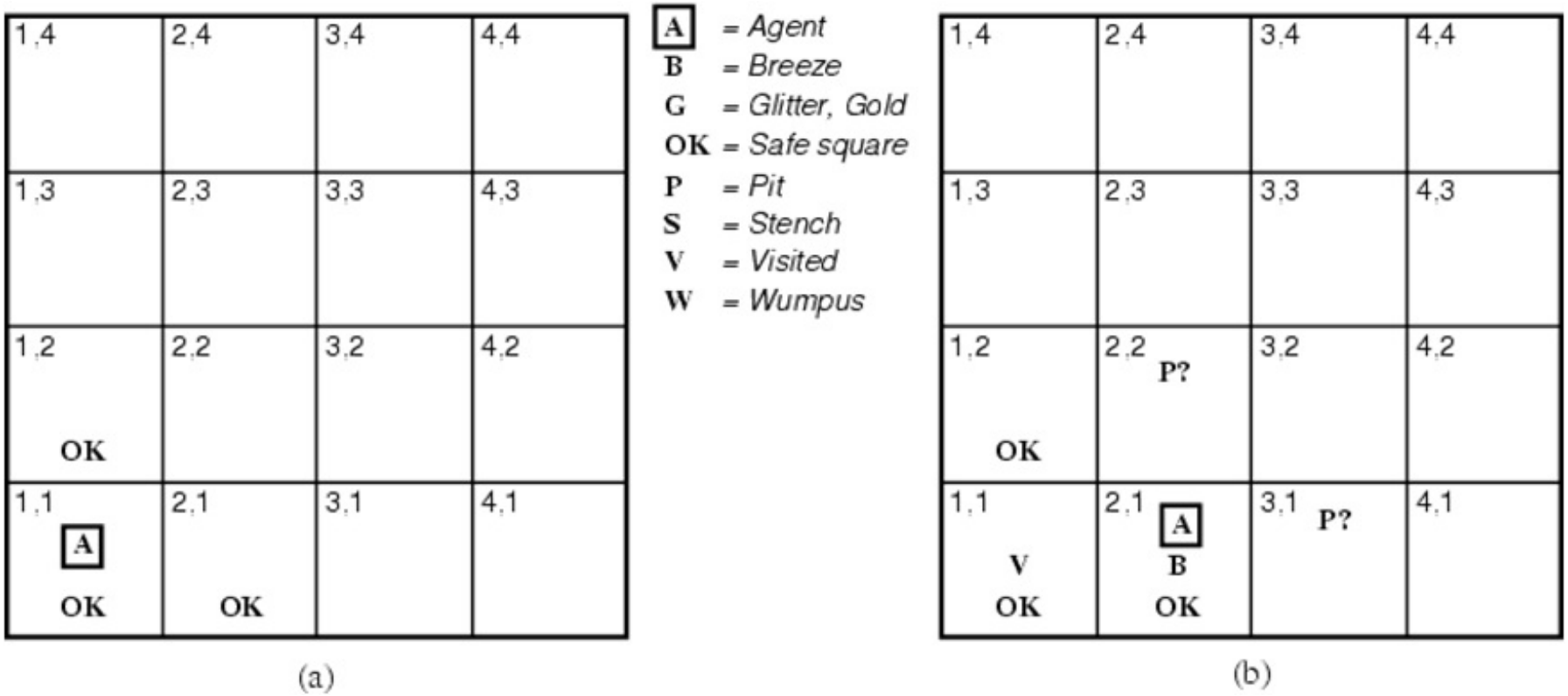


Wumpus 世界

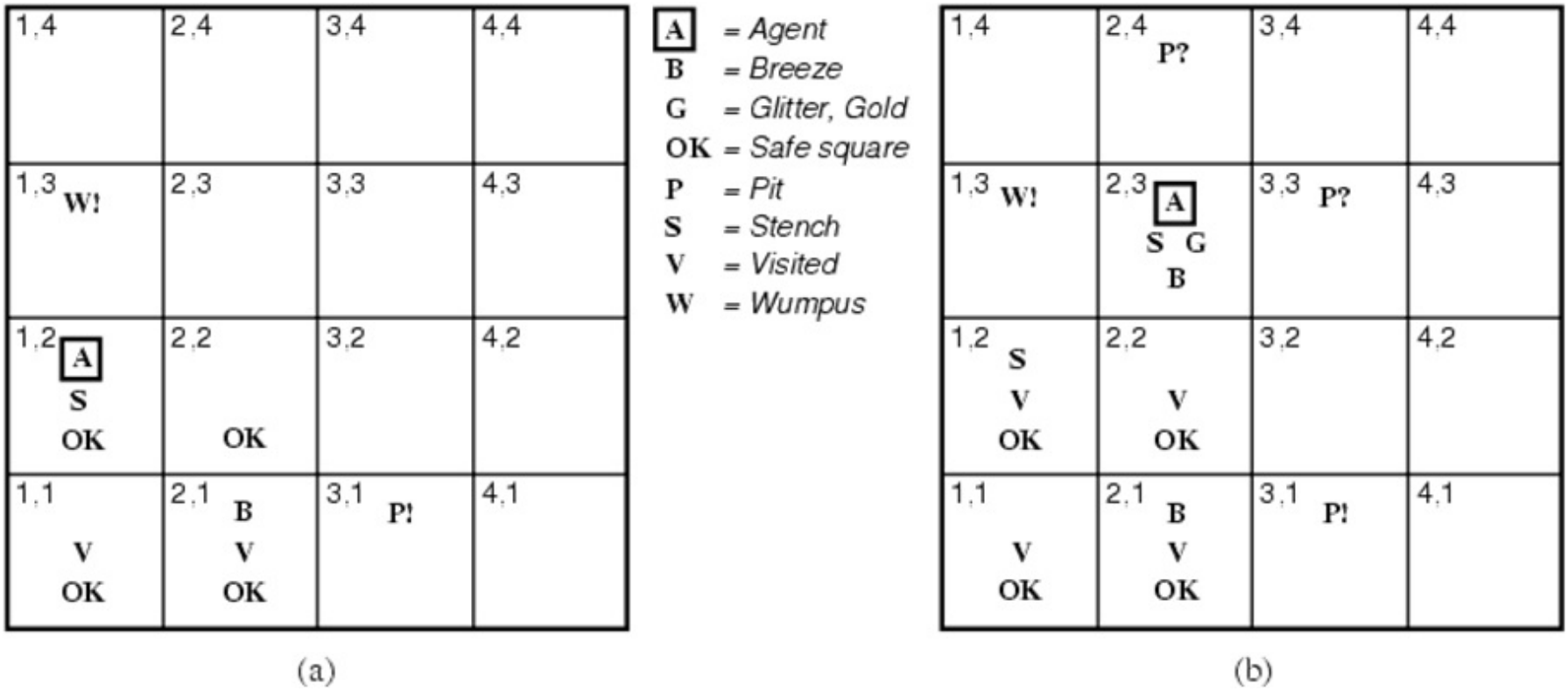
- 性能度量：gold +1000, death -1000, -1 per step, -10 for using the arrow
- 环境：4×4 网格，智能体初始在 $[1, 1]$，面向右方，金子和 wumpus 在 $[1, 1]$ 之外随机均匀分布，$[1, 1]$ 之外的任意方格是陷阱的概率是 0.2
- 执行器：go forward, turn right 90 degrees, turn left 90 degrees, grab, shoot, climb, die
- 传感器：stench, breeze, glitter, bump, woeful scream



Wumpus 世界：一个例子



Wumpus 世界：一个例子

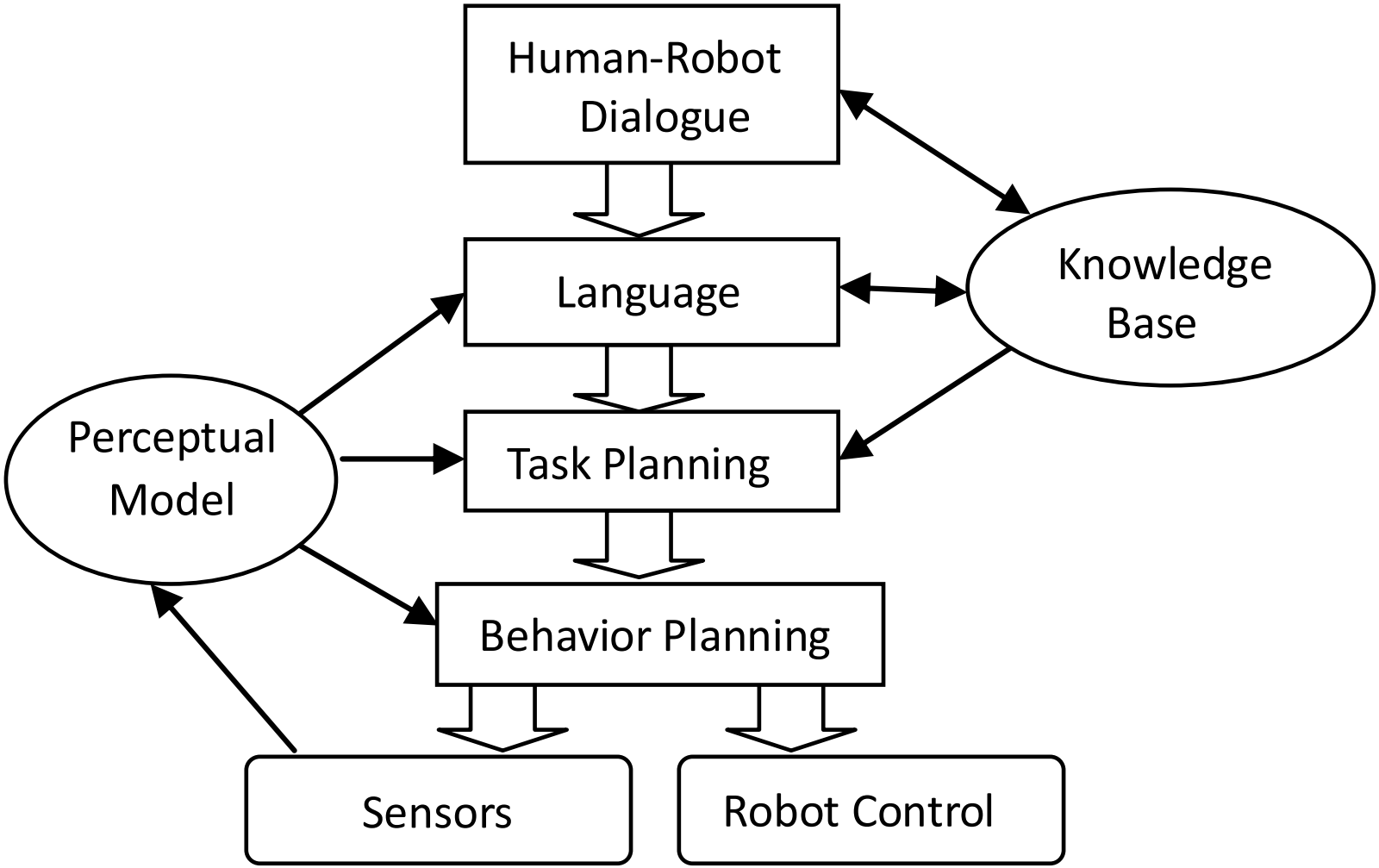


(a) (b)

作业：设计关于 Wumpus 世界的逻辑智能体

- 不要求具体实现，只要求给出“知识库”，说明“推理机”。要求：
  1. 简单说明所用的 KR 语言：可以随意选择，如命题逻辑、逻辑程序等，也可以自己设计新的语言（但需要给出其语法）；
  2. 用所选（所设计）的 KR 语言写出完整的“知识库”；
  3. 解释“知识库”中每个语句的意思（语义）；
  4. 简要说明“推理机”：若基于命题逻辑、逻辑程序等传统工具，则说明如何使用相应的推理工具；若采用自己设计的语言，则需要给出其推理规则，并证明可靠性。

逻辑智能体：可佳机器人


Human-Robot Dialogue
Language
Knowledge Base
Perceptual Model
Task Planning
Behavior Planning
Sensors
Robot Control

（部分）知识库：可佳任务规划

- 任务规划的主要任务：根据用户服务要求，结合机器人现有知识，计算出完成任务的一个行动序列。
- 用行动语言 $\mathcal{C}+$ 表达刻画行动的直接或间接效果的因果知识。

$$
\begin{aligned}
&\textbf{caused } \mathit{occurs}(\mathit{catch}(A)) \textbf{ if } \mathit{occurs}(\mathit{catch}(A)),\\
&\textbf{caused } \bot \textbf{ if } \top \textbf{ after } \mathit{occurs}(\mathit{catch}(A)) \wedge (\neg \mathit{holds}(\mathit{handempty}) \vee\\
&\qquad \neg \mathit{holds}(\mathit{samelocation}(\mathit{agent}, A))),\\
&\textbf{caused } \mathit{holds}(\mathit{holding}(A)) \textbf{ if } \mathit{size}(A, \mathit{small}) \textbf{ after } \mathit{occurs}(\mathit{catch}(A)),\\
&\textbf{caused } \neg \mathit{holds}(\mathit{handempty}) \textbf{ if } \mathit{holds}(\mathit{holding}(A)),\\
&\textbf{caused } \mathit{holds}(\mathit{holding}(A)) \textbf{ if } \mathit{holds}(\mathit{holding}(A))\\
&\qquad \textbf{after } \mathit{holds}(\mathit{holding}(A)),\\
&\textbf{caused } \mathit{holds}(\mathit{handempty}) \textbf{ if } \mathit{holds}(\mathit{handempty})\\
&\qquad \textbf{after } \mathit{holds}(\mathit{handempty}),
\end{aligned}
$$

执行层：可佳任务规划

- $\mathcal{C}+$ 中任意因果律集合都可以等价翻译为 ASP 规则的集合。上述因果律翻译为下列 ASP 规则：

$$
\begin{aligned}
occurs(catch(A),T) &\leftarrow not\ \neg occurs(catch(A),T).\\
\neg occurs(catch(A),T) &\leftarrow not\ occurs(catch(A),T).\\
\neg occurs(catch(A),T) &\leftarrow not\ holds(samelocation(agent,A),T).\\
\neg occurs(catch(A),T) &\leftarrow not\ holds(handempty,T).\\
holds(holding(A),T+1) &\leftarrow occurs(catch(A),T), size(A,small),\\
&\qquad T<lasttime.\\
\neg holds(handempty,T) &\leftarrow holds(holding(A),T).\\
holds(holding(A),T+1) &\leftarrow holds(holding(A),T), T<lasttime,\\
&\qquad not\ \neg holds(holding(A),T+1).\\
\neg holds(holding(A),T+1) &\leftarrow \neg holds(holding(A),T), T<lasttime,\\
&\qquad not\ holds(holding(A),T+1).\\
holds(handempty,T+1) &\leftarrow holds(handempty,T), T<lasttime,
\end{aligned}
$$

小结

- 智能体需要关于世界的知识以便达到良好的决策。
- 知识以**知识表示语言**（KR language）的**语句**（sentences）的形式存储在**知识库**（KB）中。
- 逻辑智能体（基于知识的智能体）由**知识库**和**推理机**构成。通过对**知识库**中语句进行**推理**，以得到新的语句，并由此进行决策。
- 基本概念：
  - 语法（Syntax）：合法语句的形式化结构
  - 语义（Semantics）：语句的真值解释
  - 语义后承（Entailment）：在语义意义下的成真结论
  - 推理（Inference）：基于语句的形式化推理
  - 可靠性（Soundness）：推理结论都是语义后承
  - 完成性（Completeness）：语义后承都能被推理出来

Outline

命题逻辑语法

- 符号表：
  - 命题符号/命题变元：$x_1, x_2, \ldots, x_n, \ldots$（可数无穷多个，代表原子命题）
  - 联结词：$\neg$, $\supset$ ($\rightarrow$), $\wedge$, $\vee$, $\equiv$ ($\leftrightarrow$)
  - 辅助符号：(,)
- 公式（语句，sentences）：p, q 语法变元，代表公式，但自身不是
  1. 任何命题符号都是公式；
  2. 若 p 是公式，则 $\neg p$ 是公式；
  3. 若 p, q 是公式，则 $p \supset q$ 是公式，$p \wedge q$ 是公式，$p \vee q$ 是公式，$p \equiv q$ 是公式；
  4. 只有经过有限次应用上述步骤生成的是公式。

命题逻辑语义

- 解释（interpretation）I 是一个映射：$Atoms \to \{true, false\}$，其中 $Atoms$ 表示语言中所有命题变元的集合
- 公式在解释 I 下的赋值：

$\neg p$	is true iff	p is false
$p \supset q$	is true iff	p is false or q is true
$p \wedge q$	is true iff	P is true and q is true
$p \vee q$	is true iff	p is true or q is true
$p \equiv q$	is true iff	p and q have the same truth value

- 解释 I 称为公式集 T 的模型（model）iff T 中所有公式在 I 下为真

$$(p \vee q) \equiv (\neg p \supset q) \quad \text{definition of } \vee$$
$$(p \wedge q) \equiv \neg(p \supset \neg q) \quad \text{definition of } \wedge$$
$$(p \equiv q) \equiv (p \supset q) \wedge (q \supset p) \quad \text{definition of } \equiv$$

有效性和可满足

- 公式（语句）p 是有效的（valid），如果 p 在所有模型下都为真
  - 重言式（tautologies）$p \supset p, (\neg p \supset p) \supset p$
  - $\{p \supset q, q \supset r\} \models p \supset r$
- 公式（语句）p 是可满足的（satisfiable），如果 P 在某些模型下为真
  - $p \vee q$
  - $\{p, p \vee \neg q\}$ 下 $p \wedge q$ 可满足
- 公式（语句）p 是不可满足的（unsatisfiable），如果不存在模型是的 P 为真
  - 永假式（contradiction）$p \wedge \neg p$
  - $\{p, \neg p \vee q\}$ 下 $\neg p \vee \neg q$ 不可满足
- 公式 p 有效 iff $\neg p$ 不可满足
  - $KB \models p$ iff $KB \wedge \neg p$ 不可满足

命题逻辑中推理方法

- 命题逻辑中，判断 $KB \models p$ 的计算复杂性为 coNP-complete；判断 p (或 $KB \wedge \neg p$) 是否可满足的计算复杂性为 NP-complete。
- 模型检测的方法：构造或找到一个可满足的模型
  - 穷举法（真值表法）：枚举所有可能的解释，逐个判断。时间复杂度为 $O(2^n)$，空间复杂度为 $O(n)$。
  - 回溯搜索方法：DPLL
  - 启发式搜索方法（可靠但不完备）
- 基于推理规则的方法：从老公式推出新公式，构造或找到一个证明过程
  - 归结（resolution）：将公式化为子句形式，不断使用归结推理规则，直到没有可以添加的新语句或出现空子句。
  - 前向/后向链接（forward/backward chaining）：线性时间，针对 Horn 子句（至多只有一个正文字的文字析取式）是完备的。

逻辑等值

- 若 $\models p \equiv q$，则称 p 与 q 逻辑等值
- 可证等价替换规则：设 q 是 p 的子公式，p' 是在 p 中用公式 q' 替换 q 所得的公式，若 $\models q \equiv q'$，则 $\models p \equiv p'$
- 例如：$p \vee q \equiv q \vee p$，则 $(r \wedge (p \vee q)) \supset s \equiv (r \wedge (q \vee p)) \supset s$

一些有用的逻辑等值

$(p \wedge q) \equiv (q \wedge p)$	commutativity of $\wedge$
$(p \vee q) \equiv (q \vee p)$	commutativity of $\vee$
$((p \wedge q) \wedge r) \equiv (p \wedge (q \wedge r))$	associativity of $\wedge$
$((p \vee q) \vee r) \equiv (p \vee (q \vee r))$	associativity of $\vee$
$\neg\neg p \equiv p$	double-negation elimination
$(p \supset q) \equiv (\neg q \supset \neg p)$	contraposition
$(p \supset q) \equiv (\neg p \vee q)$	implication elimination
$(p \equiv q) \equiv ((p \supset q) \wedge (q \supset p))$	biconditional elimination
$\neg(p \wedge q) \equiv (\neg p \vee \neg q)$	De Morgan
$\neg(p \vee q) \equiv (\neg p \wedge \neg q)$	De Morgan
$(p \wedge (q \vee r)) \equiv ((p \wedge q) \vee (p \wedge r))$	distributivity of $\wedge$ over $\vee$
$(p \vee (q \wedge r)) \equiv ((p \vee q) \wedge (p \vee r))$	distributivity of $\vee$ over $\wedge$

合取范式

- 文字（literal）：原子语句（命题变元）或原子语句的否定，例 x, $\neg x$
- 子句（clause）：文字的析取，例 $x_1 \vee \neg x_2 \vee x_3$
- 合取范式（Conjunctive Normal Form, CNF）：子句的合取，例 $(x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)$
- 公式可以等价转化为合取范式（借助上页的逻辑等值）
  1. 消去 $\supset$ 和 $\equiv$；
  2. $\neg$ 深入；
  3. 对 $\wedge$ 和 $\vee$ 进行分配、整理。

$$
\begin{aligned}
&(x_1 \wedge (x_2 \supset x_3)) \supset x_2 \\
&\neg(x_1 \wedge (\neg x_2 \vee x_3)) \vee x_2 && \text{消去 } \supset \\
&(\neg x_1 \vee (x_2 \wedge \neg x_3)) \vee x_2 && \neg \text{ 深入} \\
&(\neg x_1 \vee x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) && \text{分配率}
\end{aligned}
$$

归结算法

- 归结规则：
  - Unit Resolution:

$$\frac{\alpha \vee \beta \quad \neg\beta}{\alpha}$$

  - Resolution:

$$\frac{\alpha \vee \beta \quad \neg\beta \vee \gamma}{\alpha \vee \gamma}$$

- 归结算法：将公式化为子句集合，不断使用归结规则，直到没有可以添加的新语句（可满足）或出现空子句（不可满足）。
- 为了证明 $KB \models p$，需要证明 $KB \wedge \neg p$ 是不可满足的，将其转为 CNF，再使用归结算法。
- 命题逻辑中归结算法是可靠、完备的。

小结

- 命题逻辑是最简单的一种 KR 语言。
- $KB \models p$ iff $KB \wedge \neg p$ 不可满足。
- 命题逻辑中，判断 $KB \models p$ 的计算复杂性为 coNP-complete；判断 p (或 $KB \wedge \neg p$) 是否可满足的计算复杂性为 NP-complete。
- 命题逻辑中推理方法分为两大类：模型检测的方法、基于推理规则的方法
- 命题逻辑中归结算法是可靠、完备的。

Outline

一阶逻辑语言

- 符号表：
  - 逻辑符号:
    - 个体变元：$x_1, x_2, \ldots, x_n, \ldots$ 可数无穷多个
    - 联结词：$\neg, \supset, \wedge, \vee, \equiv$
    - 量词：$\forall, \exists$
  - 非逻辑符号：
    - 个体常元：$a_1, a_2, \ldots$ 至多可数无穷多个
    - 函项符号：$f_1^1, f_2^1, \ldots$ 一元函项符号，$f_1^2, f_2^2, \ldots$ 二元函项符号，...，至多可数无穷多个
    - 谓词符号：$P_1^0, P_2^0, \ldots$ 0 元谓词符号，$P_1^1, P_2^1, \ldots$ 一元谓词符号，...，至多可数无穷多个
  - 辅助符号：(,)

一阶逻辑语法

- 形成规则：
  - 项（term）
    1. 个体变元和个体常元是项；
    2. 若 f 是 n 元函项符号，$t_1, \ldots, t_n$ 是项，则 $f(t_1, \ldots, t_n)$ 是项；
    3. 只有经过有限次应用以上步骤得到的是项。
  - 一阶公式（语句，sentences）：
    1. 若 P 是 n 元谓词符号，$t_1, \ldots, t_n$ 是项，$P(t_1, \ldots, t_n)$ 是公式，称为原子公式；
    2. 若 p, q 是公式，$\neg p, p \supset q, p \wedge q, p \vee q, p \equiv q$ 是公式，称为复合公式；
    3. 若 x 是个体变元，p 是公式，则 $\forall x\, p, \exists x\, p$ 是公式，称为量化公式；
    4. 只有经过有限次应用以上步骤得到的是公式。
- 例如：$\forall x\, (P(x) \supset D(x)), \forall x\, ((student(x) \wedge at(x, USTC)) \supset smart(x)), \exists x\, (at(x, USTC) \wedge smart(x))$

术语

- 子公式：p 是 q 的一个子公式，如果 p 是 q 的一部分
  1. q 是 q 的子公式；
  2. 若 r 是 q 的子公式，则 r 是 $\neg q, \forall x\, q, \exists x\, q$ 的子公式；
  3. 若 r 是 p 或 q 的子公式，则 r 是 $p \supset q, p \wedge q, p \vee q, p \equiv q$ 的子公式。
- 个体变元的自由出现和约束出现
  1. $\forall x\, p$ $(\exists x\, p)$ 中 x 的所有出现都是约束出现，其中 p 称为 $\forall x$ $(\exists x)$ 的“辖域”。
  2. x 在任意公式 p 中的一个出现 x^0 是约束出现，当且仅当存在 p 的一个子公式 q，q 包含 x^0 且 x^0 在 q 约束出现。
  3. x 在任意公式 p 中的一个出现时自由出现，如果它并不是约束出现。
- 闭项/开项：不含个体变元的项称为闭项，否则称为开项。
- 闭公式/开公式：所有个体变元都没有自由出现的公式称为闭式，否则称为开式。
- 项 t 对公式 $p(x)$ 中 x 自由
  - $p(x)$ 表示个体变元 x 在公式 p 中可能有自由出现；
  - 若 x 在 $p(x)$ 中有自由出现，用项 t 处处同时替换 x 在 $p(x)$ 中所有自由出现，所得结果公式为 $p(t)$；
  - 若 t 中个体变元在 $p(t)$ 中都是自由出现，称项 t 对 $p(x)$ 中 x 自由。

一阶逻辑公理系统

- 公理模式

  (K1) $p \supset (q \supset p)$
  (K2) $(p \supset (q \supset r)) \supset ((p \supset q) \supset (p \supset r))$
  (K3) $(\neg p \supset \neg q) \supset (q \supset p)$
  (K4) $\forall x\, p(x) \supset p(t)$ 项 t 对 $p(x)$ 中 x 自由
  (K5) $\forall x\, (p \supset q) \supset (p \supset \forall x\, q)$ x 不在 p 中自由出现

- 推理规则

  (MP) 从 $p, p \supset q$ 推出 q
  (UG) 从 p 推出 $\forall x\, p$

一阶解释

- 一阶结构：设 $K(Y)$ 是任意的一阶语言，$K(Y)$ 的一个一阶结构是一个三元组 $M = (\mathcal{D}, \mathcal{F}, \mathcal{P})$，其最终 $\mathcal{D}$ 是一个非空集合，称为 M 的论域，$\mathcal{F}$ 是 $\mathcal{D}$ 上函数集合，$\mathcal{P}$ 是 $\mathcal{D}$ 上关系集合，满足：
  1. 对 $K(Y)$ 中每一个个体常元 a, $\mathcal{D}$ 中有一个个体 $a^M \in \mathcal{D}$;
  2. 对 $K(Y)$ 中每一个 $n\ (\geq 0)$ 元函数符号 f，$\mathcal{F}$ 中有一个 n 元函数$f^M : \mathcal{D}^n \to \mathcal{D}$;
  3. 对 $K(Y)$ 中每一个 $n\ (> 0)$ 元谓词符号 P，$\mathcal{P}$ 中有一个 n 元关系$P^M \subseteq \mathcal{D}^n$。
- 个体变元指派：对任给 $K(Y)$ 及其一阶结构 $M = (\mathcal{D}, \mathcal{F}, \mathcal{P})$，$K(Y)$ 的一个（相对于 M 的）个体变元指派是一个映射 $V : Y \to \mathcal{D}$。
- 一阶解释：$K(Y)$ 的一个一阶解释是一个复合映射 $I = (M, V)$，其中 M 是 $K(Y)$ 的一个一阶结构，V 是 $K(Y)$ 相对于 M 的一个个体变元指派。

一阶逻辑语义

- 公式在一阶解释 I 下的赋值：$\neg p, p \supset q, p \wedge q, p \vee q, p \equiv q$ 与命题下解释相同
  - 对任何 $x \in Y, I(x) = V(x)$；对任何个体常元 $a, I(a) = a^M$；对任何函数符号 $f, I(f) = f^M$；对任何谓词符号 $P, I(P) = P^M$；对任何项 $f(t_1, \ldots, t_n), I(f(t_1, \ldots, t_n)) = f^M(I(t_1), \ldots, I(t_n))$.

$P(t_1, \ldots, t_n)$	is true iff	$(I(t_1), \ldots, I(t_n)) \in P^M$
$\forall x\, p$	is true iff	对所有 $d \in \mathcal{D}$, $I_d^x(p)$ is true
$\exists x\, p$	is true iff	$\neg\forall x\, \neg p$ is true

- 其中，$I_d^x = (M, V|_d^x)$,

$$V|_d^x(y) = \begin{cases} d & \text{if } y = x \\ V(y) & \text{otherwise} \end{cases}$$

模型和有效性

- M 有效：任给 $K(Y)$ 的一阶结构 M，$p \in K(Y)$，若对一切 V，p 在 $I = (M, V)$ 下为真，则称 p 是 M 有效的，又称 M 是 p 的一个模型，记为 $M \models p$。
- 逻辑有效：设 $p \in K(Y)$，若对 $K(Y)$ 的一切一阶结构 M，$M \models p$，则称 p 为逻辑有效，记为 $\models p$。
- 模型：任给 $K(Y)$ 的一阶结构 M 和 $\Gamma \subseteq K(Y)$，对所有 $p \in \Gamma$ 有 $M \models p$，则称 M 是 Γ 的一个模型，记为 $M \models \Gamma$。
- 语义后承：设 $\Gamma \subseteq K(Y), p \in K(Y)$，若对一切一阶结构 M 有：$M \models \Gamma$ 则 $M \models p$，则称 p 为 Γ 的语义后承，记为 $\Gamma \models p$。
- 一阶逻辑的公理系统与语义是可靠、完备的。
- 一阶逻辑的局限：无法刻画极小、传递闭包等关系。
  例：$\forall x, y\ parent(x, y) \supset ancestor(x, y)$,
  $\forall x, y, z\ ancestor(x, y) \wedge ancestor(y, z) \supset ancestor(x, z)$.

一阶逻辑中推理方法

- 一阶逻辑是不可判定的：对于任意公式 p，可以证实没有判定过程，判定 p 是否有效（停机问题）。
  - 不存在程序可以通过有效步骤判断公式 p 是否为任意公式集 Γ 的逻辑结论。
- 一阶逻辑有效性的判定问题是半可判定的。对于任何有效的公式 p，p 是可证明的（存在有限步骤）。
- 很多 KR 语言（如，OWL, description logic）是一阶逻辑的可判定子集。
- 模型检测的方法：不可行，因为可能的解释是无穷的。
- 基于推理规则的方法：归结（resolution），Tableaux method，等

公式转换为前束范式

- 前束范式：前束范式是任何形式为 $Q_1\,x_1 \cdots Q_n\,x_n\,p$ 的公式，其中 Q_i 代表 $\forall$ 或 $\exists$，p 中不出现量词，称为原式的母式。若 q' 是前束范式且 $\models q' \equiv q$，则称 q' 为公式 q 的前束范式。
- 前束范式转换规则：令 Q^* 为 Q 的对偶量词
  1. 若 y 不在 $p(x)$ 中自由出现，且 x 不在 $p(y)$ 中自由出现，则 $\models Q\,x\,p(x) \equiv Q\,y\,p(y)$
  2. 若 x 不在 p 中自由出现，$\models (p \supset Q\,x\,q) \equiv Q\,x\,(p \supset q)$；
     若 x 不在 q 中自由出现，$\models (Q\,x\,p \supset q) \equiv Q^*\,x\,(p \supset q)$
  3. $\models \neg Q\,x\,p \equiv Q^*\,x\,\neg p$
  4. $\models (\forall x\,p \wedge \forall x\,q) \equiv \forall x\,(p \wedge q)$
  5. $\models (\exists x\,p \vee \exists x\,q) \equiv \exists x\,(p \vee q)$
  6. 若 x 不在 p 中自由出现，则 $\models (p \vee \forall x\,q) \equiv \forall x\,(p \vee q)$，
     $\models (p \wedge \exists x\,q) \equiv \exists x\,(p \wedge q)$

公式转换为合取范式

- 一阶公式转换为合取范式的步骤：
  1. 化为前束范式
  2. Skolem 化：用消元法消去存在量词。前束范式中，若 $\exists x$ 左边不存在全程量词，则将母式中 x 替换为新常量 a；若 $\exists x$ 左边存在全程量词 $\forall x_1 \cdots \forall x_m$，则将母式中 x 替换为项 $f(x_1, \ldots, x_m)$，其中 f 为新的 n 元函数词。
     - 例如：$\forall x \exists y, z\ ((Animal(y) \wedge \neg Loves(x, y)) \vee loves(z, x))$ Skolem 化结果为 $\forall x\ ((Animal(f(x)) \wedge \neg Loves(x, f(x))) \vee loves(g(x), x))$
  3. 去除全称量词
  4. 将母式化为 CNF
- 性质：任给公式 p，s 为 p 相应的 CNF，则 p 是不可满足的 iff s 是不可满足的。
- 性质：$\models s \supset p$ 但 $\not\models p \supset s$.

置换

- 置换（substitution）：置换是形如 $\{v_1/t_1,\ldots,v_n/t_n\}$ 的一个有限集。其中 v_i 是变量而 t_i 是不同于 v_i 的项（常量、变量、函数），且 $v_i \neq v_j\ (i \neq j),\ i,j = 1,2,\ldots,n.$
- 置换可作用于公式或项上。置换 $\theta = \{v_1/t_1,\ldots,v_n/t_n\}$ 作用于一个公式（或项）E，称为 E 的一个实例，记为 $E\theta$，为同时将 E 中出现的所有变量 v_i 替换为项 $t_i\ (i = 1,\ldots,n)$。
  - 例如：$E = P(x,y,f(a)),\ \theta = \{x/b,y/x\}$，则 $E\theta = P(b,x,f(a))$。
- 置换的合成：令 $\theta = \{u_1/s_1,\ldots,u_m/s_m\},\ \sigma = \{v_1/t_1,\ldots,v_n/t_n\}$ 为替换，则 θ 与 σ 的合成也是一个置换，记为 $\theta\sigma$，是从集合

  $$\{u_1/s_1\sigma,\ldots,u_m/s_m\sigma,v_1/t_1,\ldots,v_n/t_n\}$$

  中删去以下两种元素后得到的集合
  - 当 $u_i = s_i\sigma$ 时，从中删除 $u_i/s_i\sigma\ (i = 1,2,\ldots,n)$;
  - 当 $v_j \in \{u_1,\ldots,u_m\}$ 时，从中删除 $v_j/t_j\ (i = 1,2,\ldots,n).$
- 置换的合成 $\theta\sigma$ 等价于先用 θ 做置换，则用 σ 做置换。
  - 例如：$\theta = \{x/f(y),y/z\},\ \sigma = \{x/a,y/b,z/y\}$，则 $\theta\sigma = \{x/f(b),z/y\}.$

合一

- 合一（unifier）：任给原子集 $\mathcal{S} = \{S_1, \ldots, S_k\}$，若存在一个置换 θ，可使 $S_1\theta = S_2\theta = \cdots = S_k\theta$，则称 θ 是 $\mathcal{S}$ 的一个合一。同时称 $\mathcal{S}$ 是可合一的。
  - 例如：$\{P(f(x), z), P(y, a)\}$ 是可合一的，$\sigma = \{y/f(a), x/a, z/a\}$ 就是一个合一。
- 最一般合一（most general unifier, mgu）：合一 θ 称为原子集 $\mathcal{S}$ 的最一般合一，如果对 $\mathcal{S}$ 的任意合一 σ 都存在一个置换 γ 使得 $\sigma = \theta\gamma$.
  - 例如：$\{P(f(x), z), P(y, a)\}$ 的最一般合一为 $\theta = \{y/f(x), z/a\}$，$\sigma = \theta\{x/a\}$。
- 一个原子集的最一般合一是（等价）唯一的。

最一般合一计算方法

- 分歧集（disagreement set）：$\mathcal{S}$ 为原子集。先定位最左边的符号位置，使得 $\mathcal{S}$ 中表达式之间存在符号的不同；再从此位置开始，抽取出所有表达式中的子表达式（项）。这些子表达式（项）的集合构成 $\mathcal{S}$ 的分歧集。
  - 例如：$\mathcal{S} = \{P(f(x), h(y), a), P(f(x), z, a), P(f(x), h(y), b)\}$，其分歧集为 $\{h(y), z\}$.
- UNIFICATION ALGORITHM
  1. Put $k = 0$ and $\sigma_0 = \epsilon$.
  2. If σ_k 是 $\mathcal{S}$ 的合一，then 停止并且 σ_k 是 $\mathcal{S}$ 的 mgu；else 找出 $\mathcal{S}\sigma_k$ 的分歧集 D_k.
  3. If D_k 中存在 v 和 t 使得 v 是不出现于 t 中的变量，then put $\sigma_{k+1} = \sigma_k\{v/t\}, k = k + 1$ and go to 2；else 停止，$\mathcal{S}$ 是不可合一的。

谓词归结

- 归结规则

$$\frac{\alpha \vee p \quad \neg q \vee \gamma}{(\alpha \vee \gamma)\theta}$$

其中子句 $\alpha \vee p$ 与 $\neg q \vee \gamma$ 已标准化分离，原子集 $\{p, q\}$ 的 mgu 为 θ.

- 例子：$C_1 = P(x) \vee Q(f(x)), C_2 = R(g(y)) \vee \neg Q(f(a))$，已知 mgu $\sigma = \{x/a\}$，则归结式为 $P(a) \vee R(g(y))$.
- 归结过程：要证明 $KB \models \alpha \Rightarrow$ 证明 $KB \wedge \neg\alpha$ 不可满足 $\Rightarrow$ 得到公式的 CNF $\Rightarrow$ 标准化分离（换名使不同子句不同享变量）得到子句集 $\mathcal{E} \Rightarrow$ 对 $\mathcal{E}$ 中子句做归结 $\Rightarrow$ 将归结式放入 $\mathcal{E}$ 中，反复归结 $\Rightarrow$ 得到空子句 $\Rightarrow$ 得证。
- 谓词归结算法是可靠、完备的。

作业：一阶逻辑中的推理

- 9.4, 9.18

小结

- 一阶逻辑有效性的判定问题是半可判定的。
- 若原子集是可合一的，则存在（等价）唯一的 mgu，并且可以机械的计算出来。
- 谓词归结算法是可靠、完备的。

Outline

逻辑程序设计

- 逻辑程序的基本观点：Algorithm = Logic + Control
  - Logic：要解决的问题是什么
  - Control：如何去解决问题
- 传统程序设计：关注 Control，掩盖了 Logic。如，C, C++
- 逻辑程序设计：用户只需要编写 Logic 部分，而系统能自动完成 Control 部分。如，Prolog
- 逻辑程序设计的特点：
  - 将 Logic 与 Control 分离：同一个问题 Logic 部分可以通用，Control 部分可以使用不同的方法，不断改进。
  - 程序只包含纯粹的 Logic 部分便于验证，保证可靠性。
  - 为更高层次的编程，减轻用户负担。只需告诉系统做什么（What to do），而不需告知如何去做（How to do）。
  - 相对效率不那么高。

逻辑程序的语法

- 在逻辑程序中，所有“语句”都表示成“规则”的形式：

$$A \leftarrow A_1, A_2, \ldots, A_m.$$

其中 A 是一个原子，称为改规则的“头”，A_i 也是原子，合称为该规则的“体”。

- $m = 0$ 的规则称为“事实”，没有体，只有头，往往省略反箭头 $\leftarrow$.
- 文字的析取称为一个“子句”。不严格的说，一条规则 $A \leftarrow A_1, A_2, \ldots, A_m$ 代表一个子句 $A \vee \neg A_1 \vee \neg A_2 \vee \cdots \vee \neg A_m$。
- 最多只有一个正文字的子句，称为 Horn 子句。每条逻辑程序的规则对应一个 Horn 子句。
- 默认子句中出现的所有个体变元都被全称量化。因此，子句 $A \vee \neg A_1 \vee \neg A_2 \vee \cdots \vee \neg A_m$ 实际上是全称闭式：

$$\forall x_1, x_2, \ldots, x_n \; A \vee \neg A_1 \vee \neg A_2 \vee \cdots \vee \neg A_m$$

的简写，其中 $x_1, x_2, \ldots, x_n$ 是子句中出现的所有个体变元。

Herbrand 基

- 任给程序 P, P 的一个“常项”（ground term）是一个由 P 中出现的个体常元和函数符号复合而成的项。
- P 的所有常项的集合称为 P 的 Herbrand 域（Herbrand universe），记为 $U(P)$。
- 例如：$P_1 = \{p(1)., q(2)., q(x) \leftarrow p(x).\}$，则 $U(P_1) = \{1, 2\}$。
- P 的一个“常原子”是一个由 P 的常项和 P 中出现的谓词符号组成的原子。
- P 的所有常原子的集合称为 P 的 Herbrand 基（Herbrand base），记为 $B(P)$。例如，$B(P_1) = \{p(1), p(2), q(1), q(2)\}$.

逻辑程序的常例 (grounding)

- 任给一个规则 R，R 的一个“常例”（ground instance）是通过下述操作产生的不含个体变元的规则：对 R 中出现的每一个体变元 x，处处同时带入一个常项。
- 例如，P_1 中第三条规则有两个常例:

$$q(1) \leftarrow p(1).$$
$$q(2) \leftarrow p(2).$$

- 逻辑程序的常例是其规则的所有常例的集合。
- 当出现函数词时，Herbrand 基为可数无穷，逻辑程序的常例也可能是无穷的。

逻辑程序的 Herbrand 解释

- 逻辑程序的语义与经典语义不同，建立在 Herbrand 解释之上。
- 一阶逻辑语义：一个解释 I 由一个论域 D 以及一个从项和公式到 D 上函数和关系的映射 V 构成。给定 I，任何公式都可被映射为真或者假。
- 任给程序 P，P 的任何一个 Herbrand 解释 I_H 的论域取为 $U(P)$，即 P 的 Herbrand 域。I_H 的解释函数 V_H 将 P 的任意常项和常原子映射为它们自身。
- P 的一个 Herbrand 解释还包含一个由 P 的若干常原子组成的“指派”集合 $M(P) \subseteq B(P)$。一个常原子 p 在指派 $M(P)$ 下为真，当且仅当 $p \in M(P)$。
- 任何规则 R 在一个 Herbrand 解释 I_H 下的真值根据 $M(P)$ 如下：
  1. $I_H(R) = t$，当且仅当对 R 的所有常例 R^* 有 $I_H(R^*) = t$；
  2. 若 $A \leftarrow A_1, A_2, \ldots, A_m$ 是一个规则的常例，则

$$I_H(A \leftarrow A_1, A_2, \ldots, A_m) =_{df} I_H(A \vee \neg A_1 \vee \neg A_2 \vee \cdots \vee \neg A_m),$$

当且仅当，$I_H(A) = t$ 或者存在 A_i 使得 $I_H(\neg A_i) = t$，
当且仅当，$A \in M(P)$ 或者存在 $A_i \notin M(P)$。

逻辑程序的 Herbrand 模型

- 一个 Herbrand 解释称为一个规则的 Herbrand 模型，如果该规则在该解释下为真。
- 一个 Herbrand 解释称为一个程序的 Herbrand 模型，如果该程序的所有规则在该解释下为真。
- 严格说，一个程序 P 的一个 Herbrand 解释是一个三元组 $I_H = \langle U(P), V_H, M(P) \rangle$。但 $U(P)$ 和 V_H 对任何 P 都是“固定的”，所以通常用指派集合 $M(P) \subseteq B(P)$ 代表一个 Herbrand 模型，简记为 M。
- 例如，$P_1 = \{ p(1)., q(2)., q(x) \leftarrow p(x). \}$ 有两个 Herbrand 模型: $\{ p(1), q(1), q(2) \}$ 和 $\{ p(1), p(2), q(1), q(2) \}$。

逻辑程序 Herbrand 模型性质

Proposition 1

令 S 为子句的集合，S 是不可满足的 *iff* S 不存在 *Herbrand* 模型。

Proposition 2

任何逻辑程序都有 *Herbrand* 模型。

Proposition 3

逻辑程序的两个 *Herbrand* 模型的交集也是该程序的 *Herbrand* 模型。

Proposition 4

任何逻辑程序有唯一的极小*Herbrand* 模型（*least Herbrand model*）M_P。

Proposition 5

两个逻辑程序 P_1, P_2，如果 $P_1 \subseteq P_2$，则 $M_{P_1} \subseteq M_{P_2}$。

SLD-归结规则

令$H \leftarrow B_1, \ldots, B_n$ 是程序中规则，$\leftarrow A_1, \ldots, A_m$ 为目标（已经重命名变量），θ 为 A_i 与 H 的 mgu。则：

$$\frac{\leftarrow A_1, \ldots, A_m \quad H \leftarrow B_1, \ldots, B_n}{\leftarrow (A_1, \ldots, A_{i-1}, B_1, \ldots, B_n, A_{i+1}, \ldots, A_m)\theta}$$

小结

- 逻辑程序设计：只需要编写 Logic，即问题是什么。
- 逻辑程序有唯一的极小 Herbrand 模型。