COMP9414: 人工智能

第4b讲：自动推理

韦恩-沃布克

电邮：w.wobcke@unsw.edu.au

本讲座

- 证明系统
  健全性、完整性、可解性
- 解决和反驳
- 角子句和SLD解析
- ロード

到目前为止的总结

- 命题逻辑
  - 语法。由 ∧, ∨, ¬, → 建立的形式语言
  - 语义学。每个公式的真值表的定义
  - S |= *P* 如果*S*中的所有公式都是真的，那么*P*就是真的。
- 证明系统
  - 公理系统和推理规则
  - 使得可以从*S*中计算出*P*的证明。
- 基本问题
  - 计算出来的证明总是正确的吗？(健全性)
  - 如果*S* |= *P*，是否总是有一个来自*S*的*P*的证明（完备性）。

机械化证明

- Tableau方法

- 从一组前提S对公式P的证明是一个线条序列，其中证明中的任何线条都是
  1. 一个逻辑公理或S的前提，或
  2. 利用推理规则从以前的证明行中推导出的公式

  而证明的最后一行是公式P
- 从形式上把握了数学证明的概念
- 如果存在来自S的P的证明，则S证明P（$S \vdash P$）；或者，P由S得出。
- 例子。自然演绎法证明

新南威尔士大学
©W.Wobcke等人，2019-2022年

健全性和完备性

- 如果（从直觉上讲）一个证明系统保留了真理，那么它就是健全的。
  - 只要$S \vdash P$，如果S中的每个公式都是真，P也是真。
  - 每当$S \vdash P, S \models P$
  - 如果你以真实的假设开始，任何结论都必须是真实的
- 如果一个证明系统能够证明任何一组前提（包括无限集）的所有后果，那么它就是完整的。
  - 每当P被S所包含时，就有一个来自S的P的证明。
  - 每当$S \models P$，$S \vdash P$
- 如果存在一个机械程序（计算机程序），当被问及$S \vdash P$时，总能正确回答 "是 "或 "否"，那么这个证明系统就是可解的。

决议

- 另一种基于反驳的证明系统
- 比公理和规则系统更适合计算机实现（可以给出正确的 "不 "的答案）
- 在命题逻辑的情况下是可解的
- 归纳为一阶逻辑（见本学期后文）。
- 需要将所有公式转换为句子形式

正常形式

- 文字ℓ是一个命题变量或一个命题变量的否定（P或¬P）。
- 一个子句是一个字词的二元连接 $\ell_1 \vee \cdots \vee \ell_n$
- 连词正常形式（CNF）--分句的连词，例如：1. $(P \vee Q \vee \neg R) \wedge (\neg S \vee \neg R)$ - 或者只有一个子句，例如：$P \vee Q$
- 二元正态形式 (DNF) - 字词的连词的分解，例如 $(P \wedge Q \wedge \neg R) \vee (\neg S \wedge \neg R)$ - 或者只有一个连词，例如 $P \wedge Q$
- 每个命题逻辑公式都可以转换为CNF和DNF
- 每个命题逻辑公式都等同于其CNF和DNF

转换为共轭正常形式

- 消除↔，将$P \leftrightarrow Q$改写为 $(P \rightarrow Q) \wedge (Q \rightarrow P)$。
- 消除→将$P \rightarrow Q$改写为¬P $\vee Q$
- 使用德摩根定律将¬向内推（重复）。
  - 将$\neg(P \wedge Q)$改写为¬P $\vee \neg$Q
  - 将$\neg(P \vee Q)$改写为¬P $\wedge \neg$Q
- 消除双重否定：将¬¬P改写为P
- 使用分配律得到CNF[或DNF]--如果需要的话
  - 将 $(P \wedge Q) \vee R$ 改写为 $(P \vee R) \wedge (Q \vee R)$ [对于 CNF]

- 将 $(P \vee Q) \wedge R$ 改写为 $(P \wedge R) \vee (Q \wedge R)$ [对于 DNF]

新南威尔士大学
©W.Wobcke等人，2019-2022年

句子形式示例

句子形式=CNF中的句子集合

- ¬(*P*→(*Q*∧*R*))
- ¬(¬p ∨(*q* ∧*r*))
- ¬¬P ∧¬(*Q* ∧*R*)
- ¬¬p ∧(¬q ∨¬r)
- *P* ∧(¬Q ∨¬R)
- 句子形式。{P, ¬Q ∨¬R}

解决推理规则

$a_1 \vee \cdots \vee a_m \vee b$ $\qquad$ $\neg b \vee c_1 \vee \cdots \vee c_n$

$a_1 \vee \cdots \vee a \vee c_{m1} \vee \cdots \vee c_n$

其中*B*是一个命题变量，A_i 和C_j 是字面意思。

- *B*和¬B是互补的字词
- $A_1 \vee \cdots \vee A_m \vee C_1 \vee \cdots \vee C_n$

决议规则。关键理念

- 考虑$A_1 \vee \cdots \vee A_m \vee B$ 和¬B $\vee C_1 \vee \cdots \vee C_n$
  - 假设两者都是真的
  - 如果*B*为真，¬B为假，$C_1 \vee \cdots \vee C_n$ 为真
  - 如果*B*是假的，$A_1 \vee \cdots \vee A_m$ 是真。
  - 因此，$A_1 \vee \cdots \vee A \vee C_{m1} \vee \cdots \vee C_n$ 为真

因此，决议规则是合理的。

- 从真实的前提开始，任何使用分辨率做出的结论一定是真的

应用决议。天真的方法

- 将知识库转换为句子形式

对所产生的句子重复应用解决规则

当且仅当知识库中的每一条款

*P*的CNF可以用解析法从P的条款中得到。

知识库

- 例子
  - {P→*Q*， *Q*→*R*}。⊢ *P* → *R*
  - 句子¬P ∨*Q*, ¬Q ∨*R*, 显示¬P ∨*R*

是两个条款的解析式

- 从 一 个 决议步骤开始。
- 特殊情况。如果没有A_i和C_j， 解析器为空句，表示为□。

新南威
尔士大
学
©W.Wobcke等人，2019-
2022年

驳斥系统

- 要用反驳法证明P是由S推导出来的（即$S \vdash P$），首先要用S和¬P的句子形式，并使用解析法推导出一个矛盾。
- 矛盾是 "空句"（没有字面意义的句子）。
- 空句□是不可满足的（总是假的）。
- 因此，如果用解析法推导出空句□，那么原来的句子集是不可满足的（绝不是所有的真都在一起）。
- 也就是说，如果我们能从S和¬P的句子形式中推导出□，那么这些句子就不可能全部都是真。
- 因此，只要S的子句都是真，那么至少有一个子句来自于¬P一定是假的，即¬P一定是假的，P一定是真的
- 根据定义，$S \models P$（所以P可以从S中正确得出结论）。

应用决议反驳

- 否定要证明的询问（决议是一个反驳系统）
- 将知识库和否定式查询转换为CNF
- 反复应用解析，直到推导出空句（矛盾）或无法推导出更多的句子。
- 如果导出空子，回答 "是"（查询从知识库中导出），否则回答 "否"（查询不从知识库中导出）。

决议。例1

$(g \vee h) \rightarrow (\neg j \wedge \neg k)$, $g \vdash \neg j$

$(G \vee H) \rightarrow (\neg J \wedge \neg K)$ 的句子形式是

{¬g ∨ ¬j, ¬h ∨ ¬j, ¬g ∨ ¬k, ¬h ∨ ¬k}。

1. ¬G ∨ ¬J [前提]
2. ¬H ∨ ¬J [前提]
3. ¬G ∨ ¬K [前提]
4. ¬H ∨ ¬K [前提]
5. G [前提]
6. J [¬查询]
7. ¬G [1，6决议]
8. □ [5，7决议]

决议。例2

$p \rightarrow \neg q, \neg q \rightarrow r \vdash p \rightarrow r$

回顾$P \rightarrow R \Leftrightarrow \neg P \vee R$

¬(¬P ∨ R)的句子形式是{P, ¬R}。

1. ¬P ∨ ¬Q [前提]
2. $Q \vee R$ [前提]
3. P [¬查询]
4. ¬R [¬ 查询]
5. ¬Q [1，3决议]
6. R [2，5决议]

7. □ [4，6决议]

新南威
尔士大
学
©W.Wobcke等人，2019-
2022年

决议。例3

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

¬(((*P* ∨*Q*)∧¬P)→*Q*)的句子形式是{P ∨*Q*, ¬P, ¬Q}。

1. *P* ∨*Q* [¬ 查询]
2. ¬P [¬ 查询]
3. ¬Q [¬查询]
4. *Q* [1，2决议]
5. □ [3，4决议]

再谈健全性和完备性

对于命题逻辑

- 决议反驳是合理的，即它保留了真理（如果一组前提都是真的，那么从这些前提得出的任何结论也必须是真的）。
- 决议反驳是完整的，即它能够证明任何知识基础的所有后果（这里没有显示！）。
- 解析反驳是可解的，即有一种实现解析的算法，当问及*S*⊢*P*时，总能回答 "是 "或 "不是"（正确）。

应用决议时的启发式方法

- 条款消除 - 可以不考虑某些类型的条款
  纯粹子句：包含¬L不出现在其他地方的字词*L*
  - 同义词：同时包含*L*和¬L的句子
  - 归并子句：另一个子句是字词的一个子集
- 订购策略
  - 首先解决单元句（只有一个字面）。
  - 从查询条款开始
  - 旨在缩短条款

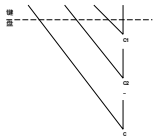
喇叭状条款

主意。少用表达性语言

- 评论
  - literal - 命题变量或命题变量的否定。
  - 子句--字词的分离连接
- 定语从句--正好是一个正面的字词
  - 例如，$B \vee \neg A_1 \vee \ldots \vee \neg A_n$，即 $B \leftarrow A_1 \wedge \ldots \wedge A_n$
- 负数条款--没有正数字样
  - 例如：$\neg Q_1 \vee \neg Q_2$ (查询的否定)
- 角子句--最多只有一个正字的句子

SLD分辨率 - ⊢SLD

- 选定的字词 线性形式 定语从句解析
- 从一组条款KB中对一个条款C的SLD反驳是一个序列
  1. 序列的第一个子句是C
  2. 每个中间子句C_i 都是通过解析前一个子句C_{i-1}和KB中的一个子句副本而得到的。
  3. 序列中的最后一个子句是 □



- 该定理。对于一个确定的KB和否定句子查询Q：$KB \cup Q \vdash \square$ 当且仅当 $KB \cup Q \vdash_{SLD} \square$

ロード

- 一阶逻辑中的霍恩条款（见本期后文）
- SLD决议
- 带有回溯功能的深度优先搜索策略
- 用户控制
  - Prolog数据库中条款的排序（事实和规则）。
  - 在规则主体中对子目标进行排序
- Prolog是一种基于决议反驳的编程语言，依赖于程序员利用搜索控制规则

Prolog实例

```
r.                  # 事实
u.
v.

q :- r, u.          # 规则
s :- v.
p :- q, r, s.

?-p.                #查询
是
```


Prolog解释器

输入。一个查询Q和一个逻辑程序KB

输出。如果Q来自KB，则为 "是"，否则为 "否"

将当前目标集初始化为{Q}。

虽然当前的目标集不是空的，但做

从当前的目标集中选择G；（目标集中的第一个）。

从KB中选择一个条款的副本G' :- B_1 , . . ., B_n 的一个条款（尝试KB中的所有条款）（如果没有这样的规则，则尝试其他规则）。

用B_1 , . . . 替换G。 , B_n 在当前目标集中

如果当前的目标集是空的，输出 "是"。

否则输出 "不"。

- 深度优先，左右逢源，逆向追踪

新南威尔士大学
©W.Wobcke等人，2019-2022年

Tableau方法

Alpha Rules:

$$\frac{A \wedge B}{\begin{array}{c} A \\ B \end{array}} \qquad \frac{\neg(A \vee B)}{\begin{array}{c} \neg A \\ \neg B \end{array}} \qquad \frac{\neg(A \rightarrow B)}{\begin{array}{c} A \\ \neg B \end{array}}$$

¬¬-Elimination:

$$\frac{\neg\neg A}{A}$$

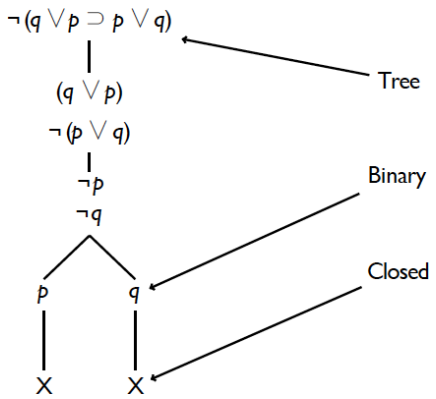
Beta Rules:

$$\frac{A \vee B}{A \mid B} \qquad \frac{A \rightarrow B}{\neg A \mid B} \qquad \frac{\neg(A \wedge B)}{\neg A \mid \neg B}$$

Branch Closure:

$$\frac{\begin{array}{c} A \\ \neg A \end{array}}{\times}$$

Tableau方法实例



结论。命题逻辑

- 由 ∧, ∨, ¬, → 建立的命题
- 健全的、完整的和可解码的证明系统（推理程序）。
  - 自然扣减
  - 决议反驳
  - 定语从句的特殊情况的Prolog
  - Tableau方法
- 表达能力有限
  - 不能表达本体，例如AfPak本体
- 一阶逻辑可以表达关于对象、属性和对象之间关系的知识