

Artificial Intelligence

人工智能实验

归结原理

中山大学计算机学院
2024年春季

目录

1. 理论课内容回顾

1.1 基本概念

1.2 命题逻辑归结算法

1.3 MGU（最一般合一）算法

1.4 一阶逻辑的归结算法

2. 实验任务

用归结算法求解逻辑推理问题（如AIpine Club问题）

1.1 基本概念

□ 以Alpine Club问题为例

- Tony, Mike, and John belong to the Alpine Club.
- Every member of the Alpine Club who is not a skier is a mountain climber.
- Mountain climbers do not like rain, and anyone who does not like snow is not a skier.
- Mike dislikes whatever Tony likes, and likes whatever Tony dislikes.
- Tony likes rain and snow.
- Is there a member of the Alpine Club who is a mountain climber but not a skier?

1.1 基本概念

□ Alpine Club问题形式化为

■ 已知条件（知识库）

□ Facts

■ $A(\text{tony})$

■ $A(\text{mike})$

■ $A(\text{john})$

■ $L(\text{tony}, \text{rain})$

■ $L(\text{tony}, \text{snow})$

□ Rules

■ $\forall x(A(x) \wedge \neg S(x)) \rightarrow C(x)$

■ $\forall x(C(x) \rightarrow \neg L(x, \text{rain}))$

■ $\forall x(\neg L(x, \text{snow}) \rightarrow \neg S(x))$

■ $\forall x(L(\text{tony}, x) \rightarrow \neg L(\text{mike}, x))$

■ $\forall x(\neg L(\text{tony}, x) \rightarrow L(\text{mike}, x))$

■ 提问

□ $\exists x(A(x) \wedge C(x) \wedge \neg S(x))$ 是否成立

1.1 基本概念

□ 相关概念

- 常量 (constant) : 任何类型的实体
 - 俱乐部成员: tony, mike, john
 - 天气类型: rain, snow
- 变量 (variable) : 如x, y这类未知量
- 项 (term) : 可以理解为谓词/变量的参数项, 由递归定义
 - 变量是项 (可以看成是0元函数)
 - $t_1, t_2, t_3, \dots, t_n$ 是项, f 是 n 元函数, 则 $f(t_1, t_2, \dots, t_n)$ 也是项

Tips: 一阶逻辑中谓词不是项, 即不能作为函数/谓词的参数, 也就是不存在 $f(P(x))$ 这种复合方式, 但是二阶逻辑中是可以的

1.1 基本概念

□ 相关概念

- 谓词 (predicate) : 谓词是对其参数 (也叫做项, term) 的
 - 零元谓词: 退化为命题
 - 单元谓词 (unary predicate) : 只有一个参数, 表示参数具备某种属性, 如 $A(x)$ 表示 x 属于 Alpine 俱乐部
 - 多元谓词: 有多个参数, 表示参数之间的关系, 如 $L(x,y)$ 表示 x 和 y 具有喜欢关系, 即 x 喜欢 y

1.1 基本概念

□ 相关概念

■ 事实 (fact) : 谓词中变量实例化后得到事实

□ $S(\text{tony})$: tony是skier

□ $L(\text{tony}, \text{rain})$: tony喜欢下雨天

■ 规则 (rule) : 也叫做公式, 通过递归定义

□ $t_1, t_2, t_3, \dots, t_n$ 是项, P 是 n 元谓词, 则 $P(t_1, t_2, \dots, t_n)$ 是原子公式

□ t_1, t_2 是项, 那么 $t_1 = t_2$ 是原子公式

□ 如果 α 和 β 是公式, 那么 $\neg\alpha, \alpha \wedge \beta, \alpha \vee \beta, \exists\alpha, \forall\alpha$ 都是公式

Tips: 由于 $(\alpha \rightarrow \beta)$ 等价于 $(\neg\alpha \vee \beta)$, $(\alpha \leftrightarrow \beta)$ 等价于 $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$, 所以在递归定义中我们没有加入 \rightarrow 和 \leftrightarrow , 它们可以被已有符号替代

1.1 基本概念

□ 相关概念

■ 可满足性：

- 以Alpine俱乐部为例， $\exists x(A(x) \wedge C(x) \wedge \neg S(x))$ 是否成立就是在问，是否存在一组实例化（一组赋值），使得 $A(x) \wedge C(x) \wedge \neg S(x)$ 成立，这就是一个可满足性问题。对于该可满足性问题，只要能够找到一组赋值（在这里对应 $\{x\}$ 的赋值），使得 $A(x) \wedge C(x) \wedge \neg S(x)$ 成立，那么“ $A(x) \wedge C(x) \wedge \neg S(x)$ ”是可满足的

■ 逻辑蕴含和逻辑推论：

- 逻辑蕴含 $S \models \alpha$ 指对于任意变量赋值，如果 S 正确，则 α 也正确
- 逻辑推论 $S \vdash \alpha$ 指存在一条推理路径，从 S 出发，推导证明 α

1.2 命题逻辑归结算法

□ 定理:

- $S \models ()$ 当且仅当 $S \models ()$, $S \models ()$ 当且仅当 S 是不可满足的
- 通过该定理, 我们可得 $KB \models \alpha$ 当且仅当 $KB \wedge \neg \alpha$ 不可满足, 于是可以通过反证法证明 $KB \models \alpha$

□ 归结算法:

- 将 α 取否定, 加入到KB当中
- 将更新的KB转换为clausal form得到S
- 反复调用单步归结
 - 如果得到空子句, 即 $S \models ()$, 说明 $KB \wedge \neg \alpha$ 不可满足, 算法终止, 可得 $KB \models \alpha$
 - 如果一直归结直到不产生新的子句, 在这个过程中没有得到空子句, 则 $KB \models \alpha$ 不成立

1.2 命题逻辑归结算法

□ 归结算法：

■ Clausal form (便于计算机处理的形式)

- 每一个子句对应一个元组，元组每一个元素是一个原子公式/原子公式的否定，元素之间的关系是析取关系，表示只要一个原子成立，该子句成立

- 如子句 $\neg\text{child} \vee \neg\text{male} \vee \text{boy}$ 对应数据结构 $(\neg\text{child}, \neg\text{male}, \text{boy})$ ，空子句 $()$ 对应False

- 元组的集合组成子句集S，子句集中每个句子之间是合取关系，表示每一个子句都应该被满足

- 由于本次实验重点是归结算法，所以问题输入是已经转换过的clausal form，关于具体转换方式感兴趣的同学可以参考课件

■ 单步归结

- 从两个子句中分别寻找相同的原子，及其对应的原子否定
- 去掉该原子并将两个子句合为一个，加入到S子句集合中
- 例如 $(\neg\text{child}, \neg\text{female}, \text{girl})$ 和 (child) 合并为 $(\neg\text{female}, \text{girl})$

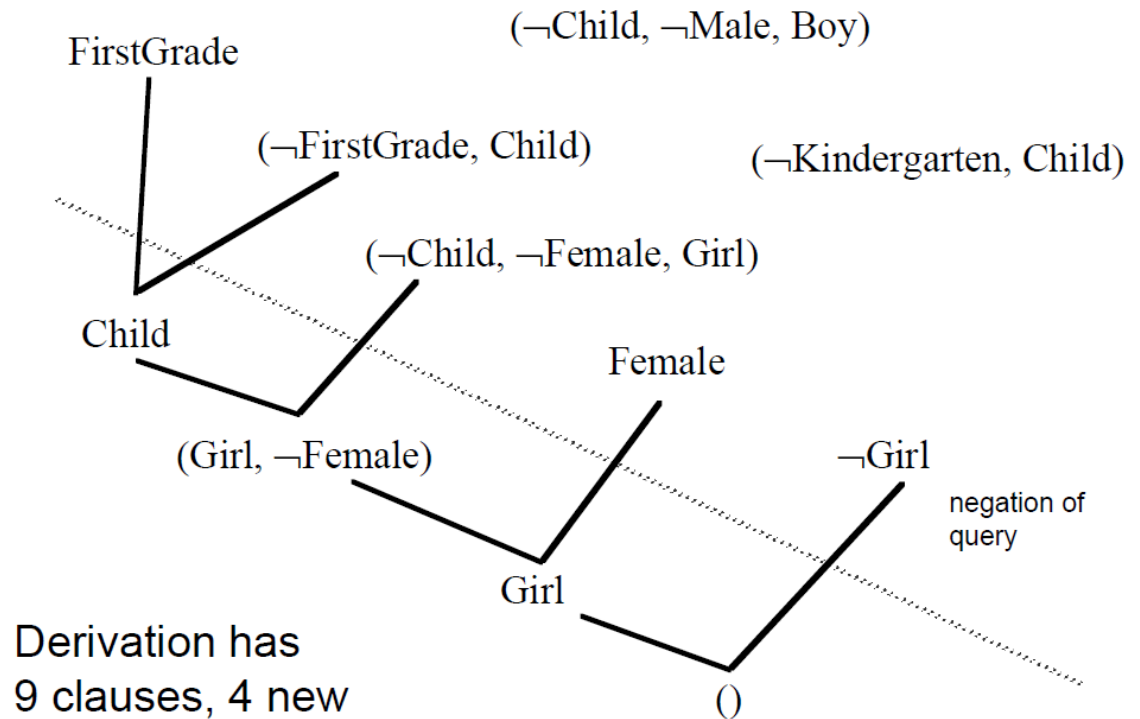
1.2 命题逻辑归结算法

□ 例子

KB

FirstGrade
FirstGrade \rightarrow Child
Child \wedge Male \rightarrow Boy
Kindergarten \rightarrow Child
Child \wedge Female \rightarrow Girl
Female

Show that $KB \models \text{Girl}$



1.3 Most general unifier 算法

□ 最一般合一算法：

■ 合一 (unifier)：

- 通过变量替换使得两个子句能够被归结（有相同的原子），所以合一也被定义为使得两个原子公式等价的一组变量替换/赋值
- 由于一阶逻辑中存在变量，所以归结之前需要进行合一，如 $(P(\text{john}), Q(\text{fred}), R(x))$ 和 $(\neg P(y), R(\text{susan}), R(y))$ 两个子句中，我们无法找到一样的原子及其对应的否定，但是不代表它们不能够归结
- 通过将 y 替换为 john ，我们得到了 $(P(\text{john}), Q(\text{fred}), R(x))$ 和 $(\neg P(\text{john}), R(\text{susan}), R(\text{john}))$ ，此时我们两个子句分别存在原子 $P(\text{john})$ 和它的否定 $\neg P(\text{john})$ ，可以进行归结

■ 最一般合一：指使得两个原子公式等价，最简单的一组变量替换

1.3 Most general unifier算法

☐ 最一般合一算法:

- 输入：两个原子公式，它们具有相同的谓词，不同的参数项和 “ \neg ”
- 输出：一组变量替换/赋值
- 算法流程：
 - ☐ $k = 0; \sigma_0 = \{\}; S_0 = \{f, g\}$
 - ☐ 如果 S_k 中的公式等价，返回 σ_k 作为最一般合一的结果
 - 否则找出 S_k 中的不匹配项 $D_k = \{e_1, e_2\}$
 - ☐ 如果 $e_1 = V$ 是变量， $e_2 = t$ 是一个不包含变量 V 的项，将 “ $V = t$ ” 添加到赋值集合 $\sigma_{k+1} = \sigma_k \cup \{V = t\}$ ；并将 S_k 中的其它 V 变量也赋值为 t ，得到 S_{k+1} ；
 $k = k + 1$ ，转到第二步
 - 否则合一失败

Tips: 变量替换是从两个原子公式中找到的，但是最后要施加给整个子句的

1.3 Most general unifier 算法

□ 例子：

- $P(f(a), g(x))$ 和 $P(y, y)$ 无法合一
- $P(a, x, h(g(z)))$ 和 $P(z, h(y), h(y))$ 最一般合一为 $\{z=a, x=h(g(a)), y=g(a)\}$
- $P(x, x)$ 和 $P(y, f(y))$ 无法合一

作业（不用提交）

1. 命题逻辑的归结推理

编写函数 `ResolutionProp` 实现命题逻辑的归结推理。该函数要点如下：

- 输入为子句集(数据类型与格式详见课件)，每个子句中的元素是原子命题或其否定。
- 输出归结推理的过程，每个归结步骤存为字符串，将所有归结步骤按序存到一个列表中并返回，即返回的数据类型为 `list[str]`。
- 一个归结步骤的格式为 `步骤编号 R[用到的子句编号] = 子句`。如果一个字句包含多个公式，则每个公式用编号 `a,b,c...` 区分，如果一个字句仅包含一个公式，则不用编号区分。（见课件和例题）

例子：输入子句集

```
1 KB = {(FirstGrade,), (~FirstGrade,Child), (~Child,)}
```

则调用 `ResolutionProp(KB)` 后返回推理过程的列表如下：

```
1 1 (FirstGrade,),
2 2 (~FirstGrade,Child)
3 3 (~Child,),
4 4 R[1,2a] = (Child,),
5 5 R[3,4] = ()
```

作业（不用提交）

2. 最一般合一算法

编写函数 `MGU` 实现最一般合一算法。该函数要点如下：

- 输入为两个原子公式，它们的谓词相同。其数据类型为 `str`，格式详见课件。
- 输出最一般合一的结果，数据类型为 `dict`，格式形如{变量：项，变量：项}，其中的变量和项均为字符串。
- 若不存在合一，则返回空字典。

例子：

调用 `MGU('P(xx,a)', 'P(b,yy)')` 后返回字典 `{'xx':'b', 'yy':'a'}`。

调用 `MGU('P(a,xx,f(g(yy)))', 'P(zz,f(zz),f(uu))')` 后返回字典 `{'zz':'a', 'xx':'f(a)', 'uu':'g(yy)'}`。

1.4 一阶逻辑归结算法

□ 归结算法：

- 将 α 取否定，加入到KB当中
- 将更新的KB转换为clausal form得到S
- 反复调用单步归结
 - 如果得到空子句，即 $S|-()$ ，说明 $KB \wedge \neg\alpha$ 不可满足，算法终止，可得 $KB \models \alpha$
 - 如果一直归结直到不产生新的子句，在这个过程中没有得到空子句，则 $KB \models \alpha$ 不成立
- 单步归结
 - 使用MGU算法从两个子句中得到相同的原子，及其对应的原子否定
 - 去掉该原子并将两个子句合为一个，加入到S子句集合中
 - 例如 $(\neg\text{Student}(x), \neg\text{HardWorker}(x))$ 和 $(\text{HardWorker}(\text{sue}))$ 合并为 $(\neg\text{Student}(\text{sue}))$

2. 实验任务

□ 编写程序，实现一阶逻辑归结算法，并用于求解给出的三个逻辑推理问题，要求输出按照如下格式：

1. $(P(x), Q(g(x)))$

2. $(R(a), Q(z), \neg P(a))$

3. $R[1a, 2c]\{X=a\} (Q(g(a)), R(a), Q(z))$

... ..

... ..

“R”表示归结步骤.

“1a”表示第一个子句(1-th)中的第一个 (a-th)个原子公式，即 $P(x)$.

“2c”表示第二个子句(1-th)中的第三个 (c-th)个原子公式，即 $\neg P(a)$.

“1a”和“2c”是冲突的，所以应用最小合一 $\{X = a\}$.

2. 实验任务

□ 存储公式的python数据结构

- 用字符串存储
- 符号 \neg 用‘~’代替
- 谓词的首字母大写, 例如用A, B, C, P1, P2, Student等表示; 谓词的每个参数之间用逗号“,”间隔且不加空格
- 常量用小写单词或a, b, c等小写字母表示;
- 本次作业的公式中不含 \exists, \forall 量词符号
 - 例子: $\neg\text{child}$ 存储为“~child” boy存储为“boy”
 - 几个公式: “R(a)”, “~P(a,zz)”, “Student(tony)”. 这里应该将a,tony看做常量,将zz看做变量

2. 实验任务

- 存储子句的python数据结构
 - 用tuple的方式存储
 - 例子:
 - $\neg \text{child} \vee \neg \text{male} \vee \text{boy}$ 存储为('~child', '~male', 'boy')
 - $\neg S(z) \vee L(z, \text{snow})$ 存储为('~S(z)', 'L(z,snow)')
- 存储子句集的python数据结构
 - 子句集用set的方式存储, 每个元素是子句(元组)

2. 实验任务

□ Alpine Club

- $A(\text{tony})$
- $A(\text{mike})$
- $A(\text{john})$
- $L(\text{tony}, \text{rain})$
- $L(\text{tony}, \text{snow})$
- $(\neg A(x), S(x), C(x))$
- $(\neg C(y), \neg L(y, \text{rain}))$
- $(L(z, \text{snow}), \neg S(z))$
- $(\neg L(\text{tony}, u), \neg L(\text{mike}, u))$
- $(L(\text{tony}, v), L(\text{mike}, v))$
- $(\neg A(w), \neg C(w), S(w))$

```
[sysu_hpcedu_302@cpn238 ~/scc22/lsr/mp_linpack/resolution]$ python main.py
11
A(tony)
A(mike)
A(john)
L(tony, rain)
L(tony, snow)
( $\neg A(x)$ ,  $S(x)$ ,  $C(x)$ )
( $\neg C(y)$ ,  $\neg L(y, \text{rain})$ )
( $L(z, \text{snow})$ ,  $\neg S(z)$ )
( $\neg L(\text{tony}, u)$ ,  $\neg L(\text{mike}, u)$ )
( $L(\text{tony}, v)$ ,  $L(\text{mike}, v)$ )
( $\neg A(w)$ ,  $\neg C(w)$ ,  $S(w)$ )
R[2,11a](w=mike) =  $\neg C(\text{mike}), S(\text{mike})$ 
R[2,6a](x=mike) =  $S(\text{mike}), C(\text{mike})$ 
R[5,9a](u=snow) =  $\neg L(\text{mike}, \text{snow})$ 
R[12b,13a] =  $S(\text{mike})$ 
R[8a,14](z=mike) =  $\neg S(\text{mike})$ 
R[15,16] = []
```

2. 实验任务

□ Graduate Student

- GradStudent(sue)
- $(\neg \text{GradStudent}(x), \text{Student}(x))$
- $(\neg \text{Student}(x), \text{HardWorker}(x))$
- $\neg \text{HardWorker}(\text{sue})$

```
[sysu_hpcedu_302@cpn238 ~/scc22/lsr/mp_linpack/resolution]$ python main.py
4
GradStudent(sue)
(¬GradStudent(x), Student(x))
(¬Student(x), HardWorker(x))
¬HardWorker(sue)
R[3b,4](x=sue) = ¬Student(sue)
R[1,2a](x=sue) = Student(sue)
R[5,6] = []
```

3. 作业提交说明

- ❑ 压缩包命名为：“学号_姓名_作业编号”，例：
20240312_张三_实验3。
- ❑ 每次作业文件下包含两部分：code文件夹和实验报告PDF文件。
 - code文件夹：存放实验代码；
 - PDF文件格式参考发的模板。
- ❑ 如果需要更新提交的版本，则在后面加_v2，_v3。如第一版是“学号_姓名_作业编号.zip”，第二版是“学号_姓名_作业编号_v2.zip”，依此类推。
- ❑ 截至日期：**2024年3月26日晚24点**。
- ❑ 提交邮箱：zhangyc8@mail2.sysu.edu.cn。