# 归结原理

# 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

# 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

- Aipine Club问题形式化为
  - 已知条件（知识库）
    - Facts
      - A(tony)
      - A(mike)
      - A(john)
      - L(tony,rain)
      - L(tony,snow)
    - Rules
      - $\forall x(A(x) \land \neg S(x)) \rightarrow C(x)$
      - $\forall x(C(x) \rightarrow \neg L(x,rain))$
      - $\forall x(\neg L(x,snow) \rightarrow \neg S(x))$
      - $\forall x(L(tony,x) \rightarrow \neg L(mike,x))$
      - $\forall x(\neg L(tony,x) \rightarrow L(mike,x))$
  - 提问
    - $\exists x(A(x) \land C(x) \land \neg S(x))$是否成立

# 1.1 基本概念

## □ 相关概念

- 常量（constant）：任何类型的实体, 通常用a,b,c等符号
  - □ 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
  - □ 常量、变量是项（可以看成是0元函数）
  - □ t1，t2，t3……tn是项，f是n元函数，则f(t1,t2,,,,tn)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(tony)：tony是skier
  - L(tony, rain)：tony喜欢下雨天
- 规则（rule）：也叫做公式，通过递归定义
  - t1，t2，t3……tn是项，P是n元谓词，则P(t1,t2,,,,tn)是原子公式
  - t1，t2是项，那么t1=t2是原子公式
  - 如果$\alpha$和$\beta$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→和↔，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$\alpha$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$\alpha$

# 1.2 命题逻辑归结算法

- 定理：
  - S |-() 当且仅当 S |= ()， S |= () 当且仅当 S 是不可满足的
  - 通过该定理，我们可得KB |= α 当且仅当 KB ∧¬α 不可满足，于是可以通过反证法证明KB |= α
- 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 ∧¬α 不可满足，算法终止，可得KB |= α
    - 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 |= α不成立

# 1.2 命题逻辑归结算法

- 归结算法：
  - Clausal form (子句形式, 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，元组之间的关系是**析取(or)**关系，表示只要一个原子成立，该子句成立
      - 如子句¬child∨¬male∨boy对应数据结构(¬child,¬male,boy)，False对应空子句()
      - 子句的集合组成子句集S，子句集中每个句子之间是**合取(and)**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(¬child,¬female,girl)和(child)合并为(¬female,girl)

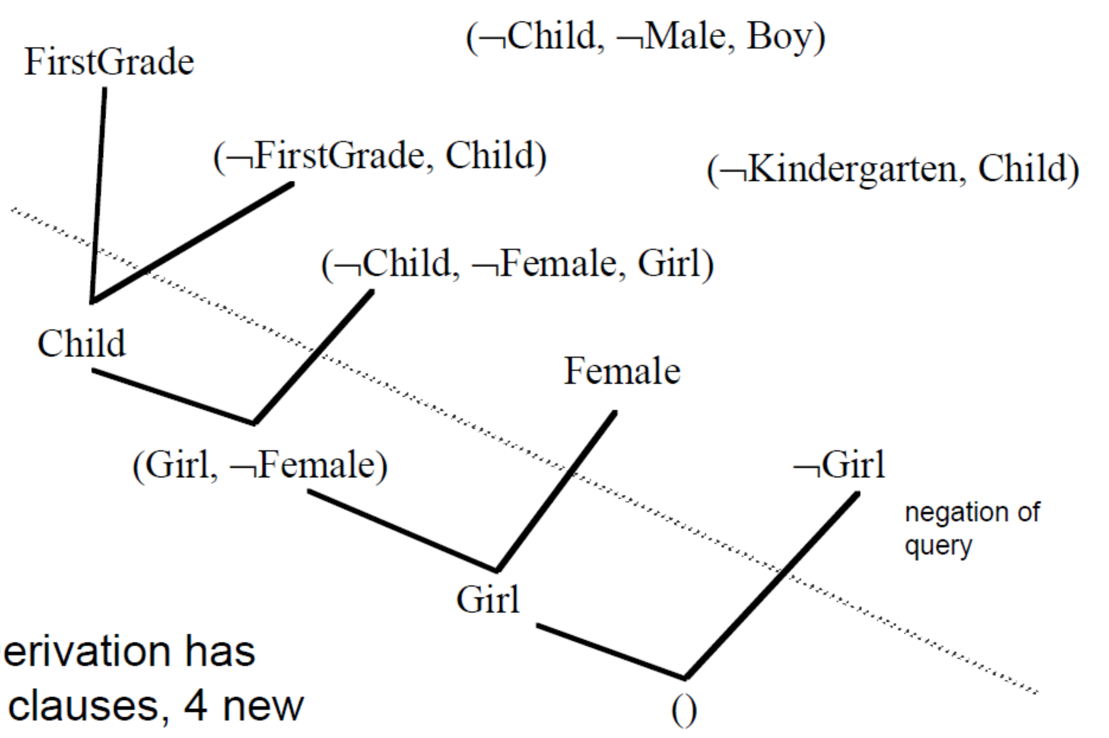
# 1.2 命题逻辑归结算法

- 例子

KB

| |
|---|
| FirstGrade |
| FirstGrade -> Child |
| Child ∧ Male -> Boy |
| Kindergarten -> Child |
| Child ∧ Female -> Girl |
| Female |

Show that KB $\models$ Girl




Derivation has
9 clauses, 4 new

# 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(john),Q(fred),R(x))和(¬P(y),R(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(john),Q(fred),R(x))和(¬P(john),R(susan),R(john))，此时我们两个子句分别存在原子P(john)和它的否定¬P(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“¬”
  - 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$\sigma_k$作为最一般合一的结果
      - 否则找出$S_k$中的不匹配项$D_k = \{e1, e2\}$
    - 如果 e1=V 是变量，e2 = 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

# 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 ∧¬α 不可满足，算法终止，可得KB |= α
    - 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 |= α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(¬Student(x),HardWorker(x))和(HardWorker(sue))合并为(¬Student(sue))

# python数据结构

- 存储**公式**的python数据结构
  - 用字符串存储
  - 符号¬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;
  - 为避免常量, 函数或谓词中含有字母'x', 'y'等, 本次作业中的变量符号仅从以下列表中选取: ['xx','yy','zz','uu','vv','ww']
  - 本次作业的公式中不含∃,∀量词符号
    - 例子: ¬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

# python数据结构

- 存储**子句**的python数据结构
  - 用tuple的方式存储
    - 例子:
    - ¬child∨¬male∨boy存储为(‘~child’, ‘~male’, ‘boy’)
    - ¬S(z) ∨ L(z, 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)

# 第3次作业(第1页/共4页)

## 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"`

例子: 输入子句集

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

则调用 `ResolutionProp(KB)`后返回推理过程的列表如下:

```
["1 ('FirstGrade',)",
 "2 ('~FirstGrade','Child')"
 "3 ('~Child',)",
 "4 R[1,2]: ('Child',)",
 "5 R[3,4]: ()"
]
```

## 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)'}`.

# 第3次作业(第3页/共4页)

## 3.(思考题,选做)一阶逻辑的归结推理

编写函数 `ResolutionFOL`实现一阶逻辑的归结推理. 该函数要点如下:

- 输入形式同第1题, 不过 `KB`子句中的每个元素是一阶逻辑公式(不含∃, ∀等量词符号)
- 输出归结推理的过程, 数据类型同第1题
- 一个归结步骤的格式为 `"步骤编号 R[用到的子句编号]{最一般合一}: 子句"`, 其中最一般合一输出格式为"{变量=常量, 变量=常量}". 若没有用到最一般合一则为"{}".

例子: 输入

```
KB = {('On(a,b)',), ('On(b,c)',), ('Green(a)',), ('~Green(c)',), ('~On(xx,yy)',
'~Green(xx)', 'Green(yy)')}
```

则调用 `ResolutionFOL(KB)`后返回推理过程的列表如下:

```
["1 ('On(a,b)',)",
 "2 ('On(b,c)',)",
 "3 ('Green(a)',)",
 "4 ('~Green(c)',)",
 "5 ('~On(xx,yy)', '~Green(xx)', 'Green(yy)')",
 "6 R[4,5]{yy=c}: ('~On(xx,c)', '~Green(xx)')",
 "7 R[3,5]{xx=a}: ('~On(a,yy)', 'Green(yy)')",
 "8 R[2,6]{xx=b}: ('~Green(b)',)",
 "9 R[1,7]{yy=b}: ('Green(b)',)",
 "10 R[8,9]{}: ()"
]
```

## 提示

1. 只含一个元素的 `tuple`类型要在末尾加 `,`. 例如 `('a')`是错误的写法, 而正确的写法是 `('a',)`.
2. `{}`会被解释成空字典. 若要定义空集合请用 `set()`.
3. 为避免常量, 函数或谓词中含有字母'x', 'y'等, 本次作业中的变量符号仅从以下列表中选取: `['xx','yy','zz','uu','vv','ww']`
4. 请下载代码模板来编写程序并确保测试程序可以跑通. 提交代码时只提交一个 `.py` 代码文件, 请不要提交其他文件.

# 作业提交

- 大家的作业通过连接ftp服务器上传
- 主机IP: 222.200.177.152
- 端口: 1021
- 用户: ftpstu
- 密码: 123456
- 第3次的【实验报告】提交到【hw3_report】文件夹，命名格式为【hw3_学号_姓名拼音.pdf】
- 【源代码文件】提交到【hw3_code】文件夹，命名格式为【hw3_学号_姓名拼音.py】
- 作业提交后不可修改，若要提交新版本请在文件名加后缀“_v1”, “_v2”等
- **第3次作业ddl: 2023年3月26日23:59**