第3周实验课作业

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

```
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[1,2a] = (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)'}` .

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()` .