**中山大学计算机学院**

**人工智能**

**本科生实验报告**

课程名称：Artificial Intelligence

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 |  | 姓名 |  |

# 实验题目

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

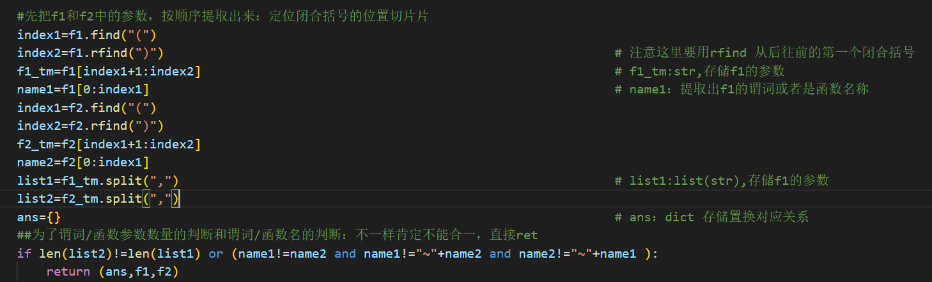
* 输入为子句集, KB子句中的每个元素是一阶逻辑公式(不含量词符号)
* 输出归结推理的过程, 每个归结步骤存为字符串, 将所有归结步骤按序存到一个列表中并返回, 即返回的数据类型为 list[str]
* 一个归结步骤的格式为 步骤编号 R[用到的子句编号]{最一般合一} = 子句, 其中最一般合一输出格式为"{变量=常量, 变量=常量}".如果一个字句包含多个公式，则每个公式用编号a,b,c...区分，如果一个字句仅包含一个公式，则不用编号区分。（见课件和例题）

# 实验内容

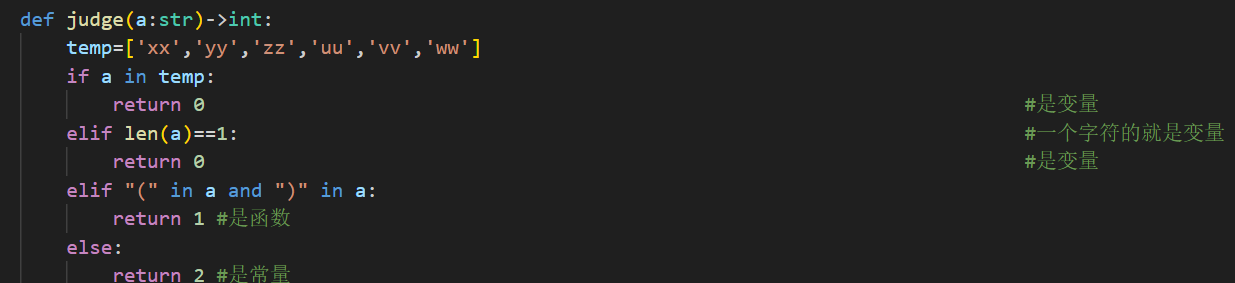
1. 算法原理
2. MGU算法：
3. 传入两个谓词原子公式，并将其作为参数调用递归MGU函数，进入步骤（b）
4. 两个函数项/谓词原子公式进行归结，对比函数项/谓词原子公式，相同则继续归结，不同则终止
5. 提取两个谓词原子公式/函数项的参数项，分为三种类型：变量，常量，函数
6. 遍历对比两个谓词原子公式/函数项相同位置上的参数项，按照不同类型的组合分类讨论，并将得到的置换规则记录下来，且应用到原子公式的所有参数项中：
7. 一个参数是常量，一个参数是函数：不能合一，直接终止
8. 一个参数是变量，一个参数是函数：要先判断变量是否出现在函数中：若出现在函数中，则直接终止，否则变量可以置换为函数(x，g(y)：x->g(y))
9. 一个参数是常量，一个参数是变量：变量可以置换为常量（x，Mary：x->Mary）
10. 两个参数都是变量：可以合一，且置换方向无限制（x，y：x->y 或者y->x）
11. 两个参数都是函数：由于合一是要对比最小单位（原子变量和常量）的参数，则转到a）步骤递归对两个函数合一
12. 两个参数都是常量：不能合一，则直接终止
13. 返回最终合一置换规则
14. 一阶逻辑归结：
    1. 得到子句集clause\_set
    2. 单步归结：在子句集中选择两个子句a，b归结
       1. 采用单子句（只含一个谓词原子公式的子句）优先策略：在子句集中遍历选择一个单子句a，即两个归结子句中必有一个单子句（单子句优先策略的完备性：子句集一开始必含有单子句，否则无法归结出空子句）
       2. 在剩下的子句中遍历选择另外一个子句b与a归结
          * 将a,b合一：遍历b中的原子公式与a中唯一的原子公式合一，（调用MGU）：并且将合一的置换规则运用到a，b整个子句中
          * 判断b中是否有a的唯一的原子公式的否定
            + 若有：则b和a可以归结，把b和a的互为否定的公式删去，将b和a中剩下的公式结合为新的子句，加入子句集中（注意不要修改掉子句a,b,也不需要将a，b从子句集中删掉，否则不完备），若新的子句为空子句，一阶逻辑归结算法结束，转到步骤c
            + 若没有，转到步骤ii继续

* 遍历完子句集中剩下的b，转到步骤i

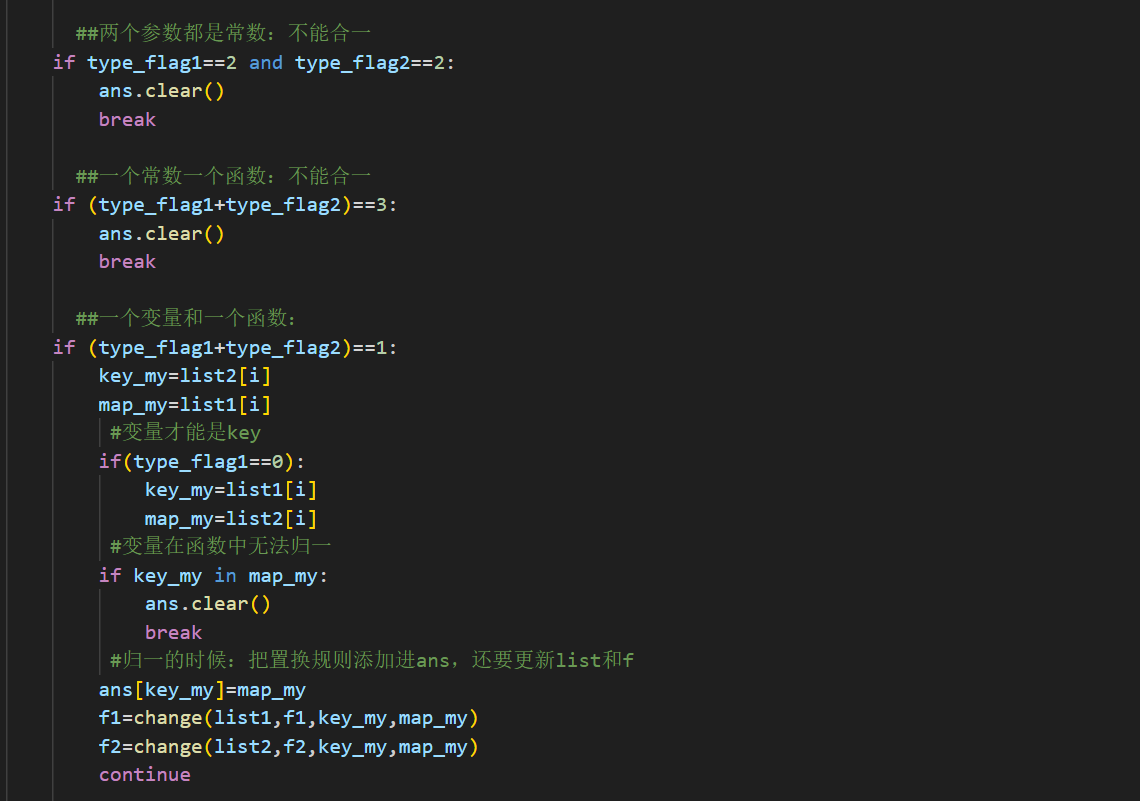
1. 得到空子句退出或者所有的子句遍历结束，没有得到空子句退出
2. 关键代码展示（可选）
3. MGU算法（具体可见图中或者代码文件中的注释）
4. 两个公式或者函数项，对比谓词/函数名，提取参数并分别存储在list1和list2中



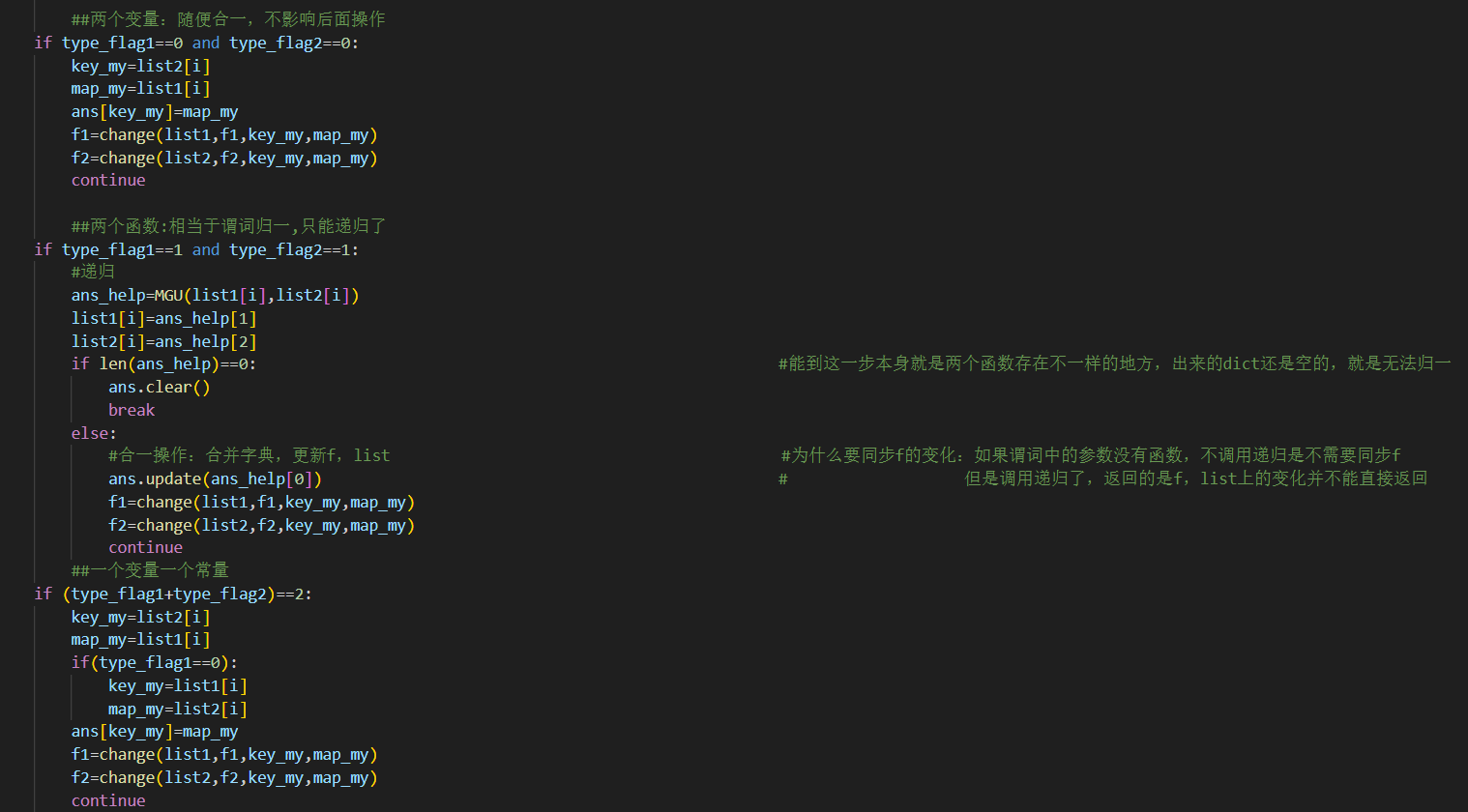
1. 遍历两个公式或者函数项相同位置上的参数，并且分类讨论：



* + 1. 两个都是常量：终结合一
    2. 一个常量一个函数；终结合一
    3. 一个变量一个函数：判断变量是否在函数中，不在则进行变量向函数的置换，并且调用得到的置换规则应用到整个原子公式的所有参数项中



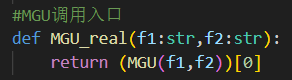
* + 1. 两个都是变量：可以任意方向置换合一，并且调用得到的置换规则应用到整个原子公式所有参数项中
    2. 两个都是函数：递归调用MGU算法，合一要对最小单位参数开始合一
    3. 一个变量一个常量：得到变量置换为常量的合一置换规则，并且调用得到的置换规则应用到整个原子公式所有参数项中



1. MGU算法的函数介绍：
2. MGU\_real：MGU调用入口：

传入参数：谓词原子公式

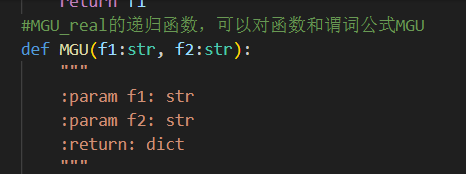
返回：最终置换规则dict



1. MGU：MGU算法的递归函数，考虑到谓词参数有函数的情况，需要递归合一，是做合一的主体函数

传入：函数项或者谓词原子公式

返回：置换规则和合一后的两个原子公式或者函数项



1. Change：把对原子公式单个位置的参数上置换合一得到的置换规则(key->map),应用到原子公式的所有参数上，得到合一更新后的原子公式

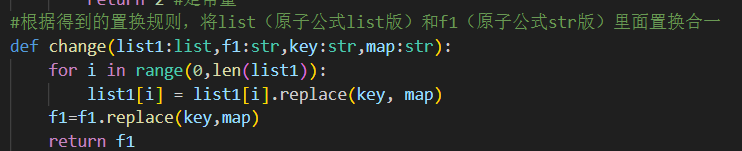
传入：list1(原子公式list版)，str(原子公式str版)，key（被置换的项），

map（要置换成的项）

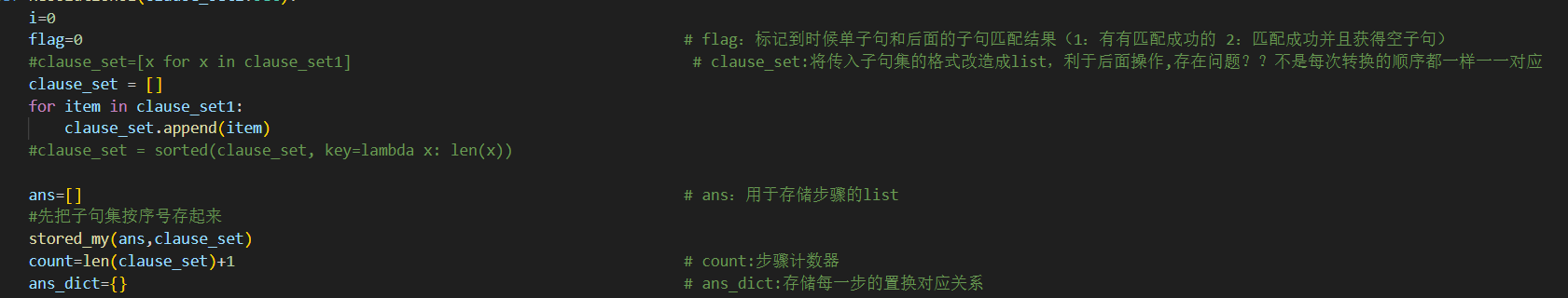
返回：置换后的原子公式（str版）

注意：为什么既要修改str版也要修改list版的原子公式，为了在递

归中方便同步嵌套函数项的置换合一（如f(x,g(x)),具体可见注释）

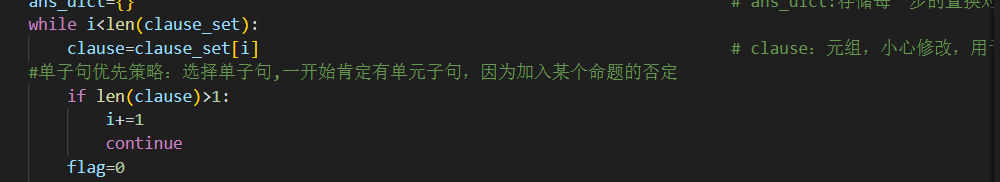


1. 一阶逻辑归结：
   1. 得到子句集，调整子句集的形式（转为list），做一些得到归结步骤的初始化工作



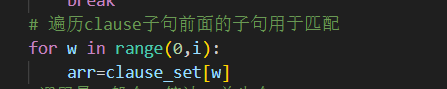
* 1. 从子句集中挑选两个子句进行单步归结：

1. 采取单子句优先策略，先挑出一个单子句clause\_set[i]

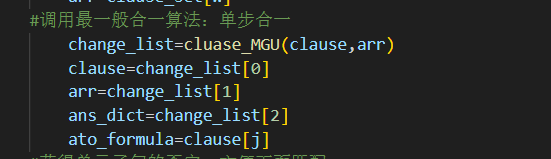


1. 在剩下子句中遍历子句clause\_set[w]，分别和单子句clause\_set[i]归结

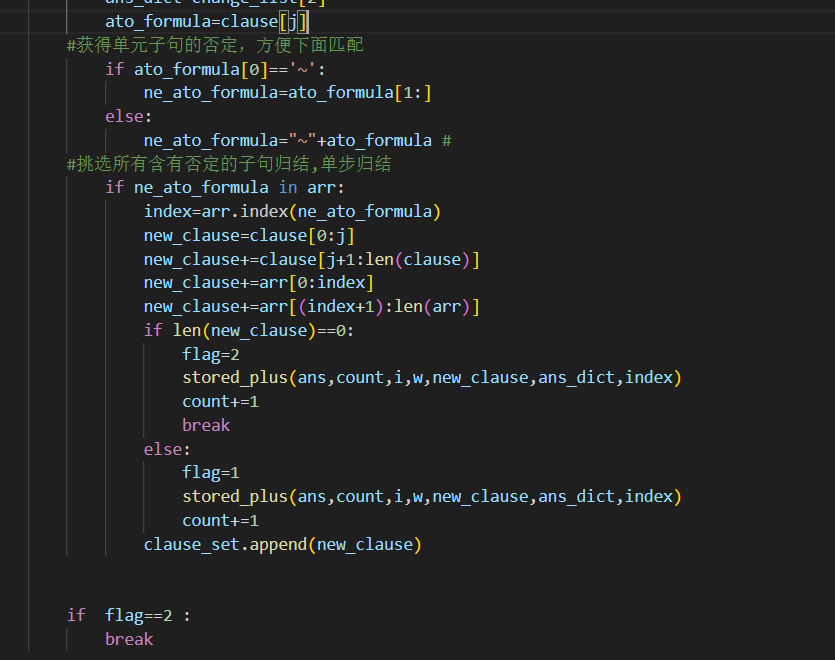




1. 对两个子句置换合一：



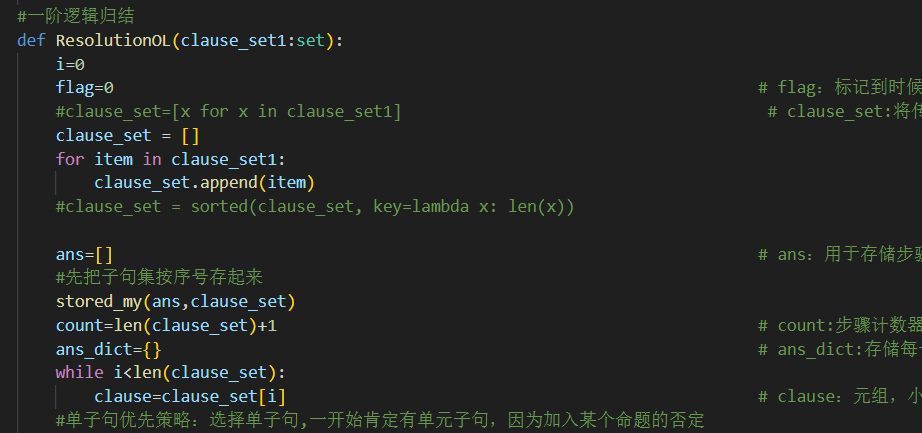
1. 在两个合一后的子句里面寻找互为否定的公式归结：注意clause是clause\_set[i]的引用。Clause[j]就是clause\_set[i]的唯一的原子公式
2. 若归结出空子句则终结



1. 一阶逻辑归结的函数介绍：
2. ResolutionOL：一阶逻辑归结的主题逻辑

输入：子句集

输出：归结步骤

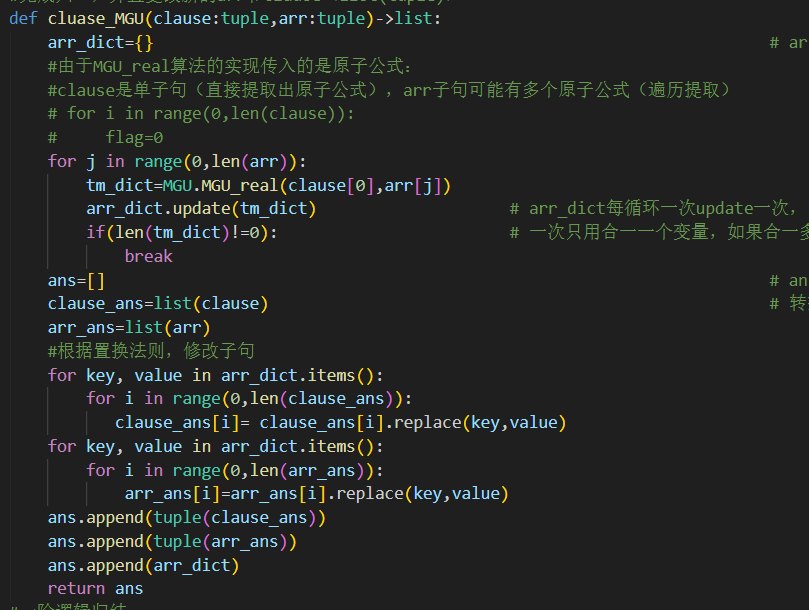


1. Clause\_MGU:一个单子句和一个不限制子句的合一

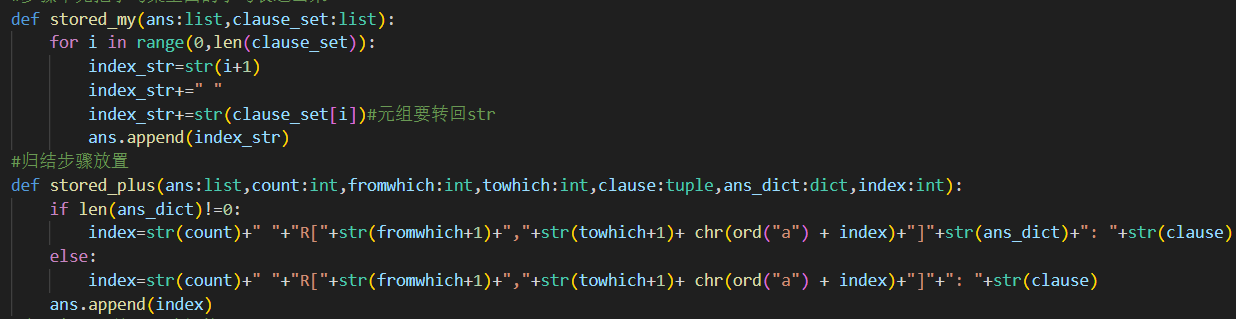
输入：一个单子句，一个子句

输出：合一时得到的置换规则

逻辑：遍历的子句中的原子公式和单子句中唯一的原子公式进行原子公式的MGU归结，得到置换规则，并将置换规则应用到子句的所有公式中



1. Stored\_my和stored\_plus：整理得到归结步骤

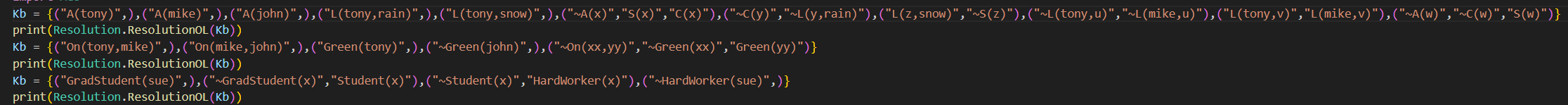


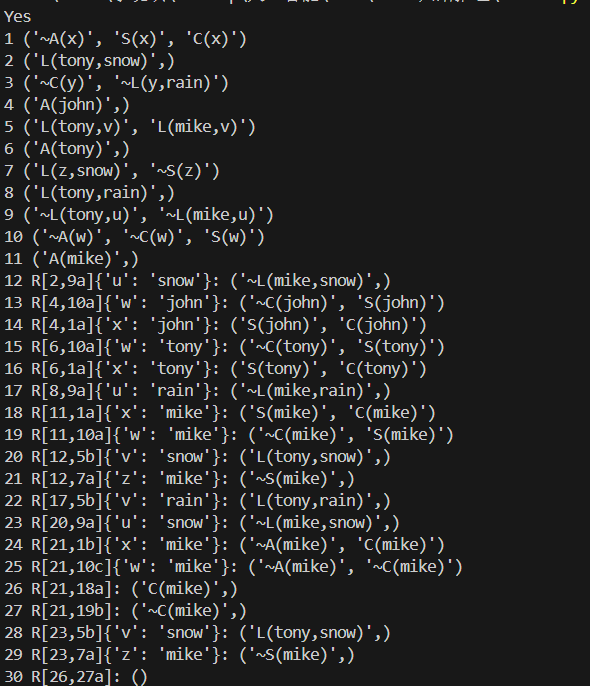
1. 创新点&优化

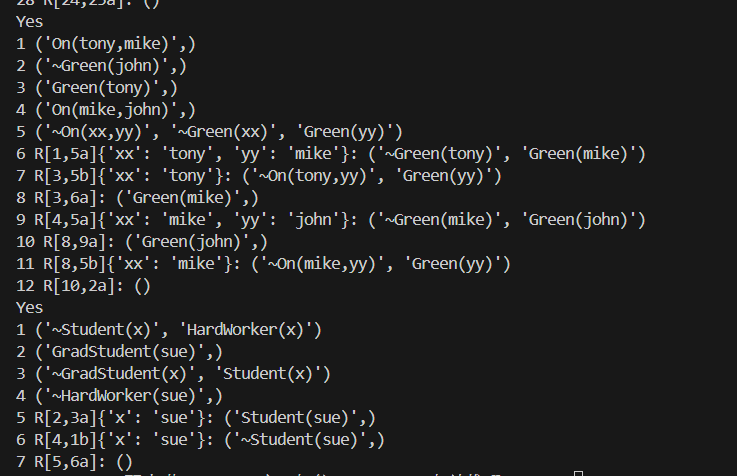
一阶逻辑的归结采用了单子句优先策略来挑选子句单步归结，使得单步归结的两个子句一定含有一个以上的单子句，减少归结和合一时的遍历开销

# 实验结果及分析

1. 实验结果展示示例

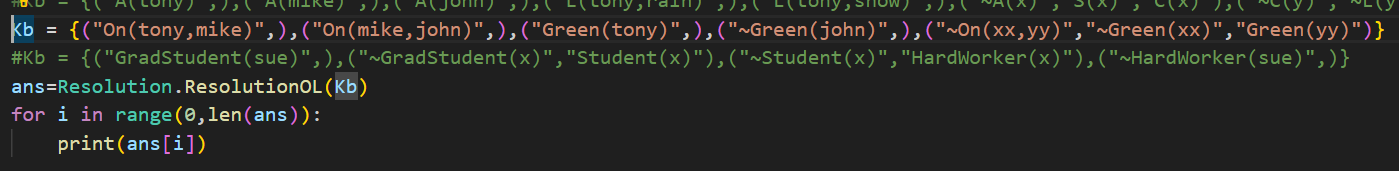




2. 评测指标展示及分析

如果取消单子句优先的策略限制，那么就是直接遍历子句集里面的子句，两两归结，步骤和花费时间会大幅增加（以1中第二个Kb为例子）：

加了单子句优先策略的限制只需要12步归结步骤，否则需要41步





# 参考资料

**参考文献：《人工智能》**