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	22336326	姓名	朱禹溪

一、 实验题目

第3周实验课作业

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：

- 输入为子句集， `KB` 子句中的每个元素是一阶逻辑公式(不含$\exists$,$\forall$等量词符号)
- 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）

例题：输入

```
KB = {(GradStudent(sue),),(~GradStudent(x),Student(x)),
(~Student(x),HardWorker(x)),(~HardWorker(sue),)}
```

则调用 `ResolutionFOL(KB)` 后返回推理过程的列表如下：

```
1 (GradStudent(sue),)
2 (~GradStudent(x),Student(x))
3 (~Student(x),HardWorker(x))
4 (~HardWorker(sue),)
5 R[1,2a]{x=sue} = (Student(sue),)
6 R[3a,5]{x=sue} = (HardWorker(sue),)
7 R[4,6] = []
```

二、 实验内容

1. 算法原理

逻辑子句转换为知识库:

输入的逻辑子句被解析为知识库（KB），KB 是一个包含逻辑子句的列表，每个逻辑子句又是一个包含原子句（predicates）的列表。

逻辑归结推理算法:

Judge(clause1, clause2)**函数检查两个子句是否可以进行变量替换，找出可替换的变量对。
resolve(KB, parent, assign, clause1_index, i, clause2_index, j, hash_result=None)**函数进行归结操作，尝试合一两个子句，生成新的子句。
mgu(KB, assign, parent)**函数遍历知识库中的所有子句，寻找可以归结的子句，并执行合一操作。
prun(n, KB, assign, parent)**函数对推导结果进行剪枝，去除重复或无效的推理步骤。
labeling(n, prunkb)**函数重新对推导结果中的子句进行标号，以便输出。
oput(n, kb, prunkb, newindex)**函数输出最终的推理结果，展示推理步骤和变量替换关系。
算法流程：
代码先解析输入的逻辑子句，构建知识库。然后通过归结推理算法尝试在知识库中寻找矛盾或可以推导的结论。最终，剪枝去除重复步骤，重新标记推理结果，输出推理步骤和替换关系。

2. 关键代码展示（可选）

```
def prun(n, KB, assign, parent):
    # 使用二叉树结构层序遍历剪枝
    prunkb = []
    q = queue.Queue()
    q.put(parent[-1])
    prunkb.append([KB[-1], parent[-1], assign[-1]])

    # 只有非知识库内的句子才会有变量替换
    while not q.empty():
        cur = q.get()
        # 大的先进队列(后推出来)，符合推理常理
        if cur[0] > cur[2]:
            if cur[0] >= n:
                prunkb.append([KB[cur[0]], parent[cur[0]], assign[cur[0]]])
                q.put(parent[cur[0]])
            if cur[2] >= n:
                prunkb.append([KB[cur[2]], parent[cur[2]], assign[cur[2]]])
                q.put(parent[cur[2]])
        else:
            if cur[2] >= n:
                prunkb.append([KB[cur[2]], parent[cur[2]], assign[cur[2]]])
                q.put(parent[cur[2]])
            if cur[0] >= n:
                prunkb.append([KB[cur[0]], parent[cur[0]], assign[cur[0]]])
                q.put(parent[cur[0]])
    return prunkb
```

```python
def mgu(KB, assign, parent):
    # 归结合一函数
    for clause1_index, clause1 in enumerate(KB):
        for clause2_index, clause2 in enumerate(KB):
            if clause1_index == clause2_index:
                continue
            for i, predicate1 in enumerate(clause1):
                for j, predicate2 in enumerate(clause2):
                    # 谓词相反
                    if (predicate1[0] == '~' + predicate2[0] or predicate2[0] == '~' + predicate1[0]) and len(predicate1) == len(predicate2):
                        if predicate1[1:] == predicate2[1:]:
                            if resolve(KB, parent, assign, clause1_index, i, clause2_index, j):
                                return
                        else:
                            hash_result = Judge(predicate1, predicate2)
                            if hash_result:
                                if resolve(KB, parent, assign, clause1_index, i, clause2_index, j, hash_result):
                                    return
```

三、 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）

Problem0:

```
1 (Student(sue),)
2 (~GradStudent(x),Student(x))
3 (~Student(x),HardWorker(x))
4 (~HardWorker(sue),)
5 R[1,3a]{x=sue} = (HardWorker(sue),)
6 R[4,5] = []
```

Problem1:

```
溪\Desktop\实验二_朱禹溪_22336326\src\problem1.py "
1 (ny),)
2 (A(mike),)
3 (A(john),)
4 (L(tony,rain),)
5 (L(tony,snow),)
6 (~A(x),S(x),C(x))
7 (~C(y),~L(y,rain))
8 (L(z,snow),~S(z))
9 (~L(tony,u),~L(mike,u))
10 (L(tony,v),L(mike,v))
11 (~A(w),~C(w),S(w))
12 R[2,6a]{x=mike} = (S(mike),C(mike))
13 R[2,11a]{w=mike} = (S(mike),~C(mike))
14 R[5,9a]{u=snow} = (~L(mike,snow),)
15 R[12b,13b] = (S(mike),)
16 R[14,8a]{z=mike} = (~S(mike),)
17 R[15,16] = []
```

Problem2:

```
1 (ony,mike),)
2 (On(mike,john),)
3 (Green(tony),)
4 (~Green(john),)
5 (~On(xx,yy),~Green(xx),Green(yy))
```

3. 评测指标展示及分析（机器学习实验必须有此项，其它可分析运行时间等）

逻辑子句转换为知识库：
在这一步骤中，将逻辑子句解析为知识库的复杂度取决于输入子句的长度和数量。假设有 m 个逻辑子句，每个子句平均有 n 个原子句，则这一步的复杂度为 O(m * n)。
逻辑归结推理算法：
Judge(clause1, clause2)：这个函数的复杂度取决于子句中出现的变量数量和结构，通常为 O(n)。
resolve(KB, parent, assign, clause1_index, i, clause2_index, j, hash_result=None)：这个函数对两个子句进行归结操作，在最坏情况下，需要遍历整个知识库。假设知识库中有 p 个子句，则这一步的复杂度为 O(p^n)，n 为变量数量。
unify(KB, assign, parent)：遍历知识库中所有子句进行合一操作，复杂度为 O(p * n)。
prun(n, KB, assign, parent)：对推导结果进行剪枝，复杂度取决于推理步骤的数量，通常为 O(k)，k 为步骤数量。

labeling(n, prunkb)：重新对推导结果中的子句进行标号，复杂度为 O(k)。

oput(n, kb, prunkb, newindex)：输出最终推理结果，复杂度通常为 O(k)。

对于代码方面似乎有一些 bug，在输出第一个元素的时候可能会输出不全，如果后续的归结未使用 1 则可以正常归结，感觉可能是由于初始分离逗号等操作产生的问题，后续还需要进一步修改。

四、 参考资料

合一算法的 Python 实现--人工智能_unify 算法-CSDN 博客

人工智能——归结推理-CSDN 博客

3-归结原理.pptx