《 智能系统 》实验报告

实验题目	推理机设计

一、实验目的

为实现十字路口红绿灯智能控制，本次实验的目的是：

（1） 了解产生式系统的推理原理与机制

（2） 设计并实现产生式系统的推理机

（3） 设计并实现支持可信度的产生式系统的推理方法（可选）

二、实验项目内容

1、设计并实现一个确定性推理机，要求支持尽可能复杂的产生式语法。

（1） 定义准备支持的产生式语法规范；

（2） 设计知识库的外部存储机制；

（3） 设计事实库的内部数据结构；

（4） 绘制推理机工作流程图；

（5） 编码实现推理机；

（6） 设计并实现相关软件界面；

（7） 推理机的运行调试；

2、设计并实现一个可信度推理机。（可选）。

只需要指出为了支持可信度，需要对上面的设计做那些改动。

三、实验过程或算法（代码）

（1） 定义准备支持的产生式语法规范；

基于实验二，设计出事实库和知识库如下：

```
事实库
1．秒数计数器达到周期
2．秒数计数器达到红绿灯转换时间
3．红绿灯转换批准
4．执行红绿灯转换时间调整函数
5．秒数计数器清零
6．南北车辆计数器清零
7．东西车辆计数器清零
8．南北灯为红，东西灯为绿
9．南北灯为绿，东西灯为红
```

报告创建时间：

```
10. 南北灯转绿，东西灯转红
11. 南北灯转红，东西灯转绿
12. 南北方有车辆经过
13. 东西方有车辆经过
14. 南北车辆计数器加 1
15. 东西车辆计数器加 1
16. 南北车辆比东西车辆多
17. 南北车辆比东西车辆少
18. 南北车辆远小于东西车辆
19. 南北车辆远大于东西车辆
20. 按照比例增加南北向绿灯时间
21. 按照比例减少南北向绿灯时间
22. 南北绿灯时间设置为 1
23. 南北绿灯时间设置为 period-1

知识库
1->3
2->3,4,5,6,7
3,8->10
3,9->11
12->14
13->15
4,16->20
4,17->21
4,18->22
4,19->23
```

（2） 设计知识库的外部存储机制；

知识库存储采取 RDF 数据模式，即主语谓语宾语三元组形式，但由于此次实验中主语和宾语间都为因果关系，所以省略了谓语。存储平台为华为云 GaussDB，存储格式如下：

	rid	reason	result
1	1	1	3
2	2	2	3,4,5,6,7
3	3	3,8	10
4	4	3,9	11
5	5	12	14
6	6	13	15
7	7	4,16	20
8	8	4,17	21
9	9	4,18	22
10	10	4,19	23

RDF 以三元组的形式描述资源，简洁明了，但是有着语义表达能力的缺陷。RDF 中没有定义类、属性等词汇。RDF 只能是对具体的事物

进行描述，缺乏抽象能力，无法对同一个类别的事物进行定义和描述。RDF 可以描述实体、实体的属性以及他们之间的关系，但是无法描述类与类之间的关系，类的属性等

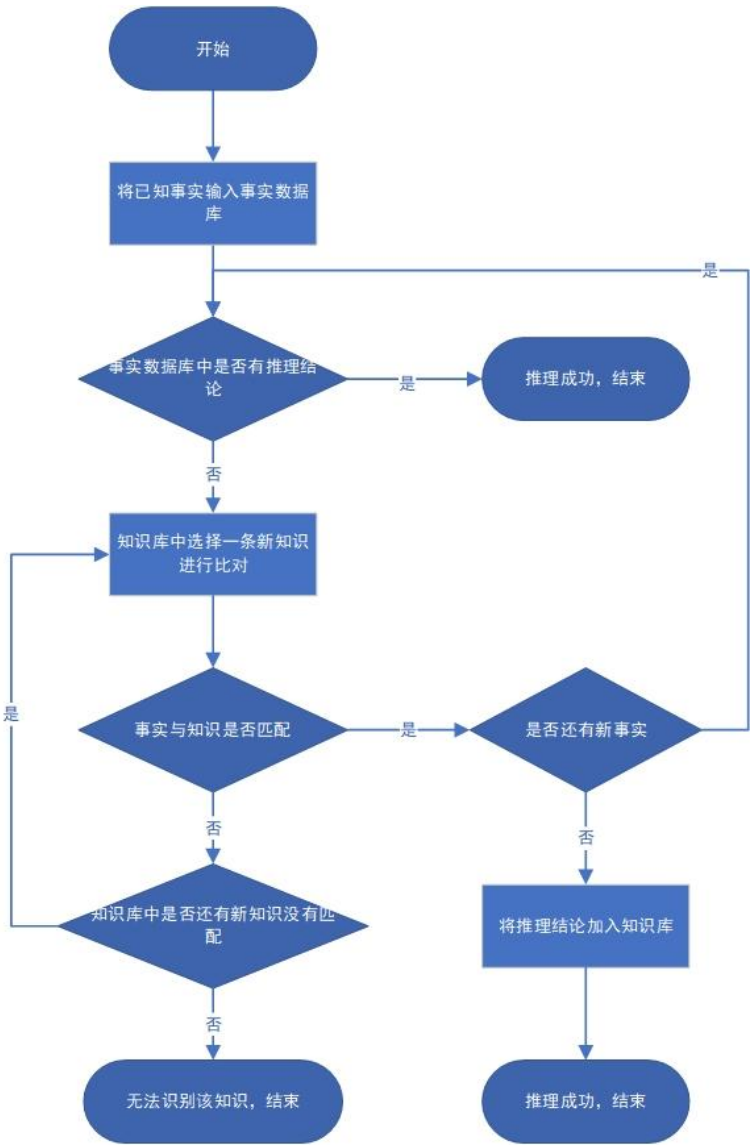
（3） 设计事实库的内部数据结构；

事实库存储平台为华为云 GaussDB，存储数据结构为：

	fid	fact
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
10	10	
11	11	

（4） 绘制推理机工作流程图；

正向推理流程图如下：



（5） 编码实现推理机；

```
#自定义去重函数
def judge_repeat(self, list1):
    tmp = set(list1)
    list2 = list(tmp)
    list2.sort()
    return list2

def resultword(self):
    # 存储前提条件
    global list_real
```

```
        # 存储结果
        global result
        # 存储推理过程
        global process
        # 默认结果
        result=[]
        process=[]
        for i in self.rule:
            # 将规则库中的前提条件分割成列表
            Reason = i[1].split(",")
            # 将规则库中的结果分割成列表
            Result = i[2].split(",")
            # 判断规则库中的前提条件是否是选择的前提条件的子集
            if set(Reason).issubset(set(list_real)):
                list_real = list_real + Result
                result    = result + Result
                process.append(i[1]+"->"+i[2])
                list_real = self.judge_repeat(list_real)
                result = self.judge_repeat(result)

        # 更新 gui 对应文字
        for i in process:
            self.message2['text']=self.message2['text']+"
"+str(i)+"\n"
        self.message2['text']=self.message2['text']+"最终
结果为："+'\n'
        for i in result:
            tmp = self.fact[int(i)-1][1]
            self.message2['text']=self.message2['text']+"
"+tmp+"\n"
```

对于推理机的实现，由于知识库的存储是有序的，即前提条件按照事实库编号排序，这样只需一次遍历即可完成所有推理。由于顺序遍历，前面推理所得到结果会成为后续推理的前提条件，如果未满足，重复多次遍历也无法完成推理。

（6） 设计并实现相关软件界面；

```
    def __init__(self,window):
        #知识库
        self.fact=self.getResult("select * from fact")
        #事实库
```

```
        self.rule=self.getResult("select * from rule")
        self.win=window
        self.win.title("产生式系统")
        width=1200
        height=600
        align_str='%dx%d'%(width,height)
        window.geometry(align_str)
        #treeview 进行左表格显示
        self.tree=ttk.Treeview(height=31,column=("#0","#1"
))
        self.tree.grid(row=0,column=0,columnspan=1)
        self.tree.heading("#0",text="编号",anchor=CENTER)
        self.tree.heading("#1",text="事实",anchor=CENTER)

        #绘制表格
        records=self.tree.get_children()
        for element in records:
            self.tree.delete(element)
        query="select * from fact"
        db_rows=self.getResult(query)
        db_rows.reverse()
        #填充表格
        for row in db_rows:
            self.tree.insert("",0,text=row[0],values=(row[
1]))
        global list_real
        list_real=[]
        ttk.Button(text='添加
—>',command=self.add).grid(row=0, column=3, sticky=W + E)
        self.message = Label(text='')
        self.message.grid(row=0, column=5,
columnspan=5,sticky=W+E)
        self.message['text']="您的选择是：\n"
        self.message2 = Label(text=' ')
        self.message2.grid(row=0, column=20,
columnspan=5,sticky=W+E)
        self.message2['text']="推理过程如下：\n"
        ttk.Button(text='结果
—>',command=self.resultword).grid(row=0, column=12,
```

```
sticky=W + E)
        ttk.Button(text='清空
',command=self.clear).grid(row=0, column=30, sticky=W + E)

    def clear(self):
        self.message['text']="您的选择是：\n"
        self.message2['text']="推理过程如下：\n"
        global list_real
        list_real=[]
        global result
        result=[]

    #数据添加处理
    def add(self):
        global list_real
        list_real.append(str(self.tree.item(self.tree.sele
ction())['text']))
        print("lastadd",list_real)
        self.message['text']=self.message['text']+'\n'+str
(self.tree.item(self.tree.selection())['text'])+str(self.
tree.item(self.tree.selection())['values'][0])
        self.message.grid(row=0, column=4,
columnspan=5,sticky=W+E)
```

最后 UI 界面如下：

（7） 推理机的运行调试；

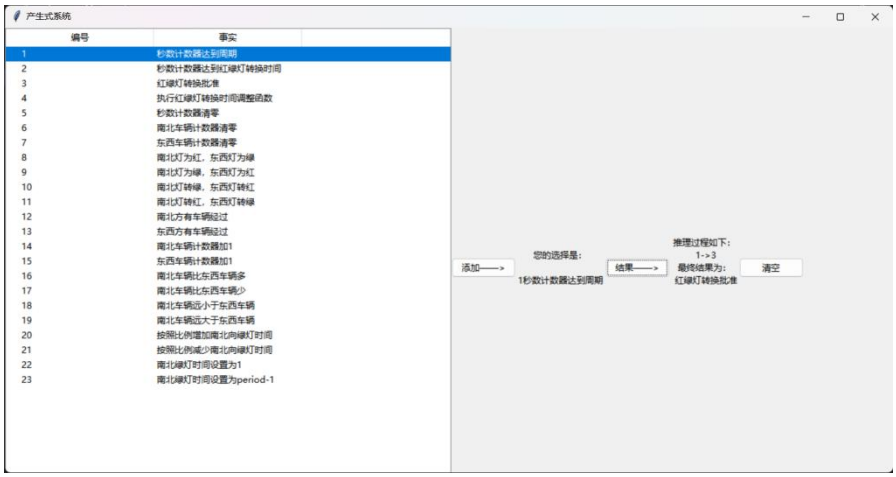
- 清空按钮未实际清除数据，是因为仅清除了 GUI 界面显示的文字，但实际存储的数据并没有清除，所以导致了数据的叠加。
- 结果按钮会累计上次的结果，本质上和上一条一致，每次点击结果按钮需手动清楚结果数据和 gui 文字。

- 由于设计问题，16，17，18，19 号事实不应该同时出现，但因为已经完成存储，所以就添加逻辑代码手动规避这种情况

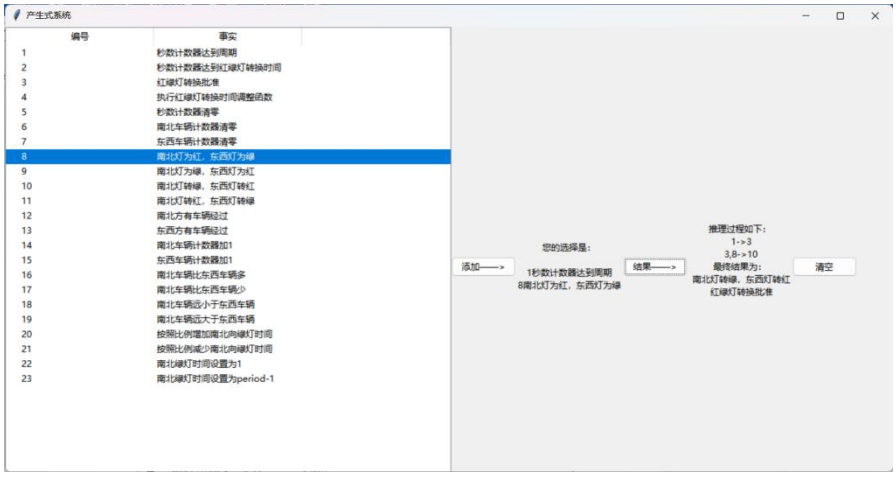
四、实验结果及分析

要求：给出测试用例、测试结果和测试分析。

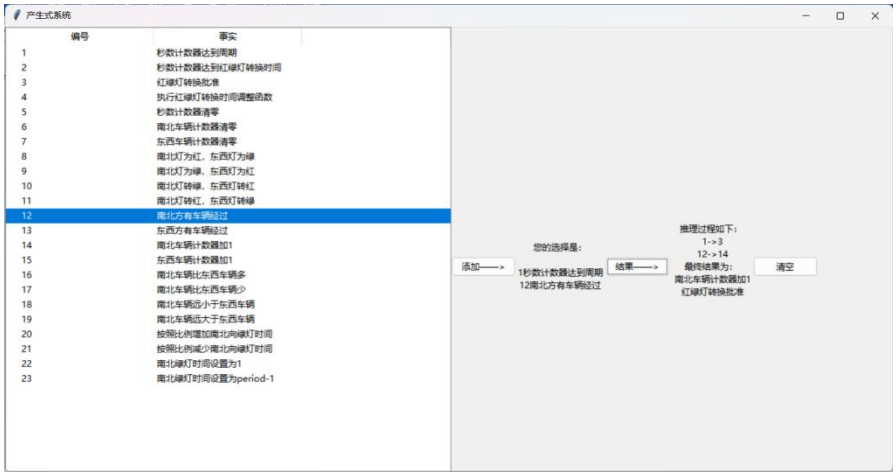
- 测试用例 1　1　　→3



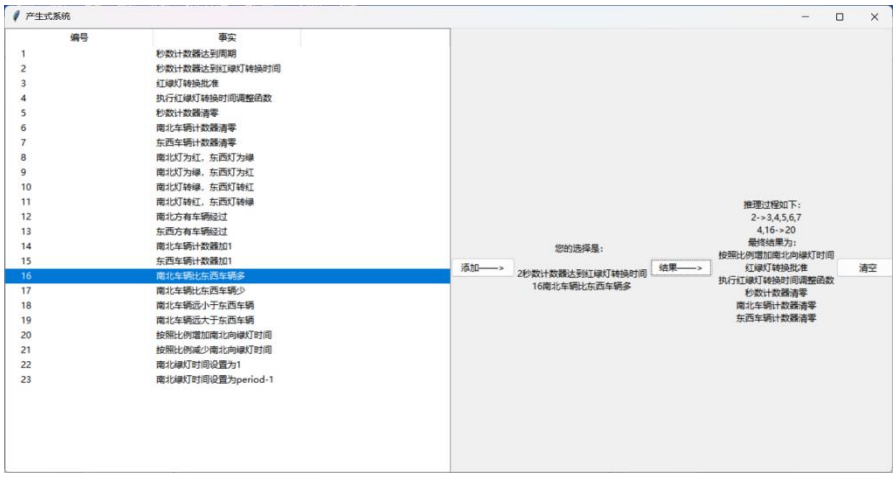
- 测试用例 2　1,8　　→3,10



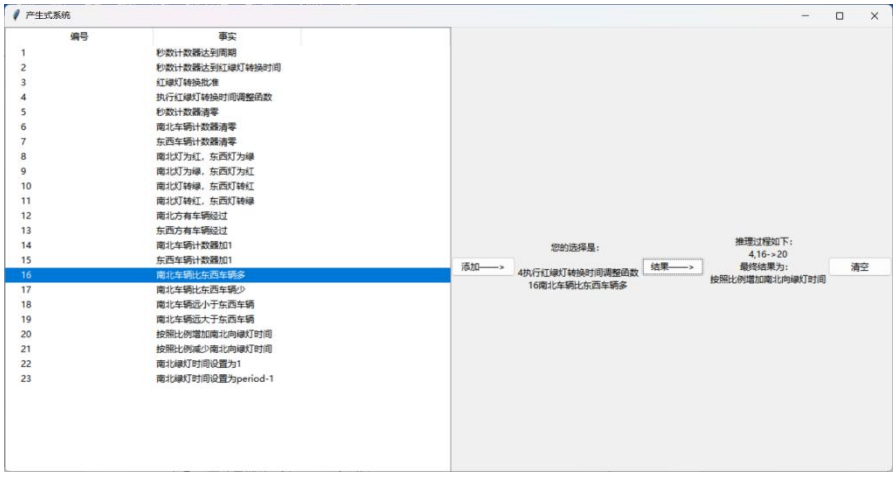
- 测试用例 3　1,12　　→3,14



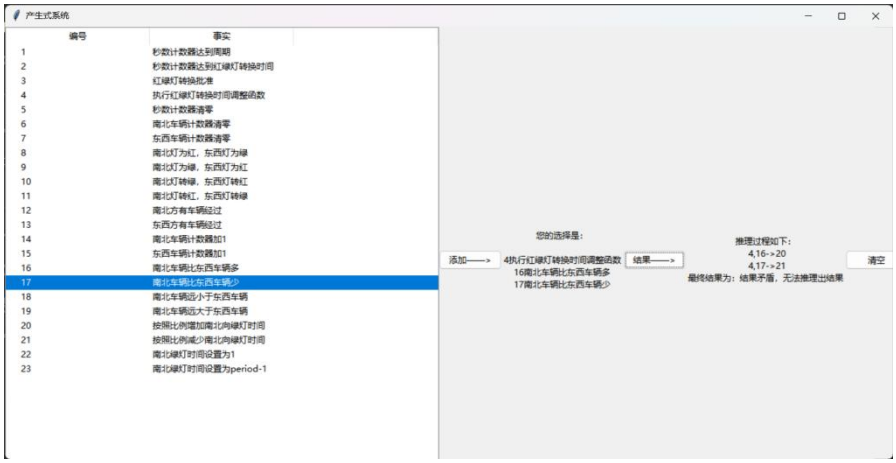
- 测试用例 4　2,16　　→3,4,5,6,7,20



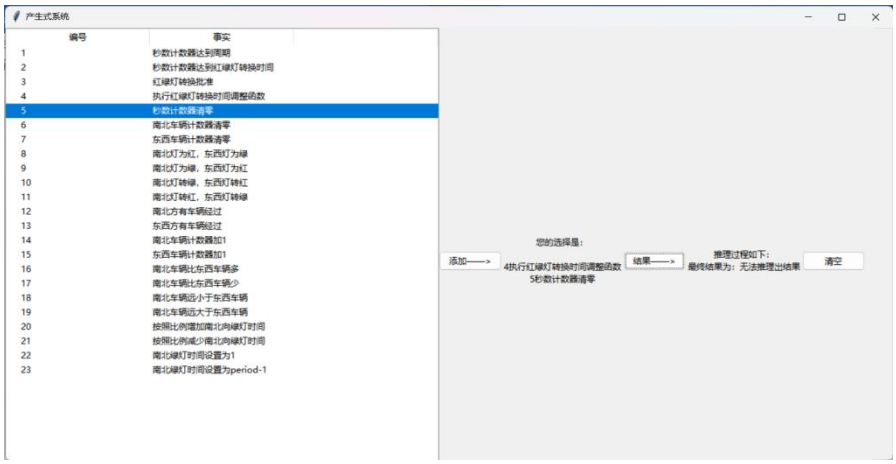
- 测试用例 5　4,16　　→20



- 测试用例 6　4,16,17　　→null



- 测试用例 7　4,5　　→null



经上述测试，推理运行正常，其中用例 6 说明，其中某一个事实不能同时出现，否则导致结果矛盾，用例 7 说明，仅有结果没有前提条件也无法推理出结果。

五、完成时间

（1）实验时间：2023.5.14，2023.5.21

（2）检查时间：2023.6.4

（3）2023 年 6 月 18 日 23:59 之前提交实验报告