華東理工大學

EAST CHINA UNIVERSITY OF SCIENCE AND TECHNOLOGY

《 人工智能 》

实验报告本

班　级：　计 203

学　号：　20002462

姓　名：　刘子言

指导教师：　陈志华

信息科学与工程学院

2022 年 12 月

《人工智能》实 验 报 告 一

实验 1 名称：产生式系统实验	**实验地点**：信息楼 215

所使用的工具软件及环境：

装有 Windows 操作系统、Python 版本：Python 3 及以上

一、实验目的

熟悉一阶谓词逻辑和产生式表示法，掌握产生式系统的运行机制，以及基于规则推理的基本方法。

二、实验内容

运用所学知识，设计并编程实现产生式系统，输入相应的条件，根据知识库推理得出相应的知识

三、实验要求

1.具体应用领域自选，具体系统名称自定。

2.用产生式规则作为知识表示，利用产生式系统实验程序，建立知识库，分别运行正、反向推理。

四、实验步骤

1、系统设计

本实验设计了一个有关动物识别的专家系统，采用 Python3.8 编程实现。通过输入一定的条件，可逐步推理出相应的结果。本产生式系统最终涉及的动物共有 7 种。

2、知识库设计

（1）系统包含的动物特征（共 27 种）及编号

1:有毛发；2:产奶；3:有羽毛；4:会飞；5:会下蛋；6:吃肉；7:有犬齿；8:有爪；9:眼盯前方；10:有蹄；11:反刍；12:黄褐色；13:有斑点；14:有黑色条纹；15:长脖；16:长腿；17:不会飞；18:会游泳；19:黑白二色；20:善飞；21：哺乳类；22:鸟类；23:食肉类；24:蹄类；32:吃草；33:食草类；34:杂食类

（2）系统可推理出的动物种类（共 7 种）及编号

25:金钱豹；26:虎；27:长颈鹿；28:斑马；29:鸵鸟；30:企鹅；31:信天翁

（3）原规则库

根据这些动物识别的专家知识建立如下规则库：

r1: IF 该动物有毛发 THEN 该动物是哺乳动物
r2: IF 该动物有奶 THEN 该动物是哺乳动物
r3: IF 该动物有羽毛 THEN 该动物是鸟类
r4: IF 该动物会飞 AND 会下蛋 THEN 该动物是鸟类
r5: IF 该动物吃肉 AND 不吃草 THEN 该动物是食肉动物
r6: IF 该动物有犬齿 AND 有爪 AND 眼盯前方 THEN 该动物是食肉动物
r7: IF 该动物吃草 AND 不吃肉 THEN 该动物是食草动物
r8: IF 该动物吃肉 AND 吃草 THEN 该动物是杂食动物
r9: IF 该动物是哺乳动物 AND 有蹄 THEN 该动物是有蹄类动物
r10: IF 该动物是哺乳动物 AND 是反刍动物 THEN 该动物是有蹄类动物
r11: IF 该动物是哺乳动物 AND 是食肉动物 AND 是黄褐色 AND 身上有暗斑点
THEN 该动物是金钱豹
r12: IF 该动物是哺乳动物 AND 是食肉动物 AND 是黄褐色 AND 身上有黑色条纹
THEN 该动物是虎
r13: IF 该动物是有蹄类动物 AND 有长脖子 AND 有长腿 AND 身上有暗斑点
THEN 该动物是长颈鹿
r14: IF 该动物是有蹄类动物 AND 身上有黑色条纹 THEN 该动物是斑马
r15: IF 该动物是鸟类 AND 有长脖子 AND 有长腿 AND 不会飞 AND 有黑白二色
THEN 该动物是鸵鸟
r16: IF 该动物是鸟类 AND 会游泳 AND 不会飞 AND 有黑白二色
THEN 该动物是企鹅
r17: IF 该动物是鸟类 AND 善飞 THEN 信天翁

（4）编号表示的规则库

将这些规则转换为编号表示：

r1: IF 1 THEN 21
r2: IF 2 THEN 21
r3: IF 3 THEN 22
r4: IF 4 AND 5 THEN 22
r5: IF 6 AND not 32 THEN 23
r6: IF 7 AND 8 AND 9 THEN 23
r7: IF 32 AND not 6 THEN 33
r8: IF 6 AND 32 THEN 34
r9: IF 21 AND 10 THEN 24
r10: IF 21 AND 11 THEN 24
r11: IF 21 AND 23 AND 12 AND 13 THEN 25
r12: IF 21 AND 23 AND 12 AND 14 THEN 26
r13: IF 24 AND 15 AND 16 AND 13 THEN 27
r14: IF 24 AND 14 THEN 28
r15: IF 22 AND 15 AND 16 AND 17 AND 19 THEN 29
r16: IF 22 AND 18 AND 17 AND 19 THEN 30
r17: IF 22 AND 20 THEN 31

3、推理的基本思路

首先根据一些比较简单的条件，如“有毛发”“有羽毛”“会飞”等对动物进行比较粗的分类，如“哺乳动物”“鸟类”等；然后随着条件的增加，逐步缩小分类范围，最后得出识别的七种动物。

这样推理的好处有：一是当已知的事实不完全时，虽不能推出最终结论，但可以得到分类结果；二是当需要增加对其他动物的识别时，规则库中只需增加关于这些动物个性方面的知识，如 r11 至 r17 那样，而对 r1 至 r10 可直接利用，这样增加的规则就不会太多。

上述的本产生式系统的推理过程是一个不断地从规则库中选择可用规则与综合数据库中的已知事实进行匹配的过程，规则的每一次成功匹配都会使综合数据库增加新的内容；随着综合数据库的不断扩充，朝着问题的解决方向前进一步。这种推理的过程，是专家系统中的核心内容。

4、可视化界面的展示

界面标题为“动物识别系统”；

界面左侧展示系统包含的所有条件及对应编号，并说明系统的使用规则；

界面右侧允许用户输入所选的条件，并以数字 0 结束，然后点击下方的“开始按钮”；

推理分为三部分展示：前提条件，推理过程，推理结果；

点击取消按钮，可以清空界面的输入输出框，然后输入新的条件重新开始新的推理。

5、动物识别系统运行截图

（学号标注在窗口名后：产生式系统_20002462）

（1）输入条件为 1,2,10，推理得到有蹄类，但无法判断为何种动物：

（2）输入条件为 2,10,14，推理得到该动物为斑马：

（3）输入条件为 6,32，推理得到杂食类，但无法判断为何种动物：

（4）输入条件为 32，推理得到食草类，但无法判断为何种动物：

五、 程序设计的核心代码

1、产生式系统基于规则库的推理代码

```
# 自定义函数，判断有无重复元素
def judge_repeat(value, list=[]):
    for i in range(0, len(list)):
        if list[i] == value:
            return 1
        else:
            if i != len(list) - 1:
                continue
            else:
                return 0


# 自定义函数，对已经整理好的综合数据库real_list进行最终的结果判断
reasoning_result = []
def judge_last(list):
    for i in list:
```

```
        if i == '23':  # 食肉类
            for i in list:
                if i == '12':  # 黄褐色
                    for i in list:
                        if i == '21':  # 哺乳类
                            for i in list:
                                if i == '13':  # 有斑点
                                    reasoning_result.append("黄褐色，有斑点,哺乳
类，食肉类->金钱豹\n")
                                    reasoning_result.append("所识别的动物为金钱
豹")
                                    return 0
                                elif i == '14':  # 有黑色条纹
                                    reasoning_result.append("黄褐色，有黑色条纹，
哺乳类，食肉类->虎\n")
                                    reasoning_result.append("所识别的动物为虎")
                                    return 0
        elif i == '14':  # 有黑色条纹
            for i in list:
                if i == '24':  # 蹄类
                    reasoning_result.append("有黑色条纹，蹄类->斑马\n")
                    reasoning_result.append("所识别的动物为斑马")
                    return 0
        elif i == '24':  # 蹄类
            for i in list:
                if i == '13':  # 有斑点
                    for i in list:
                        if i == '15':  # 长脖
                            for i in list:
                                if i == '16':  # 长腿
                                    reasoning_result.append("有斑点，有黑色条纹，
长脖，蹄类->长颈鹿\n")
                                    reasoning_result.append("所识别的动物为长颈
鹿")
                                    return 0
        elif i == '20':  # 善飞
            for i in list:
                if i == '22':  # 鸟类
                    reasoning_result.append("善飞，鸟类->信天翁\n")
                    reasoning_result.append("所识别的动物为信天翁")
                    return 0
        elif i == '22':  # 鸟类
            for i in list:
                if i == '4':  # 不会飞
```

```
                        for i in list:
                            if i == '15':  # 长脖
                                for i in list:
                                    if i == '16':  # 长腿
                                        reasoning_result.append("不会飞，长脖，长腿，
鸟类->鸵鸟\n")
                                        reasoning_result.append("所识别的动物为鸵鸟
")
                                        return 0

        elif i == '4':  # 不会飞
            for i in list:
                if i == '22':  # 鸟类
                    for i in list:
                        if i == '18':  # 会游泳
                            for i in list:
                                if i == '19':  # 黑白二色
                                    reasoning_result.append("不会飞，会游泳，黑白
二色，鸟类->企鹅\n")
                                    reasoning_result.append("所识别的动物企鹅")
                                    return 0
        else:
            if list.index(i) < len(list) - 2:
                continue
            else:
                reasoning_result.append("根据所给条件无法判断为何种动物")
                break
```

2、由前提条件推理得到中间过程并扩充综合数据库

```
# 保存、显示、遍历前提条件，产生中间过程，扩充综合数据库
list_real = []
def conditionIN():
    # 保存你输入的条件数字
    num_real_all = the_condition_you_choose.get(1.0, tk.END)
    m = 0
    while 1:
        # 循环输入前提条件所对应的字典中的键
        num_real = num_real_all[m]
        num_real_next = num_real_all[m + 1]
        if num_real == '0':
            break
        if num_real != '\n' and num_real_next == '\n':
            list_real.append(num_real)
            m = m + 1
```

```
        elif num_real != '\n' and num_real_next != '\n':
            num_real = num_real + num_real_next
            list_real.append(num_real)
            m = m + 2
        else:
            m = m + 1
    print(list_real)

    # 转换并输出前提条件的文字
    condition_real = []
    for i in range(0, len(list_real)):
        condition_real.append(dict_before[list_real[i]])
    the_condition_show.delete(0, tk.END)
    the_condition_show.insert(0, condition_real)

    # 遍历综合数据库list_real中的前提条件，得出中间结果
    reasoning_process = []
    for i in list_real:
        if i == '1':
            if judge_repeat('21', list_real) == 0:
                list_real.append('21')
                reasoning_process.append("有毛发->哺乳类\n")
        elif i == '2':
            if judge_repeat('21', list_real) == 0:
                list_real.append('21')
                reasoning_process.append("产奶->哺乳类\n")
        elif i == '3':
            if judge_repeat('22', list_real) == 0:
                list_real.append('22')
                reasoning_process.append("有羽毛->鸟类\n")
        else:
            if list_real.index(i) != len(list_real) - 1:
                continue
            else:
                break
    for i in list_real:
        if i == '4':
            for i in list_real:
                if i == '5':
                    if judge_repeat('22', list_real) == 0:
                        list_real.append('22')
                        reasoning_process.append("会飞，会下蛋->鸟类\n")
        elif i == '6':
            for i in list_real:
```

```
                    if i == '21':
                        if judge_repeat('21', list_real) == 0:
                            list_real.append('21')
                            reasoning_process.append("食肉->哺乳类\n")
            elif i == '32':   # 以下为新增的部分
                for i in list_real:
                    flag = 1
                    for j in list_real:
                        if j == '6':
                            flag = 0
                            break
                    if judge_repeat('33', list_real) == 0 and flag:
                        list_real.append('33')
                        reasoning_process.append("食草->食草类\n")
                    elif judge_repeat('34', list_real) == 0 and flag == 0:
                        list_real.append('34')
                        reasoning_process.append("食肉，食草->杂食类\n")
            elif i == '7':
                for i in list_real:
                    if i == '8':
                        for i in list_real:
                            if i == '9':
                                if judge_repeat('23', list_real) == 0:
                                    list_real.append('23')
                                    reasoning_process.append("有犬齿,有爪,眼盯前方->
食肉类\n")
            elif i == '10':
                for i in list_real:
                    if i == '21':
                        if judge_repeat('24', list_real) == 0:
                            list_real.append('24')
                            reasoning_process.append("有蹄，哺乳类->蹄类\n")
            elif i == '11':
                for i in list_real:
                    if i == '21':
                        if judge_repeat('24', list_real) == 0:
                            list_real.append('24')
                            reasoning_process.append("反刍，哺乳类->哺乳类\n")

            else:
                if i != len(list_real) - 1:
                    continue
                else:
                    break
```

```
    show_reasoning_process.delete(1.0, tk.END)
    for i in reasoning_process:
        show_reasoning_process.insert(tk.END, i)

    # 结果判断
    judge_last(list_real)
    show_reasoning_result.delete(1.0, tk.END)
    for i in reasoning_result:
        show_reasoning_result.insert(tk.END, i)
```

3、“取消”按钮清空再开始

```
# 取消函数
def conditionOUT():
    the_condition_you_choose.delete(1.0, tk.END)
    list_real.clear()
    reasoning_result.clear()
```

4、GUI界面设计的核心代码

```
# GUI标题
label1 = tk.Label(root, text="动物识别系统", fg="#DAA520", bg="#FFEBCD")
label1.place(width=200, height=30, x=280, y=20)  # 注：用label.pack()会使该布局无效

# 系统给出的条件
label2 = tk.Label(root, text="以下为本系统给出的可供选择的条件：" + '\n' + rule,
fg="#DAA520")
label2.place(x=20, y=50, width=300, height=600)

# 输入你的前提条件
label3 = tk.Label(root, text="请输入选择的条件：", fg="#DAA520")
label3.place(x=350, y=100, width=150, height=30)
the_condition_you_choose = tk.Text(root, width=20, height=30)
the_condition_you_choose.pack(padx=5, pady=10)
the_condition_you_choose.place(x=500, y=100, width=170, height=60)

# “开始”按钮
button1 = tk.Button(root, text="开始", command=conditionIN,
activebackground="#2082AA", fg="#DAA520", bg="#FFEBCD")
button1.place(x=450, y=180, width=100, height=30)

# “取消”按钮
button2 = tk.Button(root, text="取消", command=conditionOUT,
activebackground="#2082AA", fg="#DAA520", bg="#FFEBCD")
button2.place(x=600, y=180, width=100, height=30)
```

```
# 显示前提条件
Label4 = tk.Label(root, text="前提条件为：", fg="#DAA520")
Label4.place(x=350, y=240, width=150, height=30)
the_condition_show = tk.Entry(root, width=20, font=('宋体', '10'))
the_condition_show.place(x=470, y=250, width=200, height=50)

# 显示推理过程
Label5 = tk.Label(root, text="推理过程如下：", fg="#DAA520")
Label5.place(x=350, y=340, width=150, height=30)
show_reasoning_process = tk.Text(root, width=20, font=('宋体', '10'))
show_reasoning_process.place(x=470, y=350, width=200, height=80)

# 显示推理结果
Label6 = tk.Label(root, text="推理结果如下：", fg="#DAA520")
Label6.place(x=350, y=440, width=150, height=30)
show_reasoning_result = tk.Text(root, width=20, font=('宋体', '10'))
show_reasoning_result.place(x=470, y=450, width=200, height=50)
```

六、实验体会

通过本次实验，我对产生式系统的基本结构及推理原理有了更深的了解。在熟知一阶谓词逻辑和产生式表示法的基础之上，我掌握了产生式系统的运行机制，以及其基于规则推理的基本方法，并且能够通过编程，实现一个具体的产生式系统——动物识别系统的推理逻辑及可视化展示，收获颇丰。

结合本次实验和已有的知识还可以体会到，产生式系统有自身的优缺点：

产生式系统优点在于：

（1）自然性。产生式表示法用“如果……，则……”的形式表示知识，这是人们常用的一种表达因果关系的知识表示形式，既直观、自然，又便于进行推理。正是由于这一原因，才使得产生式表示法成为人工智能中最重要且应用最多的一种知识表示方法；

（2）模块性。产生式是规则库中最基本的知识单元，它们同推理机构相对独立，而且每条规则都具有相同的形式。这就便于对其进行模块化处理，为知识的增、删、改带来了方便，为规则库的建立和扩展提供了可管理性；

（3）有效性。产生式表示法既可表示确定性知识，又可表示不确定性知识；既有利于表示启发式知识，又可方便地表示过程性知识。目前已建造成功的专家系统大部分是用产生式来表达其过程性知识的；

（4）清晰性。产生式有固定的格式。每一条产生式规则都由前提与结论（操作）这两部分组成，而且每一部分所含的知识量都比较少。这既便于对规则进行设计，又易于对规则库中知识的一致性及完整性进行检测。

产生式系统的不足之处：

（1）效率不高。这一点在本系统中也有所体现，在产生式系统求解问题的过程中，

首先要用产生式的前提部分与综合数据库中的已知事实进行匹配，从规则库中选出可用的规则，此时选出的规则可能不止一个，这就需要按一定的策略进行“冲突消解”，然后把选中的规则启动执行。因此，产生式系统求解问题的过程是一个反复进行“匹配一冲突消解一执行”的过程。鉴于规则库一般都比较庞大，而匹配又是一件十分费时的工作，因此其工作效率不高，而且大量的产生式规则容易引起组合爆炸。

（2）不能表达具有结构性的知识。产生式适合于表达具有因果关系的过程性知识，是一种非结构化的知识表示方法，所以对具有结构关系的知识却无能为力，它不能把具有结构关系的事物间的区别与联系表示出来。

七、教师评语

该学生_____完成了实验任务，算法设计_____，实验结果_____，实验体会_____。

因此总体评价为_____。

教师签字：

年　月　日