2024 年秋季《人工智能导论》

实验一

动物识别系统

实验报告

学院： 自动化学院

专业： 智能科学与技术

学号： 23061922

姓名： 李锦瑄

2024 年 10 月

题目：

基于产生式规则的动物识别系统——识别虎、金钱豹、斑马、长颈鹿、鸵鸟、企鹅、信天翁等七种动物的产生式系统。

一、实验目的

熟悉产生式表示法。

掌握产生式系统的运行机制。

掌握基于规则推理的基本方法。

二、实验内容

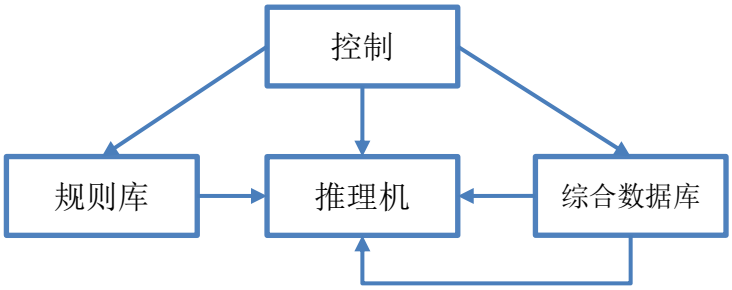
编程实现一个小型产生式系统，以动物识别系统为例。

建立规则库和综合数据库。

设计思路：正向推理/逆向推理。

三、实验原理

（一）产生式系统框图



规则库: 用于描述相应领域内知识的产生式集合。

综合数据库(事实库、上下文等)：一个用于存放问题求解过程中各种当前信息的数据结构。

控制系统（推理机构）：由一组程序组成，负责整个产生式系统的运行，实现对问题的求解。

（二）推理的方向

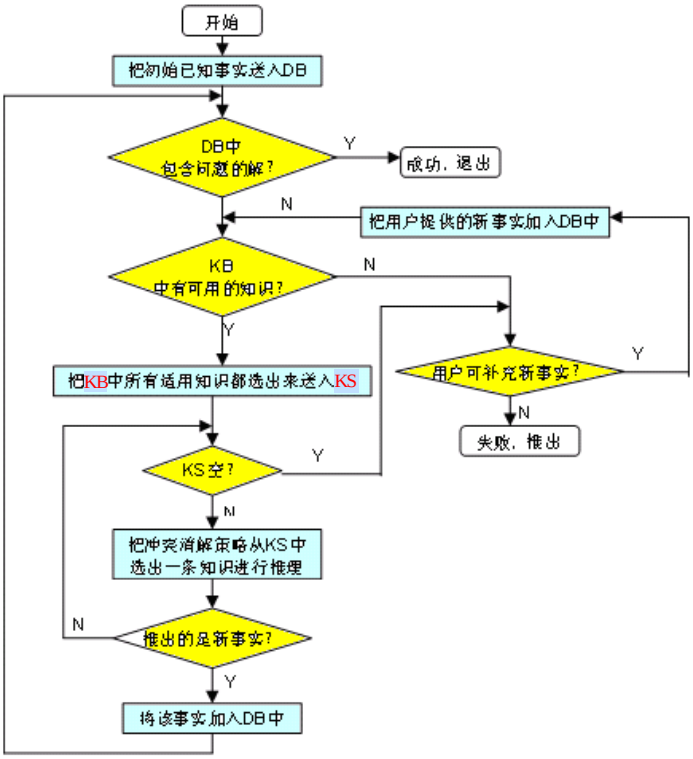
1、**正向推理（事实驱动推理）**：已知事实→结论

基本思想：

（1）从初始已知事实出发，在知识库 *KB* 中找出当前可适用的知识，构成可适用知识集 *KS*。

（2）按某种冲突消解策略从 *KS* 中选出一条知识进行推理，并将推出的新事实加入到数据库 *DB* 中作为下一步推理的已知事实，再在 *KB* 中选取可适用知识构成 *KS*。

（3）重复（2），直到求得问题的解或 *KB* 中再无可适用的知识。

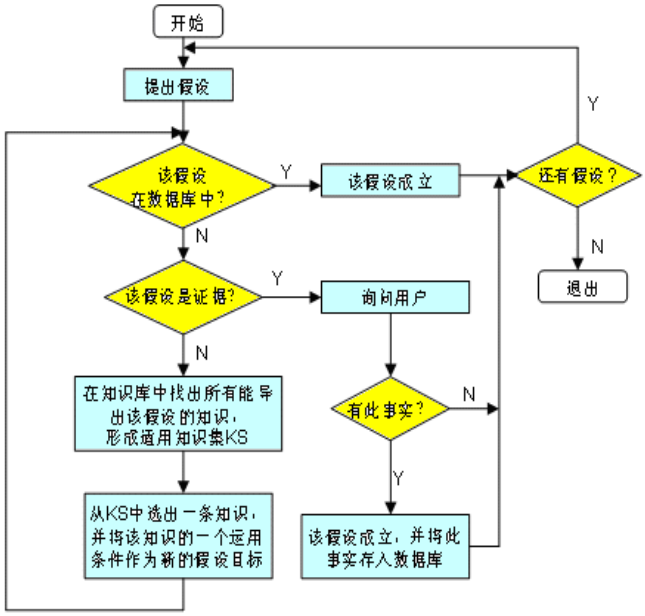


2、**逆向推理（目标驱动推理）**：以某个假设目标作为出发点。

基本思想：选定一个假设目标；寻找支持该假设的证据，若所需的证据都能找到，则原假设成立；若无论如何都找不到所需要的证据，说明原假设不成立的；为此需要另作新的假设。

主要优点：不必使用与目标无关的知识，目的性强，同时它还有利于向用户提供解释。

主要缺点：起始目标的选择有盲目性。



四、实验算法

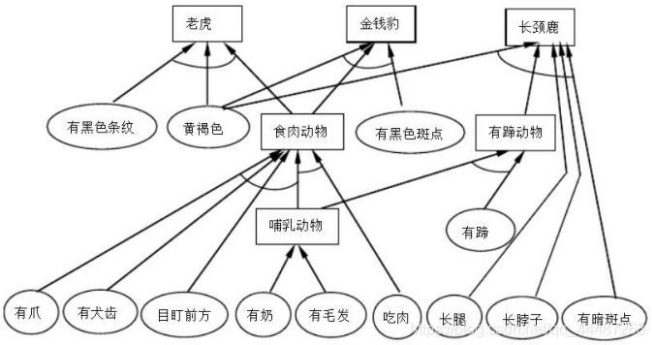
（一）规则库建立

运用以下规则，设计实现一个小型动物识别系统。

R1： if 动物有毛发 then 动物是哺乳动物
R2： if 动物有奶 then 动物是哺乳动物
R3： if 动物有羽毛 then 动物是鸟
R4： if 动物会飞 and 会生蛋 then 动物是鸟
R5： if 动物吃肉 then 动物是食肉动物
R6： if 动物有犀利牙齿 and 有爪 and 眼向前方 then 动物是食肉动物
R7： if 动物是哺乳动物 and 有蹄 then 动物是有蹄类动物
R8： if 动物是哺乳动物 and 反刍 then 动物是有蹄类动物
R9： if 动物是哺乳动物 and 是食肉动物 and 有黄褐色 and 有暗斑点 then 动物是豹
R10：if 动物是哺乳动物 and 是食肉动物 and 有黄褐色 and 有黑色条纹 then 动物是虎
R11：if 动物是有蹄类动物 and 有长脖子 and 有长腿 and 有暗斑点 then 动物是长颈鹿
R12：if 动物是有蹄类动物 and 有黑色条纹 then 动物是斑马
R13：if 动物是鸟 and 不会飞 and 有长脖子 and 有长腿 and 有黑白二色 then 动物是鸵鸟
R14：if 动物是鸟 and 不会飞 and 会游泳 and 有黑白二色 then 动物是企鹅
R15：if 动物是鸟 and 善飞 then 动物是信天翁

根据规则，建立知识库文件 knowledge.txt，内容及推理网络如下图：

```
有毛发 哺乳动物
有奶 哺乳动物
有羽毛 鸟
会飞 下蛋 鸟
吃肉 食肉动物
有犬齿 有爪 眼盯前方 食肉动物
哺乳动物 有蹄 有蹄类动物
哺乳动物 嚼反刍动物 有蹄类动物
哺乳动物 食肉动物 黄褐色 暗斑点 金钱豹
哺乳动物 食肉动物 黄褐色 黑色条纹 虎
有蹄类动物 长脖子 长腿 暗斑点 长颈鹿
有蹄类动物 黑色条纹 斑马
鸟 长脖子 长腿 黑白二色 不飞 鸵鸟
鸟 会游泳 不飞 黑白二色 企鹅
鸟 善飞 信天翁
```



（二）实验原理

一个基于产生式规则专家系统的完整结构如图。其中，知识库、推理机和工作存储器是构成专家系统的核心。系统的主要部分是知识库和推理机制，知识库由谓词演算事实和有关讨论主题的规则构成，推理机制由所有操纵知识库来演绎用户要求的信息的过程构成。

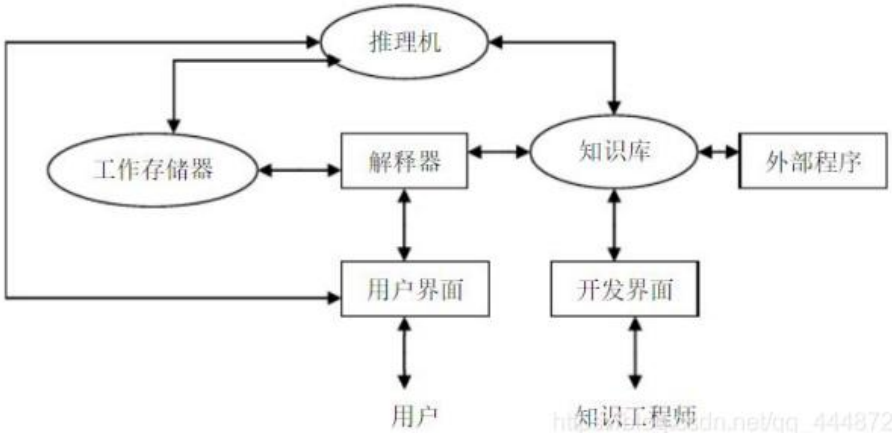


图 一个基于产生式规则的专家系统的完整结构

（三）绘制图形化界面

本实验将使用 Python 中的 **PyQt5 模块**设计界面，首先引入模块：

```
from PyQt5 import QtCore, QtGui, QtWidgets
```

QtCore：包含核心非图形功能，如事件循环、时间、文件处理、线程等。

QtGui：主要提供图形界面的基本功能，如绘制、图像处理、字体、颜色等。

QWidget：所有 UI 控件的基类，包含按钮控件、标签控件、文本输入框、下拉框、表格控件、对话框控件、主窗口控件等。

（四）具体实施

（1）首先将规则库里面的规则进行相应的排序和数据处理，以便之后的推理信息的匹配，此处将规则信息进行相应的拓扑排序，代码如下：

```
inn = []
for i in P:
    sum = 0
    for x in i:
        if Q.count(x) > 0:   # 能找到，那么
            sum += Q.count(x)
    inn.append(sum)
while (1):
    x = 0
    if inn.count(-1) == inn.__len__():
        break
    for i in inn:
        if i == 0:
            str = ' '.join(P[x])
            ans = ans + str + " " + Q[x] + "\n"   # 写入结果
            inn[x] = -1
            # 更新入度
            y = 0
            for j in P:
                if j.count(Q[x]) == 1:
                    inn[y] -= 1
                y += 1
        x += 1
```

（2）整理好规则库之后，进行推理功能的编写，根据相应的规则匹配模式，对用户输入的需要推理的信息进行相应的判断，然后一步步的询问用户相关的规则信息，进行进一步的推理，直到完全匹配出推理信息。代码如下：

```
def go(self, flag=True):
    self.Q = []
    self.P = []
    fo = open('knowledge.txt', 'r', encoding='utf-8')
    for line in fo:
        line = line.strip('\n')
        if line == '':
            continue
        line = line.split(' ')
        self.Q.append(line[line.__len__() - 1])
        del (line[line.__len__() - 1])
        self.P.append(line)
    fo.close()
    print("go 按钮按下")
    self.lines = self.textEdit.toPlainText()
    self.lines = self.lines.split('\n')   # 分割成组
    self.DB = set(self.lines)
    print(self.DB)
    self.str = ""
    print(self.str)
    flag = True
```

```
        temp = ""
        for x in self.P:    # 对于每条产生式规则
            if ListInSet(x, self.DB):    # 如果所有前提条件都在规则库中
                self.DB.add(self.Q[self.P.index(x)])
                temp = self.Q[self.P.index(x)]
                flag = False    # 至少能推出一个结论
                # print("%s --> %s" %(x, self.Q[self.P.index(x)]))
                self.str += "%s --> %s\n" % (x, self.Q[self.P.index(x)])

        if flag:    # 一个结论都推不出
            print("一个结论都推不出")
            for x in self.P:    # 对于每条产生式
                if ListOneInSet(x, self.DB):    # 事实是否满足部分前提
                    flag1 = False            # 默认提问时否认前提
                    for i in x:    # 对于前提中所有元素
                        if i not in self.DB:    # 对于不满足的那部分
                            btn = s.quest("是否" + i)
                            if btn == QtWidgets.QMessageBox.Ok:
                                self.textEdit.setText(self.textEdit.toPlainText() + "\n" + i)
                                # 确定则增加到 textEdit
                                self.DB.add(i)    # 确定则增加到规则库中
                                flag1 = True       # 肯定前提
                                # self.go(self)
                    if flag1:    # 如果肯定前提，则重新推导
                        self.go()
                        return
        self.textEdit_2.setPlainText(self.str)
        print("----------------------")
        print(self.str)
        if flag:
            btn = s.alert("啥也推不出来!!!")
            # if btn == QtWidgets.QMessageBox.Ok:    # 点击确定
            #         self.textEdit.setText(self.textEdit.toPlainText() + "\n 确定")
        else:
            self.lineEdit.setText(temp)
```

询问用户补充的推理信息进行进一步推理，代码如下：

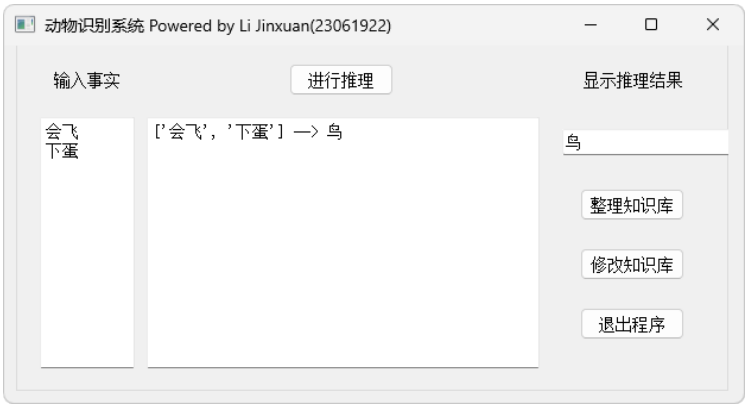
```
def quest(self, info):
            QtWidgets.QMessageBox.move(self, 200, 200)
              button = QtWidgets.QMessageBox.question(self, "Question",
                        self.tr(info),QtWidgets.QMessageBox.Ok | QtWidgets.QMessageBox.Cancel,
                          QtWidgets.QMessageBox.Cancel)
        return button
```

五、实验分析与结果

（一）完整代码：见附件 animal.py。

（二）测试样例与运行结果

（1）样例 1：输入“会飞 下蛋”。



按照预期，输出了推理结果“鸟”。

（2）样例 2：输入“鸟”。

由于输入事实较为匮乏，系统会自动进行多个提问，并将提问结果依次作为新的输入框。

可见，输入事实补充完成，最终输出预期推理结果：“鸵鸟”。

（3）样例 3：输入“会飞”

系统自动弹出进一步提问，若此时点击“Cancel”，因为知识库中不存在“会飞 不下蛋”的对应条目，故提示“无法推出结论”。

（三）实验结论

本次实验项目通过知识库结合知识图谱构建了一个动物识别的产生式系统，通过 PyQT5 生成用户交互页面，体现了 Python 在前台的优势，整体代码量相比 C/C++大大减少，体现了 Python 在编写专家系统的优势。