**程序报告**

学号：2213410 姓名：徐俊智

1. **问题重述**

（简单描述对问题的理解，从问题中抓住主干，必填）

====================================================================

5 个不同国家（英国、西班牙、日本、意大利、挪威）且工作各不相同（油漆工、摄影师、外交官、小提琴家、医生）的人分别住在一条街上的 5 所房子里， 每所房子的颜色不同（红色、白色、蓝色、黄色、绿色），每个人都有自己养的不同宠物（狗、蜗牛、斑马、马、狐狸），喜欢喝不同的饮料（矿泉水、牛奶、茶、橘子汁、咖啡）。

输出哪所房子里的人养斑马，哪所房子里的人喜欢喝矿泉水，以及正确的五条匹配信息，每条匹配信息依次包含(国家，工作，饮料，宠物，颜色)五个元素。

1. **设计思想**

（所采用的方法，有无对方法加以改进，该方法有哪些优化方向（参数调整，框架调整，或者指出方法的局限性和常见问题），伪代码，理论结果验证等… **思考题，非必填**）

====================================================================

导入kanren包。自定义左邻近规则left(), 邻近规则next()。在Agent类里，定义构造函数初始化unit变量，定义逻辑规则函数定义规则集合，定义规则求解器。初始化Agent类和规则求解器，输出提取解释器，展示解释器的输出结果。

1. **代码内容**

（能体现解题思路的主要代码，有多个文件或模块可用多个"===="隔开，必填）

====================================================================

from kanren import run, eq, membero, var, conde # kanren一个描述性Python逻辑编

程系统

from kanren.core import lall # lall包用于定义规则

import time

def left(q, p, list):

return membero((q, p), zip(list, list[1:]))

def next(q, p, list):

return conde([left(q, p, list)], [left(p, q, list)])

class Agent:

"""

推理智能体.

"""

def \_\_init\_\_(self):

"""

智能体初始化.

"""

self.units = var()# 单个unit变量指代一座房子的信息(国家，工作，饮料，宠物，颜色)

self.rules\_zebraproblem = None# 用lall包定义逻辑规则

self.solutions = None# 存储结果

def define\_rules(self):

"""

定义逻辑规则.

"""

self.rules\_zebraproblem = lall(

(eq, (var(), var(), var(), var(), var()), self.units), # self.units共包含五个unit成员，即每一个unit对应的var都指代一座房子(国家，工作，饮料，宠物，颜色)

(membero, ('英国人', var(), var(), var(), '红色'), self.units), # 英国人住在红房子里

(membero, ('西班牙人', var(), var(), '狗', var()), self.units), # 西班牙人养了一条狗

(membero, ('日本人', '油漆工', var(), var(), var()), self.units), # 日本人是一个油漆工

(membero, ('意大利人', var(), '茶', var(), var()), self.units), # 意大利人喝茶。

(eq, (('挪威人', var(), var(), var(), var()), var(), var(), var(), var()), self.units), # 挪威人住在左边的第一个房子里

(membero, (var(), '摄影师', var(), '蜗牛', var()), self.units), # 摄影师养了一只蜗牛

(membero, (var(), '外交官', var(), var(), '黄色'), self.units), # 外交官住在黄房子里

(eq, (var(), var(),(var(), var(), '牛奶', var(), var()),var(), var()),self.units),

(membero, (var(), var(), '咖啡', var(), '绿色'), self.units), # 喜欢喝咖啡的人住在绿房子里

(membero, (var(), '小提琴家', '橘子汁', var(), var()), self.units), # 小提琴家喜欢喝橘子汁

(membero,(var(), var(), var(), '斑马', var()), self.units),

(membero,(var(), var(), '矿泉水', var(), var()), self.units),

(left,(var(), var(), var(), var(), '绿色'),(var(), var(), var(), var(), '白色'),self.units),# 绿房子在白房子的右边

(next, ('挪威人', var(), var(), var(), var()),(var(), var(), var(), var(), '蓝色'), self.units), # 挪威人住在蓝房子旁边。

(next, (var(), '医生', var(), var(), var()), # 养狐狸的人所住的房子与医生的房子相邻

(var(), var(), var(), '狐狸', var()), self.units),

(next, (var(), '外交官', var(), var(), var()),(var(), var(), var(), '马', var()), self.units) # 养马的人所住的房子与外交官的房子相邻

)

def solve(self):

"""

规则求解器(请勿修改此函数).

return: 斑马规则求解器给出的答案，共包含五条匹配信息，解唯一.

"""

self.define\_rules()

self.solutions = run(0, self.units, self.rules\_zebraproblem)

return self.solutions

agent = Agent()

solutions = agent.solve()

# 提取解释器的输出

output = [house for house in solutions[0] if '斑马' in house][0][4]

print ('\n{}房子里的人养斑马'.format(output))

output = [house for house in solutions[0] if '矿泉水' in house][0][4]

print ('{}房子里的人喜欢喝矿泉水'.format(output))

# 解释器的输出结果展示

for i in solutions[0]:

print(i)

1. **实验结果**

（实验结果，必填）

====================================================================

绿色房子里的人养斑马

黄色房子里的人喜欢喝矿泉水

('挪威人', '外交官', '矿泉水', '狐狸', '黄色')

('意大利人', '医生', '茶', '马', '蓝色')

('英国人', '摄影师', '牛奶', '蜗牛', '红色')

('日本人', '油漆工', '咖啡', '斑马', '绿色')

('西班牙人', '小提琴家', '橘子汁', '狗', '白色')

1. **总结**

（自评分析（是否达到目标预期，可能改进的方向，实现过程中遇到的困难，从哪些方面可以提升性能，模型的超参数和框架搜索是否合理等），**思考题，非必填**）

====================================================================

我的代码达到了预期的功能。

针对更大规模的问题时，代码可能还需要考虑一些优化策略。

实现过程中对kanren包的使用还不够熟悉，对逻辑编程的理解还有待加强。

提升性能可以考虑调整kanren包提供的搜索策略的参数，或者使用并行计算。

模型的超参数和框架搜索很合理。