**程序报告**

学号：2311061 姓名：马淏怡

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）

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

斑马问题： 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）。

根据以下提示，你能告诉我哪所房子里的人养斑马，哪所房子里的人喜欢喝矿泉水吗？

1. 英国人住在红色的房子里

2. 西班牙人养了一条狗

3. 日本人是一个油漆工

4. 意大利人喜欢喝茶

5. 挪威人住在左边的第一个房子里

6. 绿房子在白房子的右边

7. 摄影师养了一只蜗牛

8. 外交官住在黄房子里

9. 中间那个房子的人喜欢喝牛奶

10. 喜欢喝咖啡的人住在绿房子里

11. 挪威人住在蓝色的房子旁边

12. 小提琴家喜欢喝橘子汁

13. 养狐狸的人所住的房子与医生的房子相邻

14. 养马的人所住的房子与外交官的房子相邻

问题中共有5个房子，每个房子有5种类型的数据。每个房子相当于一个逻辑变量，而这一个逻辑变量中还包含5个分别代表国家、工作、饮料、宠物、颜色的逻辑变量，要根据14条已知的信息来求解五个房子之间的关系。

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**）

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

1. 导包

from kanren import run, eq, membero, var, conde

from kanren.core import lall

2、自定义函数left，right，next。题目要求是，需要使用“相邻”、“右边”、“左边”这三个关系，使用python中的元组来实现。把每个房子都和它旁边的房子打包为元组，然后返回由这些元组组成的列表。此时元组中左右元素即为相邻房子的左右顺序，使用 membero 包含关系，赋予参数x、y左右的位置关系。这样就获得了三个包含以上关系的元组。

3、units变量

每个unit记录了一组信息（国家，职业、饮料、宠物、颜色），首先将unit赋值为空

self.units =（var， var，var，var，var）

agent 中定义了 rules\_zebraproblem 和 solutions ，用来定义规则和存储结果。在智能体中，有许多定义规则的函数，使用 kanren 包中的 lall 函数定义规则来列举：

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

(eq, (('挪威人', var(), var(), var(), var()), var(), var(), var(), var()), self.units)挪威人住在左数第一间

1. 将14条关系写出
2. 利用规则求解器求解
3. **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

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

import time

def left(x, y, units):

# x在y左

groups = zip(units, units[1:])#组合每对相邻元素

return membero((x, y), groups)#xy是否相邻

def right(x, y, units):

return left(y, x, units)

def next(x, y, units):

return conde([left(x, y, units)], [right(x, y, units)])#或左或右即相邻

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)

# 例如('英国人', '油漆工', '茶', '狗', '红色')即为正确格式，但不是本题答案

# 请基于给定的逻辑提示求解五条正确的答案

self.rules = None # 用lall包定义逻辑规则

self.solutions = None # 存储结果

def define\_rules(self):

self.rules = lall(

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

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

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

(membero, ('英国人', var(), var(), var(), '红色'), self.units),

(membero, ('西班牙人', var(), var(), '狗', var()), self.units),

(membero, ('日本人', '油漆工', var(), var(), var()), self.units),

(membero, ('意大利人', var(), '茶', var(), var()), self.units),

(eq, (('挪威人', var(), var(), var(), var()), var(), var(), var(), var()), self.units),

(right, (var(), var(), var(), var(), '绿色'), (var(), var(), var(), var(), '白色'), self.units),

(membero, (var(), '摄影师', var(), '蜗牛', var()), self.units),

(membero, (var(), '外交官', var(), var(), '黄色'), self.units),

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

(membero, (var(), var(), '咖啡', var(), '绿色'), self.units),

(next, ('挪威人', var(), var(), var(), var()), (var(), var(), var(), var(), '蓝色'), self.units),

(membero, (var(), '小提琴家', '橘子汁', var(), var()), self.units),

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

(next, (var(), var(), var(), '马', var()), (var(), '外交官', var(), var(), var()), self.units)

)

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)

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]:

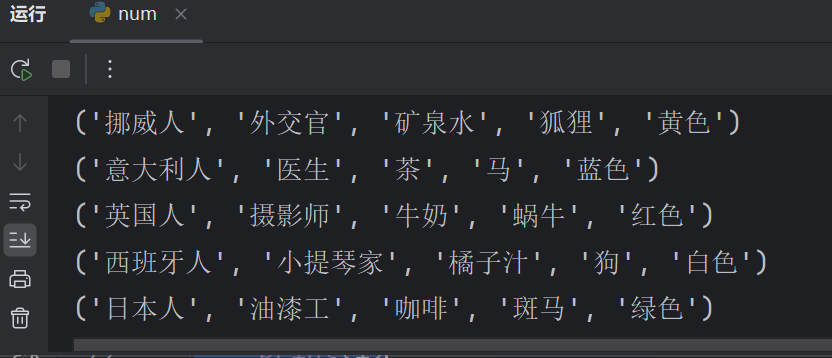
print(i)

1. **实验结果**

（实验结果，必填）

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

1. 本地pycharm运行



1. 平台测试



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**）

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

1. 达到目标预期
2. 学习使用kanren库的过程中遇到了困难，通过查阅资料解决
3. 依照示例完成了14条条件的编写
4. 本次实验常规做法也许是用穷举法？Kanren库已经极大提高性能