程序报告

学号：2313721　　姓名：许洋

一、问题重述

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。而问题的要求则是根据给出的、关于这五组变量之间的部分逻辑关系，进行逻辑推理，最后得出所有的未知数据。问题可以抽象为一个5*5的数组，已知这个数组每行的部分元素值和行之间的位置关系，逻辑推算出该数组的全部元素。

二、设计思想

首先根据提示我们可以自定义之后会用到的函数：left、right、和next函数

构建方法：

将房子列表错位zip，使得每个房子都和它旁边的房子对应打包为元组，使用kanren中的membero，即包含逻辑关系，赋予参数x、y左右的位置关系。

```
def right(x,y,lst):
    return membero((y,x),zip(lst,lst[1:]))
def left(x,y,lst):
    return membero((x,y),zip(lst,lst[1:]))
def next(x,y,lst):
    return conde([left(x,y,lst)],[right(x,y,lst)])
```

构建智能体类对象 agent ， 类中定义逻辑变量units

```
self.units = [var() for _ in range(5)]
```

units中包含5个房子的逻辑变量，而每个房子的逻辑变量var又包括5个逻辑变量 (国家，工作，饮料，宠物，颜色）。agent中还定义了rules_zebraproblem 和 solutions，分别用来定义规则和存储结果。

智能体类中定义规则函数：

使用kanren包中的lall函数定义规则

```
self.rules_zebraproblem = lall(
            (eq, (var(), var(), var(), var(), var()), self.units),
```

membero表示 包含关系,下例表明 红色的、住着英国人的房子var 包含在units里。

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

eq表示相等关系，例如中间那个房子的人喜欢喝牛奶

```
(eq, self.units[2], (var(), var(), '牛奶', var(), var())),
```

完整代码

```
(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, self.units[0], ('挪威人', var(), var(), var(),
var())),
            (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, self.units[2], (var(), var(), '牛奶', var(),
var())),
            (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),

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

规则求解，利用之前定义的逻辑关系，调用run函数得到问题解

```
    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
```

三、代码内容

```
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

###########################################################################
############
####                可在此处定义自己所需要用到的自定义函数(可选)
    ####
####    提示：定义左邻近规则left()， 定义右邻近规则right(),定义邻近规则next()
    ####
###########################################################################
############
#
          #
def right(x,y,lst):
    return membero((y,x),zip(lst,lst[1:]))
def left(x,y,lst):
    return membero((x,y),zip(lst,lst[1:]))

def next(x,y,lst):
    return conde([left(x,y,lst)],[right(x,y,lst)])
#
          #
###########################################################################
############
#################                  非必要性工作
##################
```

```
#########################################################################
############

class Agent:
    """
    推理智能体.
    """

    def __init__(self):
        """
        智能体初始化.
        """

        self.units = [var() for _ in range(5)]                # 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_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)

        # 各个unit房子又包含五个成员属性: (国家，工作，饮料，宠物，颜色)


 #########################################################################
############
            ####                  请在以下区域中添加逻辑规则，感受逻辑约束问题
              ####
            ####      输出：五条房子匹配信息('英国人', '油漆工', '茶',
'狗', '红色')        ####
```

```
 ##########################################################################
############
            #
                    #

            # 示例：基于问题信息可以提炼出，有人养斑马，有人喜欢和矿泉水等信息
            (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, self.units[0], ('挪威人', var(), var(), var(),
var())),
            (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, self.units[2], (var(), var(), '牛奶', var(),
var())),
            (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),

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

```
            #
                    #

 ##########################################################################
############
            ##################              完成后请记得提交作业
    ##################

 ##########################################################################
############

        )

    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)
```

四、实验结果

- 输出结果如下：

```
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')
('西班牙人', '小提琴家', '橘子汁', '狗', '白色')
('日本人', '油漆工', '咖啡', '斑马', '绿色')
```

- 平台检测结果：

五、总结

- 实验运用 kanren 的逻辑包进行逻辑推理，最终得到结果。
- 问题关键在于理解 kanren 包的使用规则。
- 难点在于理解利用切片和zip构造 left、right、next 位置关系。