程序报告

学号：2211044　　姓名：陆皓喆

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

对问题的理解

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，我们需要根据14条已知的信息来求解五个房子之间的关系。

二、设计思想

1.导入python包

首先，根据题目要求，我们需要使用`kanren`包和`lall`包，所以我们使用语句

```
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
```

来完成python包的调用。注意，在系统上已经自动导入了这两个包，如果在自己的python环境下还需要进行`pip install`操作。

2.自定义功能函数

然后，我们需要自定义本次实验需要使用的函数left，right，next函数。题目要求是，需要使用“相邻”、“右边”、“左边”这三个位置关系，我们使用python中的元组来实现。

我们将房子列表错位zip，使得每个房子都和它旁边的房子对应打包为元组，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`kanren`中的`membero`，即包含逻辑关系，赋予参数x、y左右的位置关系。这样，我们就获得了三个对应的元组，其中包含了这三种简单的逻辑关系。

```
def left(x,y,units):
    groups=zip(units,units[1:])
    # units为原来的房子序列，通过切片units[1:]使其错位一个房子，然后用zip打包。
    return membero((x,y),groups)

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)])
```

3.构建智能体类对象

```
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)
```

我们首先给每一个`units`变量赋值为`var()`，即为空值。

```
(eq, (var(), var(), var(), var(), var()), self.units) # 相当于 self.units =（var, var, var, var, var）
```

然后我们初始化每一个`self.units`，赋五个`var`初值，这很显然，刚开始时，智能体并不知道具体的信息，所以赋五个空值。

`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

在智能体中，有许多定义规则的函数，我们使用`kanren`包中的`lall`函数定义规则，下面我们一一列举。

(1)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)
```

(2)eq

表示相等关系， 下例表明，挪威人住在左边的第一个房子里

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

(3)left,right,next

上文我们已经详细说明，在此处不多介绍。

4.实现14条逻辑关系

代码如下，我们使用上面所描述的逻辑关系内容来进行编写代码即可。

```
def define_rules(self):
        self.rules_zebraproblem = 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)
        )
```

5.规则求解器

```
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
```

本段实现的是对于逻辑关系的求解，我们调用`run`函数，就可以解出对应的逻辑关系。

三、代码内容

本次实验的python代码如下所示。

```
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(x,y,units):
    groups=zip(units,units[1:])
    return membero((x,y),groups)

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)])

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_zebraproblem = None  # 用lall包定义逻辑规则
        self.solutions = None           # 存储结果

    def define_rules(self):
```

```
        self.rules_zebraproblem = 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_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

在线测试

我们将代码导出为`main.py`，进行在线测试，获得以下的输出。

```
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')
```

平台测试

在平台的测试端，我们得到以下结果。

五、总结

1. 本次实验巧妙运用了`kanren`包的使用，利用其强大的逻辑推理能力来完成斑马问题的推理。
2. 问题的关键之处就是理解`kanren`语句中的逻辑关系，学会编写逻辑关系的代码，能够读懂`kanren`逻辑语句。
3. 该逻辑语句的实现的核心就是在于对于自定义函数的理解与编写，我们需要利用切片与zip来实现三个函数功能——left，right，next。
4. 我们可以发现，该问题实际上可以通过穷举法来实现，方法是先生成结果，然后与条件一一匹配，但是这样做虽然思路简单，但是所消耗的时间很长，这个方法不太好，于是我们使用了`kanren`逻辑关系库来进一步实现。