斑马问题求解程序报告

学号：2312900　　姓名：禹相祐

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

自己对问题的理解：

题目中要求解的是哪个房子的人养斑马和哪个房子的人喝矿泉水。所以需要求解的其实就是房子的编号，而每个房子都可以存储 5 种信息，即（国家，工作，颜色，宠物，饮料）。所以我们需要的就是利用 kanren 将一个个已知的条件存进对应的房子，然后利用 kanren 结合这些信息实现推理。

二．设计思想

1. 导入 kanren：

```
from kanren import run, eq, membero, var, conde       # kanren 一个描述性 Python 逻辑
from kanren.core import lall                          # lall 包用于定义规则
```

2. 定义相邻函数的表示：

我们可以先实现表示一个房子在另一个房子的左边和右边，这样我们就可以利用 kanren 库内的取逻辑“或”的方式实现两个房子相邻的表示方法。

先定义函数：left_to（x,y,units）：

```
def left_to(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)
```

再定义函数：right_to（x,y,units）：

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

然后实现函数 next_to（x,y,units）：

```
def next_to(x,y,units):
    return conde([left_to(x, y, units)], [right_to(x, y, units)])
```

3. 求解实现:

```
# 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.units = var()
```

首先将 units 置为 var 类型的变量

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

然后介绍 membero 函数：

```
# 此处将英国人住在红色的房子表现了出来，等于成功实现了一个条件的添加
(eq(membero, ('英国人',var(),var(),var(),'红色'),self.units)
```

接着实现全部的 14 个条件：

```
self.rules_zebraproblem = lall(
            # 对 units 进行赋值,等于 units=(var,var,var,var,var)
            (eq, (var(), var(), var(), var(), var()), self.units),
            # 求解的 Q1
            (membero, (var(), var(), var(), '斑马', var()), self.units),
            # 求解的 Q2
            (membero, (var(), var(), '矿泉水', var(), var()), self.units),
            # 开始输入题设条件
            (membero, ('英国人', var(), var(), var(), '红色'), self.units), # Condition 1
            (membero, ('西班牙人', var(), var(), '狗', var()), self.units), # Condition 2
            (membero, ('日本人', '油漆工', var(), var(), var()), self.units),# Condition 3
            (membero, ('意大利人', var(), '茶', var(), var()), self.units), # Condition 4
            (eq, (('挪威人', var(), var(), var(), var())
                  , var(), var(), var(), var()), self.units), # Condition 5
```

```
            (right_to, (var(), var(), var(), var(), '绿色'), (var(), var(), var(),
                                                    var(), '白色'), self.units),# Condition 6
            (membero, (var(), '摄影师', var(), '蜗牛', var()), self.units), # Condition 7
            (membero, (var(), '外交官', var(), var(), '黄色'), self.units),# Condition 8
            (eq, (var(), var(), (var(), var(), '牛奶', var(), var()), var(), var()
                  ), self.units), # Condition 9
            (membero, (var(), var(), '咖啡', var(), '绿色'), self.units), # Condition 10
            (next_to, ('挪威人', var(), var(), var(), var()), (var(), var(), var(),
                                                     var(), '蓝色'), self.units), # Condition 11
            (membero, (var(), '小提琴家', '橘子汁', var(), var()), self.units), # Condition 12
            (next_to, (var(), var(), var(), '狐狸', var()), (var(), '医生', var(),
                                                    var(), var()), self.units), # Condition 13
            (next_to, (var(), var(), var(), '马', var()), (var(), '外交官', var(),
                                                   var(), var()), self.units) # Condition 14
        )
```

最后实现求解：

```
 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：

完整代码如下：

```
    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
# units 为原先的房子序列
def left_to(x,y,units):
    # zip 返回的是(1,2),(2,3),(3,4),(4,5)
    groups=zip(units,units[1:])
    # 我们将所有可能的关系都在 groups 内，所以利用 membero() 根据输入的 x&y，我就知道对不对了
    return membero((x,y),groups)
def right_to(x,y,units):
    # 同上，只不过变为（2，1）类似的
    groups = zip(units[1:], units)
    # 同上
    return membero((x,y),groups)
def next_to(x,y,units):
    # 此处采用或操作，使用 conde() 实现
    return  conde([left_to(x, y, units)], [right_to(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.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)
        self.rules_zebraproblem = lall(
            # 对units进行赋值,等于units=(var,var,var,var,var)
            (eq, (var(), var(), var(), var(), var()), self.units),
            # 求解的Q1
            (membero, (var(), var(), var(), '斑马', var()), self.units),
            # 求解的Q2
            (membero, (var(), var(), '矿泉水', var(), var()), self.units),
            # 开始输入题设条件
            (membero, ('英国人', var(), var(), var(), '红色'), self.units), # Condition 1
            (membero, ('西班牙人', var(), var(), '狗', var()), self.units), # Condition 2
            (membero, ('日本人', '油漆工', var(), var(), var()), self.units),# Condition 3
            (membero, ('意大利人', var(), '茶', var(), var()), self.units), # Condition 4
            (eq, (('挪威人', var(), var(), var(), var())
                  , var(), var(), var(), var()), self.units), # Condition 5
            (right_to, (var(), var(), var(), var(), '绿色'), (var(), var(), var(),
                                                          var(), '白色'), self.units),# Condition 6
            (membero, (var(), '摄影师', var(), '蜗牛', var()), self.units), # Condition 7
            (membero, (var(), '外交官', var(), var(), '黄色'), self.units),# Condition 8
            (eq, (var(), var(), (var(), var(), '牛奶', var(), var()), var(), var()
                  ), self.units), # Condition 9
            (membero, (var(), var(), '咖啡', var(), '绿色'), self.units), # Condition 10
            (next_to, ('挪威人', var(), var(), var(), var()), (var(), var(), var(),
                                                           var(), '蓝色'), self.units), # Condition 11
            (membero, (var(), '小提琴家', '橘子汁', var(), var()), self.units), # Condition 12
            (next_to, (var(), var(), var(), '狐狸', var()), (var(), '医生', var(),
                                                         var(), var()), self.units), # Condition 13
            (next_to, (var(), var(), var(), '马', var()), (var(), '外交官', var(),
```

```
                                                    var(), var()), self.units) # Condition 14
        )
    # 求解函数
    def solve(self):
        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：

```
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')

进程已结束，退出代码为 0
```

五. 总结

kanren 库的效果奇佳，而且内容不多，用起来也十分的简单。虽然感觉没这个库也会有办法实现，但如果得靠自己写判断函数来实现推断的话，工作量想一想就很恐怖。这次也了解到了逻辑推理的有意思之处，这样的问题如果不靠代码，自己求解出来也很有意思！