斑马问题程序报告

学号： 2111460　　姓名：张洋

一、问题重述

逻辑编程是一种编程典范，它设置答案须匹配的规则来解决问题，而非设置步骤来解决问题。过程是**“事实+规则=结果”**。

本次实验我们将 Python 应用于逻辑编程，并尝试自主撰写逻辑规则解决斑马问题。

斑马问题重述：题目中给出了十四条提示，我们将利用这些提示进行推理，将同一组的属性一一对应起来，将不可能的结果排除。例如，根据“英国人住在红色的房子里”和“外交官住在黄房子里”可以推出“英国人不是外交官”；根据“挪威人住在左边的第一个房子里”和“挪威人住在蓝色的房子旁边”可以推出“蓝色的房子是第二间”等等。

利用 python 的 kanren 库，将属性打包为一个整体，进而对问题做出解答。问题为：哪所房子里的人养斑马，哪所房子里的人喜欢喝矿泉水。

二、设计思想

使用 python 逻辑编程 kanren 库，将题目中的条件形式化为一个个的表达式，加入约束到 kanren 的一个集合中，然后利用 kanren 内置的 run 求解。

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

def left(a,b,list):
    return membero((a,b), zip(list, list[1:]))

def right(a,b,list):
    return membero((b,a), zip(list, list[1:]))

def is_next(a,b,list):
    return conde([left(a,b,list)],[right(a,b,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)
            # 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)

                #1.英国人住在红色的房子里
                (membero,('英国人', var(), var(), var(), '红色'), self.units),
                #2.西班牙人养了一条狗
                (membero,('西班牙人', var(), var(), '狗', var()), self.units),
                #3.日本人是一个油漆工
                (membero,('日本人', '油漆工', var(), var(), var()), self.units),
                #4.意大利人喜欢喝茶
                (membero,('意大利人', var(), '茶', var(), var()), self.units),
                #5.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),
                #6.绿房子在白房子的右边
                (left, (var(), var(), var(), var(), '绿色'), (var(), var(), var(), var(), '白色'), self.units),
                #7.摄影师养了一只蜗牛
                (membero,(var(), '摄影师', var(), '蜗牛', var()), self.units),
                #8.外交官住在黄房子里
                (membero,(var(), '外交官', var(), var(), '黄色'), self.units),
                #9.中间那个房子的人喜欢喝牛奶
                (eq, (var(), var(), (var(), var(), '牛奶', var(), var()), var(), var()), self.units),
                #10.喜欢喝咖啡的人住在绿房子里
                (membero,(var(), var(), '咖啡', var(), '绿色'), self.units),
                #11.挪威人住在蓝色的房子旁边
                (is_next, ('挪威人', var(), var(), var(), var()), (var(), var(), var(), var(), '蓝色'), self.units),
                #12.小提琴家喜欢喝橘子汁
```

```
            (membero,(var(), '小提琴家', '橘子汁', var(), var()), self.units),
            #13.养狐狸的人所住的房子与医生的房子相邻
            (is_next, (var(), var(), var(), '狐狸', var()), (var(), '医生', var(), var(), var()), self.units),
            #14.养马的人所住的房子与外交官的房子相邻
            (is_next, (var(), var(), var(), '马', var()),(var(), '外交官', var(), var(), var()), self.units),
            #有人养斑马，有人喜欢喝矿泉水
            (membero,(var(), var(), var(), '斑马', var()), self.units),
            (membero,(var(), var(), '矿泉水', 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

```
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')
```

五、 总结

本次实验已经达到了预期目标，通过了测试用例，给出了正确结果。

可能改进的方向：可以在掌握更多知识后自己定义逻辑语法，提高性能。通过更深入的学习和搜索，我们可以在网上找到其他的逻辑定义：

（1）利用 google or-tools 。

OR-Tools 是一个用于优化的开源软件套件，用于解决车辆路径、流程、整数和线性规划以及约束编程等世界上最棘手的问题。

当前 ortools 提供的优化器包括： - 约束规划 - 线性与混合整数规划 - 路径规划 - 调度规划 - 网络规划 - 装箱。

（2）利用 prolog

可以根据人将题目中的六个属性联系在一起，即分为人的个人属性和住房属性，定义人的谓词 people(Country, Work, Drink, Pet) 以及房子谓词 house(Country, Place, Color) 两个谓词来进行基本属性的书写，其中 Country 为两个谓词中的共有属性。

实验过程中遇到的困难：

（1） 由于对于 kanren 包不太熟悉，一开始不知道如何下手，导致前期一直在研究 kanren 包中的逻辑语法。现在已经可以熟练地运用包中的逻辑语句。

（2） 对于实验平台不太熟悉。一开始对于平台中代码的提交，运行等过程不太了解，经过询问老师和助教对此平台加深了解，最终完成此实验。