知识表示：用谓词逻辑表示法求解简单问题

# 1实验目的

1. 熟悉一阶谓词逻辑表示法；
2. 了解一阶谓词表示法的优点和局限性；
3. 了解如何将一阶谓词逻辑转化为计算机语言来求解问题。

# 2实验要求

本次试验后，要求学生能：

1. 熟悉用Python语言编写程序；
2. 理解一阶谓词逻辑的表示方法，以及如何利用一阶谓词逻辑求解问题。

# 3实验原理

## 3.1谓词

**谓词**（predicate）逻辑是基于命题中谓词分析的一种逻辑。一个谓词可分为谓词名与个体两个部分。个体表示某个独立存在的事物或者某个抽象的概念；谓词名用于刻画个体的性质、状态或个体间的关系。

谓词的一般形式是：

P（x1, x2, … , xn）

其中，P为**谓词名**，x1, x2, … , xn是**个体**，个体可以是常量，也可以是变元，还可以是函数。当P（x1, x2, … , xn）中个体全是常量或变元或函数（没有谓词作为个体），则称为**一阶谓词**。

需要注意，谓词和函数（这里的函数指逻辑中的函数，而非编程里的“函数”）表面上相似，实际上却是两个完全不同的概念，谓词的真值是“真”或“假”，而函数的值是个体域中的某个个体，函数无真值可言，只是个体域中一个个体到另一个个体的映射。

## 3.2谓词公式

按以下规则可得到谓词公式：

（1）单个谓词是谓词公式，称为原子谓词公式；

（2）若A是谓词公式，则﹁A也是谓词公式；

（3）若A，B都是谓词公式，则A∧B，A∨B，A→B，A↔B也都是谓词公式；

（4）若A是谓词公式，则(∀x)A，(∃x)A也是谓词公式；

（5）有限步应用①~④生成的公式也是谓词公式。

## 3.3一阶谓词逻辑知识表示方法

用一阶谓词逻辑表示知识的一般步骤为：

（1）定义谓词及个体，确定每个谓词及个体的确切意义；

（2）根据要表达的事务或概念，为谓词中的变元赋予特定的值；

（3）根据语义用适当的连接符号将各个谓词连接起来，形成谓词公式。

# 4实验步骤

## 4.1实验介绍

本实验将用一阶谓词逻辑求解一个简单的问题**（注：为简化问题，本题认为一个人不是男性就是女性）：**

规则一：任何人的兄弟不可能是女性。

规则二：任何人的姐妹不可能是男性。

已知事实：Marry是Bill的姐姐或妹妹

问：

（1）Marry是不是Tom的兄弟？

（2）Marry是不是Bill的姐妹？

（3）Marry是不是Tom的姐妹？

本实验需要大家有一定的Python语言编程基础

## 4.2用谓词公式表示问题

### 4.2.1定义谓词：

brother（x , y）：x是y的兄弟。

sister（x , y）：x是y的姐妹。

is\_male（x）：x是男性。

is\_female（x）：x是女性。

### 4.2.2得到谓词公式

把已知事实和规则表示成谓词公式，得：

规则一：∀x∀y（brother（x , y）→ ﹁is\_female（x））

规则二：∀x∀y（sister（x , y）→ ﹁is\_male（x））

事实：sister（Marry，Bill）

同时，可得到规则一、规则二的逆否命题：

规则一逆否：∀x∃y is\_female（y）→ ﹁brother（x , y）

规则二逆否：∀x∃y is\_male（y）→ ﹁sister（x , y）

## 4.3将谓词公式转化为计算机语言

（1）创建集合，用于保存人物关系、人物性别等信息。

sister\_of\_Bill = {""}

sister\_of\_Tom = {""}

brother\_of\_Bill = {""}

brother\_of\_Tom = {""}

male = {""}

female = {""}

（2）实现谓词sister(x1 , x2)、brother(x , y）。

定义函数sister（x , y , sister\_of\_y），函数的功能为将x加入y姐妹对应的集合sister\_of\_y中；函数brother（x , y , brother\_of\_y）同理。

def sister(x1, x2, sister\_of\_x2):

print(x1 + "是" + x2 + "的姐妹")

sister\_of\_x2.add(x1)

print("已将" + x1 + "添加进" + x2 + "姐妹的集合")

def brother(x3, x4, brother\_of\_x4):

print(x3 + "是" + x4 + "的兄弟")

brother\_of\_x4.add(x3)

print("已将" + x3 + "添加进" + x4 + "兄弟的集合")

（3）实现谓词is\_male（x）、is\_female（x）。

定义函数is\_female（x），功能为将x加入female集合；函数is\_male（x）同理。

def is\_female(y1):

global female

print(y1 + "是女性")

female.add(y1)

def is\_male(y2):

global male

print(y2 + "是男性")

male.add(y2)

（4）用规则一、规则二来实现性别判断逻辑。

定义函数judge\_gender(A)，功能为判断A是男性还是女性。若A是男性，则谓词is\_male(A)为真，调用函数is\_male(A)将A加入male集合；若A为女性，则谓词is\_female(A)为真，调用函数is\_female(A)将A加入female集合

def judge\_gender(A):

# 规则一 ：任何人的兄弟不可能是女性（所以是男性）。

if A in brother\_of\_Tom or A in brother\_of\_Bill:

is\_male(A)

# 规则二 ：任何人的姐妹不可能是男性（所以是女性）。

if A in sister\_of\_Tom or A in sister\_of\_Bill:

is\_female(A)

（5）用规则一、规则二的逆否命题来实现判断兄弟姐妹关系的逻辑。

定义函数judge\_brother(x, y, brother\_of\_y)，功能为判断x与y是否有兄弟关系。先遍历brother\_of\_y集合，看x是否已在其中，若在则直接输出x是y的兄弟；否则遍历female集合，如果x在female集合中说明x是女性，根据规则一的逆否命题，x不可能是y的兄弟。如果x也不在female集合中，说明信息不足，无法判断。

函数judge\_sister(x, y, sister\_of\_y)同理。

# 判断x是否是y的兄弟

def judge\_brother(x, y, brother\_of\_y):

if x in brother\_of\_y:

print(x + "是" + y + "的兄弟")

# 这里用到了规则一的逆否命题： 女性不可能是任何人的兄弟

elif x in female:

print(x + "不是" + y + "的兄弟")

else:

print("信息不足！" + x + "有可能是" + y + "的兄弟")

# 判断x是否是y的姐妹

def judge\_sister(x, y, sister\_of\_y):

if x in sister\_of\_y:

print(x + "是" + y + "的姐妹")

# 这里用到了规则二的逆否命题： 男性不可能是任何人的姐妹

elif x in male:

print(x + "不是" + y + "的姐妹")

else:

print("信息不足！" + x + "有可能是" + y + "的姐妹")

（6）最后在主函数中，根据事实和已知的规则进行判断。

if \_\_name\_\_ == '\_\_main\_\_':

# 事实：Marry是Bill的姐姐

sister("Marry", "Bill", sister\_of\_Bill)

# 利用已知规则和事实进行推理

judge\_gender("Marry")

print("")

print("可以得出结论：")

judge\_brother("Marry", "Tom", brother\_of\_Tom)

judge\_sister("Marry", "Bill", sister\_of\_Bill)

judge\_sister("Marry", "Tom", sister\_of\_Tom)

# 5实验结果

运行程序后，运行结果如下：

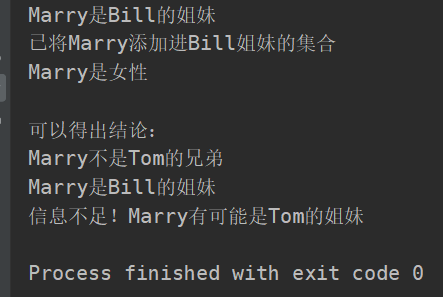


图1 推理结果