Learn Prolog Now 翻译 - 第一章 - 事实，规则和查询 - 第一节， 一些简单的例子

该系列文章是网上的Prolog学习资料：www.learnprolognow.org的中文翻译。希望能够通过翻译此学习资料，达到两个目的：第一、系统学习prolog的知识；第二、提升英文文章理解

和翻译能力。

内容提要：

给出一些Prolog编程的简单例子；

Prolog的基本结构：事实，规则和查询；

环境说明：

本系列文章使用的Prolog运行环境是：SWI-Prolog，官网地址是：http://www.swi-prolog.org。

Prolog中只有三种基础结构：事实（facts），规则（rules）和查询（queries）。事实和规则的集合称为知识库（knowledge base）（或者称为数据库，为了和传统意义上的

数据库进行区分，统一使用知识库），Prolog的编程几乎就是在编写知识库。换种说法，Prolog编程基本上等于编写由有趣的事实和规则组合而成的知识库。

那么如何使用Prolog程序呢？通过查询，即通过问一些问题，这些问题的信息和答案是存储在知识库中的。

可能听上去很奇怪，这和传统意义上的编程似乎没有什么关系，毕竟编程应该是告诉计算机做什么的吧？但是我们将会看到，Prolog的编程方式是非常有意义的，至少在特定的领域；

比如计算机语言学和人工智能领域。让我们进入具体编写一些简单知识库的实践，在学习Prolog的过程中，实践是最好也是唯一的方法。

Knowledge Base 1

Knowledge Base 1 （KB1）只是一些简单事实的集合。事实是指无条件为真的一些事物、状态或者关系。比如：我们可以定义Mia，Jody和Yolanda是女士，Jody在弹吉他，然后有

一个聚会，在Prolog中,可以通过下面5个事实进行定义：（注意每个事实的定义都是使用英文字符的句号作为结束标识符）

```
woman(mia).
woman(jody).
```

--Benjamin Yan


```
woman(yolanda).
playsAirGuitar(jody).
party.
```

上面的事实集合就是KB1，是我们第一个关于Prolog编程的例子。注意，上面提及的名字mia，jody，yolanda，她们的women属性（可以先这么理解）和弹吉他，及其聚会，都是小写字

母开头，我们稍后会详细解释其中的原因。

如何使用KB1呢？通过查询。即，通过查询一些KB1包括的信息来使用KB1。下面是一些例子，比如我们通过下面的查询可以问Prolog，Mia是不是一位女士：

?- women(mia).

Prolog会回答：true

因为在KB1中明确地定义了women(mia)的事实，所以Prolog的答案是true。注意，在实际使用Prolog查询的时候，我们不需要显式输入?-，这个Prolog（可能不同的Prolog解释器会略

有不同）是待输入符号。在查询语句的最后，一定要输入英文句号作为结束符，如果没有输入，那么Prolog不会执行查询操作，而是一直等待。

类似地，我们可以通过下面的查询语句问Prolog，jody是否弹吉他：

?- playsAirGuitar(jody).

Prolog同样会问答：true，因为这也是KB1中的一个事实。但是，如果我们尝试问Mia是否弹吉他：

?- playsAirGuitar(mia).

Prolog会回答：false，因为这不是KB1中的事实，而且KB1很简单，其他事实也不能推导出这个结论，所以Prolog认为playsAirGuitar(mia)在KB1中不能成立。

下面是两个重要的例子。首先，如果我们这样查询：

?- playsAirGuitar(vincent).

Prolog又会回答false。为什么？因为这个查询中提及的vincent这个人，在KB1中没有与之相关的任何信息，所以Prolog认为KB1中不能推导出关于他的任何其他信息。

类似的，如果我们这样查询：

?- tatooed(jody).

Prolog同样会回答false。为什么？因为这个查询中提及的tatooed这个属性，在KB1中没有与之相关的任何信息，所以Prolog认为不能推导出这个属性相关的任何其他信息。

无需多说，我们可以查询关注的属性，比如：

?- party.

Prolog会回答true。如果我们查询：

?- rockConcert.

Prolog会回答false，和我们的期望是一致的。

Knowledge Base 2

下面是KB2，我们的第二个知识库的定义：

```
happy(yolanda).
listen2Music(mia).
listen2Music(yolanda) :- happy(yolanda).
playsAirGuitar(mia) :- listen2Music(mia).
playsAirGuitar(yolanda) :- listen2Music(yolanda).
```

在KB2中，有两个事实：listen2Music(mia)和happy(yolanda)，剩下的三个都是规则。**Prolog中的规则是有条件为真的一些事物、状态或者关系**。比如规则一可以这么理解：

Yolanda听音乐如果Yolanda很高兴，最后一个规则可以这么理解：Yolanda弹吉他如果Yolanda听音乐。更抽象地理解，符号:-理解为“如果”，或者“以什么为前提”。在:-左边的部分

是规则的头部，在:-右边的部分是规则的主干，所以规则可以这么理解：如果一个规则的主干为真，那么这个规则的头部也为真。所以，以下是Prolog规则运用的要点：

如果知识库包括了一个规则，head :- body，并且Prolog知道在知识库中，body部分为真，那么Prolog就能够推导head为真。推导的基础步骤称为假言推理（modus ponens）。

让我们继续看具体的例子，如果我们查询Mia是否弹吉他：

?- playsAirGuitar(mia).

Prolog会回答true，为什么？毕竟在KB2中，playsAirGuitar(mia)不是一个事实，但是可以找到关于它的一个规则：

playsAirGuitar(mia) :- listen2Music(mia).

可以明确知道，KB2中包含了listen2Music(mia)的事实。所以Prolog可以使用规则的假言推理推导出playsAirGuitar(mia)为真。

下面另外一个例子显示，Prolog可以使用假言推理链，如果我们查询：

?- playsAirGuitar(yolanda).

Prolog会回答true，为什么？首先，通过使用happy(yolanda)的事实，及其相关的规则：

listen2Music(yolanda) :- happy(yolanda).

Prolog可以推导出一个新的事实：listen2Music(yolanda)。这个新事实在知识库中是隐式存在的（通过推导得出），但是，Prolog可以像使用显式的事实一样使用它。接下来，

通过这个推导的事实及其规则：

playsAirGuitar(yolanda) :- listen2Music(yolanda).

Prolog可以推导中新的事实：playsAirGuitar(yolanda)，即我们的查询结果为真。总结一下：一个假言推理的任何事实，可以用作其他规则的输入，通过链接的方式，将所有的假

言推理应用组合起来，Prolog就可以从知识库包含的事实和规则中推导出任何符合逻辑的信息。

在知识库中的事实和规则统称为子句（clauses）。所以KB2包括了5个子句，其中有3个规则和2个事实。另外一种看待KB2的方式可以这么说，它是由3个谓词（predicates）（或者

称为procedures）组成，三个谓词是：listen2Music，happy，playsAirtGuitar。（谓词和之前出现的属性是相同的含义，为了统一，今后统一使用谓词）

其中happy谓词由一个独立的子句（一个事实）组成。listen2Music和playsAirGuitar谓词分别由两个子句（listen2Music两个子句一个是事实，另外一个是规则；playsAirGuitar

两个子句都是规则）组成。可以认为Prolog编程就是由谓词构成的。本质上来说，谓词的概念很重要，编程中各种子句都是关于谓词代表的含义及其推导的含义。

还有一点可以提及的是，我们可以把事实看着没有主干的规则，即我们可以认为事实是无论任何条件都成立的规则。

Knowledge Base 3

KB3，我们的第三个知识库，由5个子句组成：

```
happy(vincent).
listen2Music(butch).

playsAirGuitar(vincent) :-
       listen2Music(vincent),
       happy(vincent).

playsAirGuitar(butch) :-
       happy(butch).

playsAirGuitar(butch) :-
       listen2Music(butch).
```

KB3中定义了两个事实，happy(vincent)和listen2Music(butch)，及其三个规则。KB3中的定义了三个名字和KB2中一样的谓词（happy，listen2Music，和playsAirGuitar），但是

其实现不同，特别是在playsAirGuitar的谓词中引入了一些新的含义。首先，分析这个规则：

playsAirGuitar(vincent) :-

```
        listen2Music(vincent),
        happy(vincent).
```

其中主干部分有两项，或者说两个目标组成。这里最重要的是英文逗号字符，它分隔了目标listens2Music(vincent)和目标happy(vincent)。这是逻辑与在Prolog中的表现形式。所以，

可以这么理解：“Vincent弹吉他如果他听音乐并且他很快乐”。

所以，如果我们查询：

```
?- playsAirGuitar(vincent).
```

Prolog会回答false。这是因为，KB3包含happy(vincent)的事实，但是没有明确地包含listen2Music(vincent)，并且也不能被推导出来。所以KB3只能满足playsAirGuitar(vincent)

两个条件之一，所以查询失败，Prolog回答false。

顺便提及一下，空格在Prolog中是没有意义的，比如，我们也可以这么书写：

```
playsAirGuitar(vincent) :-
        happy(vincent),
            listen2Music(vincent).
```

这个和之前的定义是同样效果。Prolog提供了很高的书写自由度，便于我们书写出可读性高的程序代码。

接下来，分析KB3中有相同头部的两个规则：

```
playsAirGuitar(butch) :- happy(butch).
playsAirGuitar(butch) :- listen2Music(butch).
```

这里表达的意思是，Butch弹吉他如果他听音乐，或者他很高兴。即，多个有相同头部的规则是Prolog中逻辑或的表达方式。所以，如果我们查询：

```
?- playsAirGuitar(butch).
```

Prolog会回答true。虽然第一个规则对于这个查询没有作用（因为happy(butch)在KB3中不存在，也不能被推导），但是KB3包含了listen2Music(butch)，所以Prolog可以使用

假言推理：

```
playsAirGuitar(butch) :- listen2Music(butch).
```

推导出playsAirGuitar(butch)为真。

Prolog中存在另外一种方式表示逻辑或，可以使用如下的定义来替代之前的两个规则：

```
playsAirGuitar(butch) :-
        happy(butch);
        listen2Music(butch).
```

英文分号字符在Prolog中也表示逻辑或，所以这个单一规则的含义，和之前两个规则的含义是相同的。那么使用多个规则，还是使用英文分号，哪个更好呢？这个需要根据情况来判断。

一方面，使用分号会让Prolog代码的可读性变差，但是另一方面，使用分号后规则数量变少，使得处理效率更好。

在上面的学习中，可以看出Prolog中明确包括了很多逻辑标识，比如，:-表示“如果”；英文字符逗号“,”表示逻辑与；英文字符分号“;”表示逻辑或。而且，我们可以看到标准的

逻辑证明规则（假言推理）在Prolog中起到了重要作用。所以，我们可以开始理解，为什么“Prolog”这个名字是“Programming with logic”的简写了，：）。

Knowledge Base 4

下面是KB4，我们的第四个知识库的定义：

```
woman(mia).
woman(jody).
woman(yolanda).

loves(vincent, mia).
loves(marsellus, mia).
loves(pumpkin, honey_bunny).
loves(honey_bunny, pumpkin).
```

哈哈，这是一个相当无聊的知识库。这里没有规则，只有事实的集合。当然，我们可以第一次接触一个谓词中存在两个名字（这里指loves定义的关系）。

但是事实并非如此，这个例子的新意不在于知识库的定义，而是我们查询的方式。事实上，这是我们第一次接触Prolog的变量使用，如下：

```
?- woman(X).
```

X即代表一个变量（在Prolog中，大写字母开头的单词代表变量，这也是为什么我们之前的例子中，所有出现的字符都是小写字母开头的原因）。这里X不是一个具体的名字，

它更像一个信息的占位符号。即，这个查询就是问Prolog：告诉我们你知道都有哪些人是女士（woman）？

Prolog通过在KB4中从上至下遍历来回答这个查询，Prolog会试图找到（或者匹配）表达式woman(X)的信息。在KB4中，第一个事实是woman(mia)，所以Prolog会将X和mia合一，

这样可以完美地符合查询（顺便提及一下，这个处理流程中Prolog做了很多的操作：我们可以简单地理解为，Prolog将X初始化为mia，或者将X值绑定成mia）。Prolog会将结果返回，

如下：

```
X = mia
```

这里不仅仅说至少有一个满足查询的结果存在于KB4中，而且明确地告诉了这个结果值。这里Prolog不是回答true，而是实际给出能够满足查询的变量绑定（变

量初始化）。

但是，这不是结束。变量的要点是它们能够代表，或者说能够合一不同的符合查询的信息。而且有其他的women在知识库中，也满足这个查询。所以，我们可以输入英文字符”；“

继续查询下一个匹配的结果：

```
X = mia;
```

因为英文字符”；“代表是逻辑或，所以这个查询可以理解为：还有其他结果吗？所以Prolog有会继续遍历知识库（它会标记上一次结果的地点，并且从那里继续），寻找下一个可能

的结果，并且找到jody也满足，所以Prolog会回答：

```
X = mia;

X = jody
```

这里就告诉了我们基于KB4第二个符合查询的结果值。当然，如果我们继续输入英文字符”；“，Prolog会继续回答：

```
X = mia;

X = jody;

X = yolanda
```

但是当我们第三次输入英文字符”；“会发生什么？Prolog会回答false，没有其他合一的可能了。因为在KB4中没有women开头的事实了，剩余的4个规则都是关于loves关系的，

所以没有办法再对woman(X)进行合一。

接下来，我们尝试一个更为复杂的查询，如下：

```
?- loves(marsellus, X), woman(X).
```

前面已经介绍过，英文字符逗号“,”表示逻辑与，所以这个查询的含义是：是否有这样的X，它既能够满足Marsellus爱它，并且还是一名女士？如果你再看知识库，会发现：

Mia是一名女人（事实一），同时Marsellus爱Mia（事实5）。Prolog能够模拟这个能力，把结果找出来。即Prolog能够遍历整个知识库，并将X和Mia合一，使得我们查询中的两个

子查询都满足。最后，Prolog会回答：

```
X = mia
```

能够将变量和知识库中的信息进行合一是Prolog的核心。随着我们的深入学习，我们会发现很多Prolog的有趣思想——但是Prolog能够进行合一并返回变量绑定信息的能力是所有

这一切的关键点。

Knowledge Base 5

好的，我们之前介绍了变量，但是仅仅是在查询中使用到。其实变量也可以应用在知识库中。而且只有这样做，我们才能写出真正有用的Prolog程序。下面是

一个简单的例子，

知识库KB5的定义：

```
loves(vincent, mia).
loves(marsellus, mia).
loves(pumpkin, honey_bunny).
loves(honey_bunny, pumpkin).

jealous(X, Y) :- loves(X, Z), loves(Y, Z).
```

KB5包含了4个loves关系的事实和一个规则，这个规则是我们至今定义的最有趣的一个：它包括了三个变量（X，Y，Z），这个规则如何理解？

本质上来说，这个规则定义“情敌”的概念。可以这里理解，如果X爱Z，同时Y也爱Z，那么X和Y就是情敌（呵呵，这里的情敌仅仅限于学习，现实情况会复杂的多，：））。

关键的一点在于这是一个通用的陈述，其中不涉及具体的人，比如mia，pumpkin等——从某种程度来说，是指世界上的每个人。这就是抽象。

如果我们进行查询：

```
?- jealous(marsellus, W).
```

这个查询的含义是：是否存在这样的一个人W，他和Marsellus是“情敌”？Vincent就是这个人。如果你检查“情敌”的定义，你可以发现Marsellus和Vincent就是“情敌”，

因为他们两个人都爱同样的一个女士，即Mia。所以Prolog会回答：

```
W = vincent
```

分类: Prolog

标签: 事实, 规则, 查询, 假言推理, 谓词

« 上一篇: Haskell学习笔记二：自定义类型
» 下一篇: Learn Prolog Now 翻译 - 第一章 - 事实，规则和查询 - 第二节， Prolog语法介绍

posted on 2015-06-25 15:09 seaman.kingfall 阅读(1087) 评论(2) 编辑 收藏

Feedback

#1楼 2015-08-28 00:30 Benjamin Yan

翻译了这么多了，而且每天一篇，不能望其项背啊。

支持(0) 反对(0)

#2楼 2016-03-11 13:39 深蓝医生

学习!

支持(0) 反对(0)

相关博文：
· Learn Prolog Now 翻译 - 第二章 - 合一和证明搜索 - 第一节， 合一
· Learn Prolog Now 翻译 - 第一章 - 事实，规则和查询 - 第二节， Prolog语法介绍
· Learn Prolog Now 翻译 - 第五章 - 数字运算 - 第一节，Prolog中的数字运算
· Learn Prolog Now 翻译 - 第三章 - 递归 - 第二节，规则顺序，目标顺序，终止
· Learn Prolog Now 翻译 - 第六章 - 列表补遗 - 第一节，列表合并