Learn Prolog Now 翻译 - 第十一章 - 知识库相关操作和解决方案的收集 - 第一节， 知识库相关操作

Prolog中有四个知识库相关的操作命令：assert,retract,asserta,assertz。让我们学习它们是如何使用的。假设从一个空白的知识库开始，如果输入命令：

```
?- listing.
```

Prolog会简单地回复true，列表是空白的。

假设我们输入这个命令：

```
?- assert(happy(mia)).
```

Prolog会回复true（assert/1命令始终会成功）。但是重点不是这个命令能够成功，而是它对知识库带来的副作用。如果现在我们输入：

```
?- listing.
happy(mia).
```

即，知识库已经不再是空白的了：它现在包含了我们声明的一个事实。

假设我们继续输入四个assert命令：

```
?- assert(happy(vincent)).
true

?- assert(happy(marcellus)).
true

?- assert(happy(butch)).
true

?- assert(happy(vincent)).
true
```

如果我们现在查询知识库的内容：

```
?- listing.

happy(mia).
happy(vincent).
happy(marcellus).
happy(butch).
happy(vincent).
true
```

我们声明的所有事实现在都存在在知识库中了。注意happy(vincent)在知识库中存在两个，因为我们声明了两次，看上去是合理的。

我们使用的知识库操作实际上已经更新了谓词happy/1的含义。更通用地讲，知识库操作命令给予了我们在运行程序时更新谓词的能力。在运行期间更新谓词定义称为动态谓词，与之相对的是我们之前定义和使用的静态谓词。大多数Prolog

--Benjamin Yan

解释器都坚持认为应该显式地声明动态谓词。我们将会稍后介绍包含动态谓词的例子，现在让我们继续讨论知识库操作命令。

到此为止，我们只通过声明往知识库中添加了事实，但是我们也可以添加规则。假如我们想要声明一个规则说如果任何人很高兴，那么他就很天真，即：

```
naive(X) :- happy(X).
```

我们可以这么做：

```
assert((naive(X) :- happy(X))).
```

请注意这个命令的语法：我们声明的规则使用一对小括号括起来。如果我们现在问知识库有哪些内容：

```
happy(mia).
happy(vincent).
happy(marcellus).
happy(butch).
happy(vincent).

naive(A) :- happy(A).
```

现在我们已经了解如果声明新的信息到知识库中，我们应该也了解如果在不需要这些信息的时候，将它们从知识库中移除。存在一个和assert/1相反的谓词，名为retract/1来达到这个目的。比如，如果我们使用下面的命令：

```
?- retract(happy(marcellus)).
```

然后列出现在知识库中的所有内容：

```
happy(mia).
happy(vincent).
happy(butch).
happy(vincent).

naive(A) :- happy(A).
```

可以看到，happy(marcellus)这个事实已经被移除。

如果我们继续：

```
?- retract(happy(vincent)).
```

然后列出现在知识库中的所有内容：

```
happy(mia).
happy(butch).
happy(vincent).

naive(A) :- happy(A).
```

请注意第一个happy(vincent)，而且只有第一个这样的事实被移除。

如果想要移除我们定义的happy/1所有的相关信息，可以使用变量：

```
?- retract(happy(X)).
X = mia;
X = butch;
```

```
X = vincent;
false
```

现在的知识库中，只剩下一个规则：

```
?- listing.
naive(A) :- happy(A).
```

如果我们希望对声明的位置有更多的控制，这里有两个assert/1的变种，分别是：

1. assertz。将声明的内容放在知识库的最后。
2. asserta。将声明的内容放在知识库的开头。

比如，假设我们从一个空白知识库开始，然后给出如下的命令：

```
?- assert( p(b)), assertz(p(c)), asserta(p(a)).
```

然后列出知识库中所有的内容：

```
?- listing.

p(a).
p(b).
p(c).
true
```

知识库操作是一项有用的技术。特别是用于保存计算结果时，所以在以后再问相同的问题，我们就可以不用再重新计算一次：我们只需要在声明的事实中直接查询保存的结果即可。这种技术称为内存化，或者缓存，这种技术在一些应用中可以显著地提升性能。下面是如何使用这项技术的简单示例：

```
:- dynamic lookup/3.

add_and_square(X, Y, Res) :- lookup(X, Y, Res), !.
add_and_square(X, Y, Res) :- Res is (X + Y) * (X + Y),
assert(lookup(X, Y, Res)).
```

这个程序做了什么？基本上讲，它使用两个数字X和Y，将它们相加，然后进行平方运算得出结果。比如，我们查询：

```
?- add_and_square(3, 7, X).
X = 100
true
```

但是重点在于：程序如何实现？首先，需要注意的是我们已经声明lookup/3为一个动态谓词。我们需要在运行时能够修改lookup的定义。其次，请注意定义add_and_square/3时存在两个子句。其中第二个子句是数学运算，并且将结果使用lookup/3谓词保存到知识库中（即，缓存了运算结果）。第一个子句检查Prolog的当前知识库，看是否存在已经运算过的结果，如果存在，就简单地返回结果，并中断第二个子句的执行。

下面是程序运行的例子。假设我们进行另一个查询：

```
?- add_and_square(3, 4, Y).
Y = 49
true
```

如果我们现在查询知识库中存在的信息会发现已经包括了：

```
lookup(3, 7 ,100).
lookup(3, 4, 49).
```

如果我们再问Prolog关于3,4相加后平方的查询，将不会再进行计算，而是直接返回已经计算过的结果。

有一个问题：我们如何删除所有我们不再需要的事实，如果我们输入命令：

```
?- retract(lookup(X, Y, Z)).
```

Prolog将会一个一个搜索所有的事实，然后询问我们是否想要删除它们。但是存在一个更加简便的方式，使用下面的命令：

```
?- retract(lookup(_, _, _)).
```

这个命令将会移除知识库中所有lookup/3相关的事实。

关于知识库操作的应用，还有一些建议：虽然这是一项有用的技术，但是知识库操作能够导致一些不美观，难以理解的代码出现；如果你在一个存在很多回溯的程序中大量使用它们，理解程序含义会成为噩梦。它是Prolog中一项没有良好声明性，非逻辑的技术，我们需要非常小心地使用它。

分类: Prolog

标签: 知识库, 缓存

seaman.kingfall
关注 - 1
粉丝 - 4
+加关注

« 上一篇: Learn Prolog Now 翻译 - 第十章 - 中断和否定 - 第三节， 使用否定作为失败判定
» 下一篇: Learn Prolog Now 翻译 - 第十一章 - 知识库相关操作和解决方案的收集 - 第二节， 解决方案的收集

posted on 2015-08-06 13:16 seaman.kingfall 阅读(307) 评论(0) 编辑 收藏

相关博文：

· 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 - 第十二章 - 文件相关操作 - 第二节， 文件的读写
· SQL 相关操作收集
· Learn Prolog Now 翻译 - 第十二章 - 文件相关操作 - 第一节， 使用不同文件组织程序
· Learn Prolog Now 翻译 - 第十一章 - 知识库相关操作和解决方案的收集 - 第二节， 解决方案的收集