Learn Prolog Now 翻译 - 第九章 - 语句深究 - 第二节， 特殊表示法的语句

内容提要

- 算术相关的语句
- 列表语句

有时候语句会看上去非常地奇怪，但是Prolog内部能够正确地进行处理。比如，当我们比较 a 和 'a'时，作为人我们看到了两个不同的字符符号，但是Prolog会认为它们是同一个东西。而且事实上，Prolog还会认为其他很多情况下，两个字符串是相同的语句。为什么？因为这使得编程更加容易。有时候对Prolog有用的表示法，对于开发者来说不是那么友好。所以，如果能够让开发者在编程时使用对人来说较为友好的表示法，然后也让Prolog了解其内部优化的表示法，就是很好的事情。

算术相关的语句

之前介绍过的一些算术谓词就是很好的例子。正如我们在第五章提及的，+，-，*，/都是函子，算术表达式比如，2+3，都是语句。这不是一个类比，除了借助is/2谓词的帮助，算术表达式能够进行计算的事实，Prolog会把字符串（比如 2+3）看做是普通的复杂语句，下面的查询会让这个过程更加清晰：

```
? 2 + 3 == +(2, 3).
true

? +(2, 3) == 2 + 3.
true

?- 2 - 3 == -(2, 3).
true

?- *(2, 3) == 2 * 3.
true

?- 2 * (7 + 2) == * (2, +(7, 2)).
true
```

简而言之，熟悉的算术中缀表达法是为了符合我们的使用习惯，Prolog不会认为这些和普通的语句有什么不同。

类似地可以使用的算术语句包括比较相关的谓词：<, =<, =:=, ==, >, >= ：

```
?- (2 < 3) == <(2, 3).
true

?- (2 =< 3) == =<(2, 3).
true

?- (2 =:= 3) == =:=(2, 3).
true

?- (2 =\= 3) == =\=(2, 3).
true

?- (2 > 3) == >(2, 3).
true

?- (2 >= 3) == >=(2, 3).
true
```

这些例子展示了为什么有符合我们习惯的算术表达式的好处（你情愿不得不使用类似：=:=(2,3)的表达式么？）。注意，我们在语句等式判断的左右两边加上了括号，比如，不会这样查询：

```
?- 2 =:= 3 == =:=(2,3).
```

而是这样查询：

```
?- (2 =:= 3) == =:=(2, 3).
```

为什么？因为Prolog会发现查询，2 =:= 3 == =:=(2,3)有二义性。因为不能判断这个表达式是：(2 =:= 3) == =:=(2, 3)，还是： 2 =:= (3 == =:=(2,3))，所以我们必须使用小括号明确含义。

最后，我们已经介绍了一些看上去很相像的符号，比如=， ==，和 =:=（还有=, ==, ==），如下是总结：

= 合一谓词。当参数能够合一时成功，否则失败。

= 反合一谓词。当=失败时成功，=成功时失败。

== 判等谓词。如果参数相等时成功，否则失败。

== 反判等谓词。==失败时成功，==成功时失败。

=:= 算术相等谓词，当参数为相等的整数时成功，否则失败。

== 反算术相等谓词，当=:=失败时成功，=:=成功时失败。

列表语句

列表是Prolog中另外一个有内部表示法的例子。Prolog为使用者提供了很友好的外部表示方法（即，中括号表示法[]），事实上，因为Prolog同时还提供了"|"构建符号，所以存在很多效果相同的不同外部表示方法，如下：

```
?- [a, b, c, d] == [a | [b, c, d]].
true

?- [a, b, c, d] == [a, b | [c, d]].
true

?- [a, b, c, d] == [a, b, c | [d]].
true

?- [a, b, c, d] == [a, b, c, d | []].
true
```

但是Prolog内部如果表示列表的？事实上，它会把列表看作由两个特殊的语句构成，其一是：[]，代表空列表，其二是："."（英文字符句号），一个元数为2的函子，用于构建非空列表，语句 []和 . 被称为列表构造器。

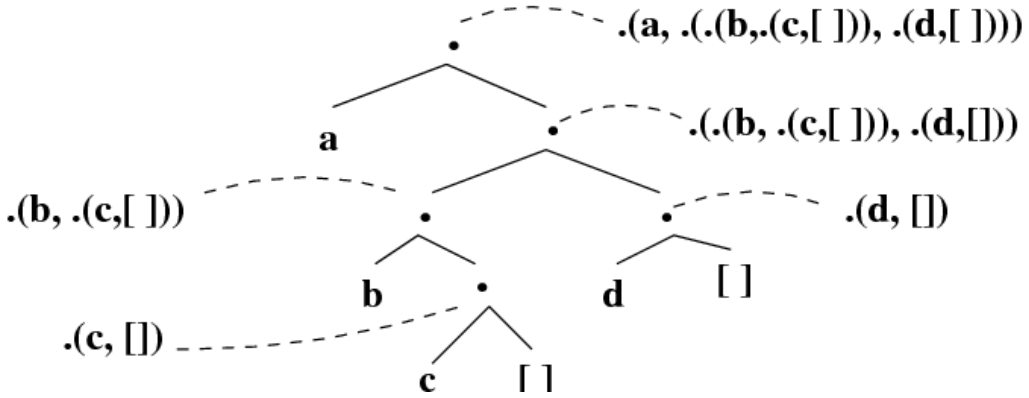
如下是这些列表构造器如何构建列表的。无需多说，这个定义肯定是递归的：

- 空列表是[]，它的长度为0.
- 非空列表是由形式为：.(term, list)构成的语句，其中term是任意Prolog的语句，list是任意列表，如果list的长度是n，那么.(term, list)的长度为 n+1。

通过下面的例子确保完全理解这个定义：

```
?- .(a, []) == [a].
true

?- .(f(d,e), []) == [f(d,e)].
true

?- .(a, .(b, [])) == [a, b].
true

?- .(a, .(b, .(f(d,e), []))) == [a, b, f(d,e)].
true

?- .(.(a,[]), []) == [[a]].
true

?- .(.(.(a, []), []), []) = [[[a]]].
true

?- .(.(a, .(b, [])), []) == [[a, b]].
true

?- .(.(a, .(b, [])), .(c, [])) == [[a, b], c].
true

?- .(.(a, []), .(b, .(c, []))) == [[a], b, c].
true

?- .(.(a, []), .(.(b, .(c, [])), [])) == [[a], b, c].
true
```

Prolog的列表内部表示法没有外部中括号表示法那么友好，不过也不是像第一眼看上去的那么糟。事实上，它工作原理和"|"语法类似，它将列表表示为两个部分：列表的第一个元素（头部），和列表的剩余部分组成的列表（尾部）。内部表示法很像一棵树。树的内部节点，如果被标记了"."，都有两个子节点。在左端的子节点代表列表的头元素，在右端的子节点代表剩余的列表。比如，.(a, .(.(b, .(c, [])), .(d, []))），即[a, [b, c], d]的内部树看上去像下图的样子：



最后提及一下，Prolog非常的友好，不仅仅局限于我们能够使用习惯的表示法查询，它回答问题也使用便于我们阅读的方式：

```
?- .(f(d,e), []) = Y.
Y = [f(d,e)]
true

?- .(a, .(b, [])) X, Z = .(.(c,[]),[]), W = [1, 2, X].
X = [a, b]
Z = [[c]]
W = [1, 2, [a, b]]
true
```

分类: Prolog

标签: 算术语句, 列表语句

seaman.kingfall
关注 - 1
粉丝 - 4
+加关注

« 上一篇: Learn Prolog Now 翻译 - 第九章 - 语句深究 - 第一节， 语句的比较
» 下一篇: Learn Prolog Now 翻译 - 第九章 - 语句深究 - 第三节， 语句的检查

posted on 2015-07-25 10:01 seaman.kingfall 阅读(236) 评论(0) 编辑 收藏