

公告

昵称: seaman.kingfall
园龄: 4年3个月
粉丝: 4
关注: 1
[+加关注](#)

搜索

找找看

谷歌搜索

常用链接

[我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

我的标签

[练习题\(6\)](#)
[合一\(3\)](#)
[递归\(3\)](#)
[中断\(2\)](#)
[类型变量\(2\)](#)
[数字\(2\)](#)
[列表\(2\)](#)
[Haskell\(2\)](#)
[recursive\(2\)](#)
[比较\(2\)](#)
[更多](#)

随笔分类

[Haskell\(2\)](#)
[Prolog\(32\)](#)

随笔档案

[2015年8月 \(7\)](#)
[2015年7月 \(22\)](#)
[2015年6月 \(5\)](#)

最新评论

1. Re: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
学习!
--深蓝医生
2. Re: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
翻译了这么多了, 而且每天一篇, 不能望其项背啊。

Learn Prolog Now 翻译 - 第五章 - 数字运算 - 第一节, Prolog中的数字运算

内容提要

Prolog中的数字运算
Porlog运算的本质

Prolog中的数字运算

Prolog语言本身提供了一些基础的运算符号, 对整数进行一些操作 (即类似...-3, -2, -1, 0, 1, 2, 3等)。多数Prolog的实现同时也提供了一些工具对实数进行操作 (比如浮点数, 1.53, 6.35, 等等)。但是我们不会讨论浮点数, 因为浮点数在典型的Prolog应用中很少, 所以不是本文的重点。但是另一方面, 整数是在Prolog

中有典型应用场景的 (比如记录列表的长度), 所以掌握起应用是十分重要的。我们从Prolog处理简单的一些操作开始, 比如加减乘除。

| 数学运算 | Prolog表达式 |
|------------------|------------------------------|
| $6 + 2 = 12$ | <code>8 is 6 + 2.</code> |
| $6 * 2 = 12$ | <code>12 is 6 * 2.</code> |
| $6 - 2 = 4$ | <code>4 is 6 - 2.</code> |
| $6 - 8 = -2$ | <code>-2 is 6 - 8.</code> |
| $6 \div 2 = 3$ | <code>3 is 6 / 2.</code> |
| $7 \div 2$ 的余数为1 | <code>1 is mod(7, 2).</code> |

在Prolog中可以进行如下的查询:

```
?- 8 is 6 + 2.  
yes  
?- 12 is 6 * 2.  
yes  
?- -2 is 6 - 8.  
yes  
?- 3 is 6 / 2.  
yes  
?- 1 is mod(7,2).  
yes
```

--Benjamin Yan

阅读排行榜

1. Learn Prolog Now 翻译
- 第三章 - 递归 - 第一节,
递归的定义(1168)
2. 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
(1087)
3. Learn Prolog Now 翻译
- 第一章 - 事实, 规则和查询
- 第二节, Prolog语法介绍
(781)
4. Haskell学习笔记二: 自定义类型(767)
5. Learn Prolog Now 翻译
- 第六章 - 列表补遗 - 第一节,
列表合并(753)

评论排行榜

1. 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
(2)

推荐排行榜

1. Haskell学习笔记二: 自定义类型(1)
2. Learn Prolog Now 翻译
- 第三章 - 递归 - 第四节,
更多的实践和练习(1)

更重要的是, 我们可以将运算结果使用变量进行表示, 比如:

```
?- X is 6 + 2.
```

```
X = 8
```

```
?- X is 6 * 2.
```

```
X = 12
```

```
?- R is mod(7,2).
```

```
R = 1
```

而且, 我们可以使用在谓词定义中使用运算符号。这里有一个简单的例子, 定义一个谓词, `add_3_and_double/2`, 其中的参数都是整数类型; 这个谓词获取第一个参数,

加上3, 在乘以2, 并将其计算结果放到第二个参数中。我们定义这个谓词代码如下:

```
add_3_and_double(X, Y) :- Y is (X + 3) * 2.
```

使用Prolog查询, 得出结果如下:

```
?- add_3_and_double(1, X).
```

```
X = 8
```

```
?- add_3_and_double(2, X).
```

```
X = 10
```

另外, Prolog中运算符的优先级和我们平时使用的优先级一致, 当我们写: `3 + 2 * 4`, 我们知道的含义是: `3 + (2 * 4)`, 并不是: `(3 + 2) * 4`, Prolog也是遵循这种优先级:

```
?- X is 3 + 2 * 4.
```

```
X = 11
```

Prolog数字运算的本质

以上内容是Prolog数字运算的基础, 现在我们更深入地进行学习。最重要的是首先理解, `+`, `-`, `*`, `/`, `mod`, 并不能进行任何的运算, 事实上, 诸如`3+2`, `3*2`之类的

表达式只是Prolog中的复杂语句。这些语句的函子是: `+`, `-`, `*`, `/`, `mod`, 参数是数字 (比如3, 2)。实际上, Prolog不会对这类复杂语句进行特殊的处理, 比如, 如果

我们查询:

```
?- X = 3 + 2.
```

```
X = 3 + 2
```

```
true
```

即, Prolog只会简单地将变量X和复杂语句 $3+2$ 合一, 而不会进行运算。它只会执行谓词 $=/2$ 的常规作用: 进行合一。

类似地, 如果我们查询:

```
?- 3 + 2 * 5 = X
```

```
X = 3 + 2 * 5
```

```
true
```

同样地, Prolog只是将变量X和复杂语句 $3+2*5$ 进行绑定, 而不会计算出结果:

13. 促使Prolog真正进行数字运算的, 是我们之前使用的:

```
is
```

事实上, is会有特殊的作用, 它会发送给Prolog一个信号说: “嘿, 不要当成普通的复杂语句对待这个表达式, 请调用内置的运算方式进行计算!” 简而言之, 它会促使Prolog进行

非常规的动作。通常而言Prolog会很乐意只是将变量和语句合一, 毕竟这就是它的工作。然而, 数字运算由于很重要, 所以是被额外附加于Prolog的实现中。所以不必感到惊讶, 对

这个额外的功能肯定会有某些限制, 我们也应该知道这些限制。

首先, 运算表达式必须出现在 is 的右边。比如之前的例子:

```
?- X is 6 + 2.
```

```
X = 8
```

是正确的使用方式, 如果我们这样查询:

```
?- 6 + 2 is X
```

Prolog会报类似变量没有初始化的错误。

其次, 虽然我们可以在 is 右端都使用变量, 但是当需要进行计算时, 变量必须是已经被初始化的。如果变量没有初始化, 或者被初始化为错误的类型, 我们也会收到类似变量没有初始化

的错误提示。这是因为数字运算不会使用Prolog常规的合一和知识库搜索机制, 而是通过调用内置的数字运算模块进行特殊处理的, 如果给了数字运算模块错误的数据类型, 就会报错。

这里有一个例子, 回忆上面的“加3并且乘以2”的谓词:

```
add_3_and_double(X, Y) :- Y is (X + 3) * 2
```

当我们描述这个谓词时, 我们会说它将第一个参数加上3, 然后乘以2, 并且将结果保持到第二个参数中。比如, `add_3_and_double(3, X)` 会返回 `X = 12`。我们没有提及这个谓词能够从

反方向使用, 比如, 如果查询:

```
?- add_3_and_double(X, 12).
```

并且希望结果是: `X = 3`, 但是Prolog不会这么运行, 只会报错。为什么? 因为我们相当于在问: `12 is (X + 3) * 2`, 这里X出现在is右边, 并且没有被初始化。

最后是两个建议。正如我们之前已经提到, 对于Prolog而言, $3+2$ 只是一个语句, 对应的真正的语句是: $+(3,2)$, 表达式 $3+2$ 只是语法糖, 便于阅读和使用。这意味着, 我们能够进行如下

的查询:

```
?- X is +(3, 2).
```

```
X = 5
```

事实上, 我们甚至可以这么查询:

```
?- is(X, +(3,2)).
```

```
X = 5
```

这是因为, 对于Prolog而言, 表达式: $X \text{ is } +(3,2)$ 真正的语句是: $\text{is}(X, +(3,2))$ 。前者只是语法糖。

总结一下, Prolog的数字运算还是十分简单的。我们需要注意的是使用 “is” 促使Prolog进行运算, 并且需要计算的部分在 “is” 右端, 而且如果右端出现了变量, 必须是已经初始化的。但是

有一个更深入影响是: 通过这种形式为Prolog加入数字运算的能力, 在声明性和程序性上都会扩大Prolog程序的内涵。

分类: Prolog

标签: 数字, 运算

好文要顶

关注我

收藏该文



seaman.kingfall

关注 - 1

粉丝 - 4

+加关注

0

0

« 上一篇: Learn Prolog Now 翻译 - 第四章 - 列表 - 第四节, 练习题和答案

» 下一篇: Learn Prolog Now 翻译 - 第五章 - 数字运算 - 第二节, 数字运算与列表

posted on 2015-07-15 21:57 seaman.kingfall 阅读(526) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论, 请 [登录](#) 或 [注册](#), [访问网站首页](#)。

【推荐】超50万C++/C#源码: 大型实时仿真组态图形源码

【活动】看雪2019安全开发者峰会, 共话安全领域焦点

【培训】Java程序员年薪40W, 他1年走了别人5年的路

最新新闻:

- 微信公开课聚焦“增长”: 墨迹天气小程序DAU环比增100%
 - 知否 | 太空垃圾如何清理? 卫星测试用鱼叉击中太空垃圾碎片
 - 一线 | “美团配送”品牌发布: 对外开放配送平台 共享配送能力
 - 苍蝇落在食物上会发生什么? 让我们说的仔细一点
 - 科学家研究板块构造变化对海洋含氧量影响
- » 更多新闻...

Copyright © seaman.kingfall
Powered by: .Text and ASP.NET
Theme by: .NET Monster