

Seaman.h.zhang

博客园 :: 首页 :: 新随笔 :: 联系 :: 订阅  :: 管理 34 Posts :: 0 Stories :: 2 Comments :: 0 Trackbacks

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 - 第十章 - 中断和否定 - 第二节, 中断的运用

中断在Prolog的运用

上一节我们学习了中断是什么。但是在实际编程时如何使用, 为什么这个机制会有作用呢? 第一个例子, 让我们定义一个没有中断的谓词max/3, 其中所有的参数都是整数, 并且当第三个参数为前两个参数中较大的一个时, 谓词为真。比如, 查询:

```
?- max(2, 3, 3).
true

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

?- max(3, 3, 3).
true
```

以上都是为真的查询, 如果查询:

```
?- max(2, 3, 2).
false

?- max(2, 3, 5).
false
```

都是为假的查询。当然, 我们希望这个谓词在第三个参数为变量的情况下去使用, 即我们能够找到前两个参数的最大值:

```
?- max(2, 3, Max).
Max = 3
true

?- max(2, 1, Max).
Max = 2
true
```

现在, 可以很容易地实现这个谓词。如下是第一个尝试:

```
max(X, Y, Y) :- X <= Y.
max(X, Y, X) :- X > Y.
```

这是一个完美的程序, 我们可能到此为止了。但是, 我们不应该这样, 因为它还不够好。

问题是什么? 这里存在潜在的性能问题。假设这个谓词会被一个更大型的程序使用, max(3, 4, Y)会被调用。这个谓词会正确地得出结果: Y = 4。但是思考一下如果强制的回溯会导致什么? 这个谓词将会使用第二个子句重新进行满足。但是这是没有意义的: 3和4的最大值就是4, 而且只能是4。没有第二个解决方案了。换种说法: 在上面谓词中的两个子句是排他的: 如果第一个为真, 那么第二个就为假, 反之亦然。所以进行回溯尝试完全是浪费时间。

--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)

cut可以修正这个问题。我们应当坚持Prolog不要尝试所有的子句, 所以下面的代码可以做到:

```
max(X, Y, Y) :- X =< Y, !.  
max(X, Y, X) :- X > Y.
```

注意这是如何起作用的。如果max(X, Y, Y)被调用并且X =< Y时, Prolog会达到中断。在这种情况下, 第二个参数就是最大值, 而且就只有这种解决方案了, 中断将这个选择进行提交。在另一方面, 如果X =< Y 失败了, 那时Prolog会使用第二个子句进行尝试。

注意这个中断没有改变程序的含义。新的代码能够得到和旧代码完全一致的结果, 但是更加高效。事实上, 除了中断, 这个程序和之前的版本完全一致, 这就是中断有意义的使用场景。像这样不改变程序含义的中断, 有一个特殊的名称: 绿色中断。

但是有一些读者可能不喜欢这样的代码。毕竟, 第二个子句是不是有所冗余? 如果我们不得不使用这个子句, 那么我们已经知道第一个参数比第二个参数要大了。我们能否在中断的帮助下, 优化我们的代码呢? 让我们进行一下尝试, 这里是第一次(注: 错误的尝试)尝试:

```
max(X, Y, Y) :- X =< Y, !.  
max(X, Y, X).
```

注意这个版本除了移除了第二个子句的>测试外, 其他都是和之前绿色中断的版本一致的。这个版本如何? 对于一些查询而言, 是正确的。特别是当第三个参数是变量时, 得出的结果是正确的, 比如:

```
?- max(100, 101, X).  
X = 101  
true  
  
?- max(3, 2, X).  
X = 3  
true
```

然而, 和绿色中断版本不同的是: 新的max/3不能正确的工作。思考在三个参数都有值时会发生什么, 例如, 如果查询:

```
?- max(2, 3, 2).
```

显然这个查询是为假的。但是在我们新的版本中, 返回结果为真。为什么? 因为查询由于不能和第一个子句合一, 所以直接使用了第二个子句。这个查询将会和第二个子句合一, 所以查询为真。所以移除第二个子句的>测试, 并不是一个明智的做法。

但是这里存在另外一种方式。新版本代码的问题简单地说是由于在到达中断前执行了变量合一。假设我们将变量处理得更加智能(使用三个变量代替两个), 并且在中断后显式地进行合一:

```
max(X, Y, Z) :- X =< Y, !, Y = Z.  
max(X, Y, X).
```

通过验证, 这个版本的程序能够正常工作, 并且(正如我们的期望)它避免了之前绿色中断版本中第二个子句的显式比较。

但是这个新的版本和之前绿色中断版本之间有一个重要的区别: 新版本中的中断是被称为红色中断的典型例子。技术上来说, 此类中断存在潜在的危险。为什么? 因为如果我们去掉此类中断, 我们不能达到完全一致的程序。即, 如果我们

移除中断, 程序将无法得出两个参数中的最大值。从另外一个角度理解, 这个中断的存在对于程序的正确性是不可或缺的。(这和绿色中断不同——绿色中断都是提高性能) 因为红色中断是不可或缺中断, 它们的存在意味着包含它们的程序没有完全的声明性。现在, 红色中断在某些情况下是有用的, 但是要小心。使用它们可能导致潜在的错误, 或者导致代码难以调试。

那么, 应该如何做呢? 建议如下: 尝试并且获得清晰的, 没有中断的程序, 同时只是通过中断提高其性能。如果可能, 尽量使用绿色中断。红色中断只有在十分必要时才使用, 同时在红色中断处显式地给出注释。通过这种方式, 将最大化地平衡代码的清晰的声明性和高效的程序性。

分类: Prolog

标签: 绿色中断, 红色中断

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 - 第十章 - 中断和否定 - 第一节, 中断

» 下一篇: 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

posted on 2015-08-03 12:57 seaman.kingfall 阅读(231) 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

相关博文:

- Learn Prolog Now 翻译 - 第十章 - 中断和否定 - 第一节, 中断
- Learn Prolog Now 翻译 - 第四章 - 列表 - 第二节, 列表成员
- 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 - 第十章 - 中断和否定 - 第三节, 使用否定作为失败判定
- Learn Prolog Now 翻译 - 第六章 - 列表补遗 - 第二节, 列表反转

最新新闻:

- 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
 - 日本程序员节假日全员加班? 都是“令和”惹的祸
- » 更多新闻...

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