

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 - 第六章 - 列表补遗 - 第三节, 练习题和答案

练习题 6.1

如果一个列表是由两段连续并且相同的元素段组成, 那么我们称之为双重列表。比如, `[a, b, c, a, b, c]`是双重列表 (因为它是由两个`[a, b, c]`构成), `[foo, gubble, foo, gubble]`也是双重列表。另一方面, `[foo, gubble, foo]`就不是双重列表。请写出一个谓词`doubled(List)`, 可以检查参数中的List是否为双重列表。

我的答案和解释

```
doubled(List) :- append(L, L, List).
```

思路: 由于双重列表是由两段相同的元素段组成, 所以可以看做两个相同的列表, 组成了双重列表, 借助之前定义的`append/3`谓词, 将第一个参数和第二个参数传入相同的变量, 第三个参数就是需要验证的List。由于只是验证List是否满足双重列表, 所以使用`append/3`不会有太多性能问题。

练习题 6.2

回文是指一个单词或者短语有相同的字母序列正向和反向组成, 比如, `'rotator'`, `'eve'`和`'nurses run'`都是回文。请写出一个谓词`palindrome(List)`, 检查输入的List是否是一个回文, 比如, 如果查询:

```
?- palindrome([r, o, t, a, t, o, r]).
```

或者查询:

```
?- palindrome([n, u, r, s, e, s, r, u, n]).
```

Prolog会回答true, 但是如果查询:

```
?- palindrome([n, o, t, h, i, s]).
```

Prolog会回答false。

我的答案和解释

```
palindrome(List) :- rev(List, R), R = List.
```

思路: 回文的描述如果从另外一个角度来解读的话, 就是列表和反转后的列表是相等的。所以或者输入列表的反转列表R, 然后R和List应该能够合一, 那么List就是一个回文。

练习题 6.3

请写出一个谓词`toptail(InList, OutList)`, 当InList预算少于2个时返回false, 否则删除第一个和最后一个元素, 并且将剩余的列表作为结果返回到第二个参数中。比如:

--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)

```
toptail([a], T).  
false  
  
toptail([a, b], T).  
T = []  
  
toptail([a, b, c], T).  
T = [b]
```

(提示: 这里可以考虑使用append/3)

我的答案和解释

```
toptail([H|T], Result) :- rev(T, [RevH|RevT]), rev(RevT,  
Result).
```

思路: 我自己没有想出使用append/3构建这个谓词的场景, 而是使用了rev/2, 首先正向获取输入列表的尾部, 然后反转尾部列表, 然后获取其尾部(即去掉了最后一个元素), 最后将RevT在反转回来, 得到结果。

练习题 6.4

请写出一个谓词last(List, X), 其中List为至少存在一个元素的列表, 并且X是列表List的最后一个元素时为真。请使用两种方式写出这个谓词的实现:

1. 使用rev/2完成last/2的定义。
2. 使用递归完成last/2的定义。

我的答案和解释

1. 使用rev/2的方式如下:

```
last(List, X) :- rev(List, [X|_]).
```

2. 使用递归的方式如下:

```
last([X], X).
```

```
last([_|T], X) :- last(T, X).
```

练习题 6.5

请写出一个谓词swapfl(List1, List2), 其中List1和List2都是列表, 如果List1和List2除了头尾元素是互相调换的, 其他部分都是相同的, 那么谓词会返回真。可以借助append/3, 或者递归, 或者其他谓词实现。

我的答案和解释

首先是使用append/3实现的版本, 这个版本判断为真的情况没有问题, 但是为假的情况, 会报错, 内存耗尽, 应该是实现的方式存在问题, 计算量太大了:

```
swapfl(List1, List2) :-  
    append(H1, Same, H1_Same),  
    append(H1_Same, H2, List1),  
    append(H2, Same, H2_Same),  
    append(H2_Same, H1, List2).
```

接下来是递归版本, 这个版本我个人觉得不错, 思路清晰明了, 性能也较好。基础子句是只包含两个元素的列表对比, 递归子句将其中相同的元素逐一去掉:

```
swapfl_recursive([H1, H2], [H2, H1]).
swapfl_recursive([H1, H | T1], [H2, H | T2]) :-
    swapfl_recursive([H1 | T1], [H2 | T2]).
```

练习题 6.6

这里有一个有趣的逻辑谜题: 一条街上有三栋相邻并且颜色不同的房子, 三种颜色是红、蓝和绿。不同国家的人住在不同的房子里, 他们都拥有不同的宠物。下面是关于他们的一些事实:

- 英国人住在红色房子里。
- 西班牙人的宠物是豹子 (jaguar) 。
- 日本人住在宠物是蛇的房子右边。
- 宠物是蛇的主人住在蓝色房子左边。

问题: 那个主人的宠物是斑马 (zebra)? 不要自己得出答案, 而是定义一个谓词zebra/1告诉宠物主人的国籍。

(提示: 思考如何使用Prolog描述房子和街道。为四个限制条件编写代码。也许member/2和sublist/2会有用。)

我的答案和解释

```
street(house(english, red, Pet1), house(spanish, Color1,
jaguar), house(japanese, Color2, Pet2)) :-
    member(Pet1, [zebra, snake]),
    member(Color1, [blue, green]),
    member(Pet2, [zebra, snake]), Pet2 \= Pet1, Pet2 \= snake,
    Color2 = blue, Color1 \= Color2.

zebra(X) :- street(house(X, _, zebra), _, _);
    street(_, house(X, _, zebra), _);
    street(_, _, house(X, _, zebra)).
```

运行结果如下:

```
?- zebra(X).
X = japanese
```

思路: 个人感觉这个解决方案不是很好, 主要想法是street由三个house构成, house是包含了人、宠物和房子颜色的复杂语句; 其中四个事实由规则的主干描述: 主要是限制每个条件的可能值, 题目中最后描述的2个事实, 可以稍微翻译一下, 比如“日本人住在宠物是蛇的房子右边”, 其理解为日本人的宠物不能是蛇, 这个理解是否由Prolog自动完成, 我还没有好的思路。

最后zebra/1谓词就是获取street/3中, house复杂语句中最后一项值为zebra的第一个参数值, 即国籍。

分类: Prolog

标签: 练习题, puzzles

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 - 第六章 - 列表补遗 - 第二节, 列表反转

» 下一篇: Learn Prolog Now 翻译 - 第九章 - 语句深究 - 第一节, 语句的比较

posted on 2015-07-23 13:57 seaman.kingfall 阅读(410) 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:**

- 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