生成表

1. 简介

正因为Scheme是Lisp语言大家族的一员，它擅长于处理表。你应该理解表以及表操作来掌握Scheme。表在我后面章节会解释的递归函数和高阶函数中扮演重要的角色。

在本章中，我会解释基本的表操作，比如cons，car，adr，list和quote。

2. Cons单元和表

2.1 cons单元

首先，让我解释一下表的元素cons单元（cons cells）。Cons单元是一个存放了两个地址的内存空间。Cons单元可用cons函数生成。

在前端键入(cons 1 2)

(cons 1 2)

;Value 11: (1 . 2)

系统返回(1 . 2)。函数cons如图表1所示那样给两个地址分配了内存空间，并把存放指向1的地址放在第一个空间，把存放指向2的地址放在另一个空间。存放指向1的地址的内存空间被称作car部分，对应的，存放指向2的地址的内存空间被称作cdr部分。car和cdr分别是Contents of the Address part of the Register和Contents of the Decrement part of the Register的简称【待翻译】。这些名字最初来源于Lisp首次被实现所使用的硬件环境中内存空间的名字。这些名字同时也表明cons单元的本质就是一个内存空间。cons这个名字是术语结构（construction）的简称。

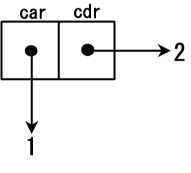


图 1

Cons单元也可以被串起来。

(cons 3 (cons 1 2))

;Value 15: (3 1 . 2)

(3 . (1 . 2))可以更方便的表示为(3 1 . 2)。这种情况的内存空间如图2所示

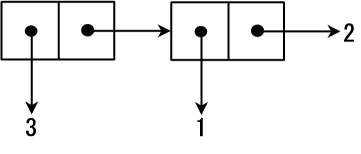


图 2 串连cons单元

Cons单元可以存放不同类型的数据，也可以嵌套。

(cons #\a (cons 3 "hello"))

;Value 17: (#\a 3 . "hello")

(cons (cons 0 1) (cons 1 2))

;Value 23: ((0 . 1) 2 . 3)

这是因为Scheme可以通过地址操作所有的数据。（#\c代表了一个字符c。例如，#\a就代表字符a）

2.2 表

表是cons单元通过用cdr部分连接下一个cons单元的开头实现的。表中包含的’()被称作空表。就算数据仅由一个cons单元组成，只要他的cdr单元是’()，那他就是一个表。图标三展示了表(1 2 3)的内存结构。

事实上，表可以像下面这样递归的定义：

1. ‘()是一个表

2. 如果ls是一个表且obj是某种类型的数据，那么(cons obj ls)也是一个表

正因为表是一种被递归定义的数据结构，将它用在递归的函数中显然是合理的。

2.3 原子

不适用cons单元的数据结构称为原子（atom）。数字，字符，字符串，向量和空表（’()）都是原子。’()既是原子，又是表。

练习1

3.引用

所有的记号都会依据Scheme的求值规则求值：所有的标记都会从最内层的括号依次向外层括号求值，且最外层括号返回的值讲作为S-表达式的值。一个被称为引用（quote）的形式可以用来组织标记被求值。他是用来程序原封不动的返回符号或者表本身的，而不是求值后变成的其他东西。【待议】

例如，(+ 2 3)会被求值为5，然而(quote (+ 2 3))则向程序返回(+ 2 3)本身。因为quote的使用频率很高，他被简写为’。

比如：

·’(+ 2 3)代表列表(+ 2 3)本身；

·’+代表符号+本身；

实际上，’()是对空表的引用，也就是说，尽管解释器返回()代表空表，你也应该用’()来表示空表。

3.1 特殊形式

Scheme有两种不同类型的操作符：其一是函数。函数会对所有的参数求值并返回值。另一种操作符则是特殊形式。特殊形式不会对所有的参数求值。除了quote，lambda，define，if，set！，等都是特殊形式。

4. car函数和cdr函数

返回一个cons单元的car部分和cdr部分的函数分别是car和cdr函数。如果cdr部分串连着cons单元，解释器会打印出整个cdr部分（原文是car，似有误）。如果cons单元的cdr部分不是’()，那么其值稍后亦会被展示。

(car '(1 2 3 4))

;Value: 1

(cdr '(1 2 3 4))

;Value 18: (2 3 4)

5. list函数

List函数使得我们可以构建包含数个元素的表。函数list有任意多个数的参数，且返回由这些参数构成的表。

(list)

;Value: ()

(list 1)

;Value 24: (1)

(list '(1 2) '(3 4))

;Value 25: ((1 2) (3 4))

(list 0)

;Value 26: (0)

(list 1 2)

;Value 27: (1 2)

啊