

## Class 27: Pairs and Pair Structures

**Held:** Tuesday, 10 October 2006

**Summary:** Today we visit some of the “behind the scenes” structures in Scheme, particularly the *pairs* that the `cons` procedure creates.

### Related Pages:

- EBoard.
- Lab: Pairs and Pair Structures.
- Reading: Pairs and Pair Structures.

### Notes:

- Extra credit for attending Thursday’s convo on *Interpretation of Race in American Museums*.
- Extra credit for attending Thursday’s Tuesday Extra on *Women in Computer Science at Grinnell*.

### Overview:

- Memory and Naming.
- Pairs and Cons cells.
- Why care?
- Lab.

## Behind the Scenes: Scheme and Memory

- Many of the issues we’ve been considering lately (e.g., naming) have a hidden dimension: The behavior is governed, in part, by the way that Scheme represents values “behind the scenes”, particularly how it stores values in the computer’s memory.
- You can think of Scheme as maintaining a few tables that map names to values.
- Each value must be *stored* somewhere in memory.
- Each value must be *tagged* with its type. (Memory is really only a sequence of 0’s and 1’s; we need a way to interpret that type.)
- Different types are stored in different ways.
  - Many basic values, such as moderately-sized numbers, are stored in a fixed amount of memory.
  - Other values get stored in a larger, value-specific amount of memory (along with explicit or implicit info about the size).
- I’ll draw some pictures in class.

## Cons Cells

- One set of interesting values in Scheme are the things that `cons` creates. We often call these *cons cells* or *pairs*
- The first value of a pair is a reference to the first parameter to `cons`. The second value is a reference to the second parameter.
- I'll also draw some pictures of these things in class.

## Dotted Pairs

- Although we've used lists as the second parts of each cons cell, you can use other values. If the second part of a cons cell is not a list, you get what is called a *dotted pair*.
- The shorthand for a dotted pair is  $(x . y)$ .
- You'll also see the dot at the end of a long sequence of cons cells.

## Why Care?

- *Why should we care about the underlying structure used for lists?*
- Because it helps explain why Scheme does some things the way it does.
- Because it helps us understand the hidden costs associated with some operations.
- Because it provides background for future discussions.

## Lab

- Any questions?
- Do the lab.
- Be prepared to reflect.

---

Copyright © 2006 Samuel A. Rebelsky. This work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.