

Class 05: Symbols and Lists

Held: Friday, 1 September 2006

Summary: Today we consider two of Scheme's most important types of data: symbolic values and lists.

Related Pages:

- EBoard.
- Lab: Symbols and Lists.
- Reading: Symbols in Scheme and Lists in Scheme.

Due

- HW2.

Assignments

- HW3: Reconfiguring Lists.

Notes:

- Extra credit for attending next week's convocation.
- Extra credit for attending next week's CS extra.
- Although the outline indicates some content, we'll focus on the lab today.

Overview:

- Symbolic values.
- Lists.
- Lab.

Symbols

- A key aspect of Scheme.
- Symbols are useful when you want to represent something atomically (as an idea that can't be pulled apart).
- The easiest way to build a symbol is with the quote operation.

```
> 'a
a
> a
reference to undefined identifier: a
> (quote a)
a
```

- Note that symbols and variables are different

```
> (define a 12)
> a
12
> 'a
a
```

- As you might be able to tell, the quote means “Take it literally; don’t attempt to interpret or evaluate it.”
 - Note that quote can be used for other types of values, although I will usually discourage you from doing so.

Lists

- The core of Scheme.
- Lists are ordered collections of information.
- Dynamic: Lists can "grow" and "shrink"
 - Or at least you can build larger or smaller lists from other lists.
- Lists look a lot like procedure application (intentionally).
- The simplest list: the empty list is `null` or `()`.
- You extend lists at the front (or, more precisely, build new lists) with

```
(cons thing-for-the-front rest-of-the-list)
```

- For example,

```
> (cons 'a null)
(a)
> (cons 'b (cons 'a null))
(b a)
```

- You can also create lists with `list`.

```
> (list 'a 'b 'c)
(a b c)
```

- You extract parts of lists with three operations (we’ll reflect on the naming later in the semester):
 - `car`: get the initial element of the list.
 - `cdr`: get all but the initial element of the list.
 - `list-ref`: get the *ith* element of the list.
- One disadvantage: Scheme lists look a lot like procedure applications, so it’s sometimes hard for beginning programmers to tell what’s going on.
 - You can always tell by context, but it takes some time to get used to the context.
- We’ll look at some behind-the-scenes issues in a few weeks.

Lab

- Do the lists lab.
 - We'll try to use the last five minutes of class to reflect.
 - You may have to complete the remainder of the lab on your own.
-

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.