University of Virginia 12 October 2011
cs1120: Introduction of Computing
Explorations in Language, Logic, and Machines

Class 21: Changing State

Upcoming Schedule

- **Regular office hours resume tomorrow**
  Thursday: 9:45-11am (Dave, Rice 507), 1-2:30pm (Joseph, Rice 1st),
  4:30-6pm (Jonathan, Rice 1st), 6-7:30pm (Jiamin, Rice 1st)
- **Reading:** if you haven't already, should soon finish reading the course book through the end of Chapter 9, and *The Information* through the end of Chapter 7.
- **Friday, 21 October (beginning of class): Problem Set 5**
- **Wednesday, 26 October: Quiz 3** (covers the course book through Chapter 10, *The Information* through the end of Chapter 8, and some other readings, details to be provided later)

Asymptotic Operators

Prove $0.001\,n^2 \notin O(9999n^{1.9})$.

Is a procedure with running time in $\Theta(n^{1.9})$ always better than one with running time in $\Theta(n^2)$?

State

Names and Places: define creates a new place, associates a name with that place, and stores a value in that place.

set! ("set bang") changes the value associated with a place

A *pure function* is a procedure that: (1) always produces the same result when applied to the same inputs, and (2) does not cause any other "noticeable" effects. What procedures have you used so far that are not pure functions?

Define a procedure nextx that produces the behavior shown:

```
> (define x 2)
> (nextx)
3
> (nextx)
4
```

Mutable Pairs

mcons – creates a *mutable* cons cell
(mcar m) – first part of a mutable cons cell
(mcdr m) – second part of a mutable cons cell

(set-mcar! *p v*) – Replaces the car of mutable cons *p* with *v*.
(set-mcdr! *p v*) – Replaces the cdr of mutable cons *p* with *v*.

How can we use these procedures to create a mutable list that cycles through the numbers 1, 2, and 3 (using only three mutable cons cells):

```
(mcar p) => 1
(mcar (mcdr p)) => 2
(mcar (mcdr (mcdr p))) => 3
(mcar (mcdr (mcdr (mcdr p)))) => 1
(mcar (mcdr (mcdr (mcdr (mcdr p))))) => 2
(mcar (mcdr (mcdr (mcdr (mcdr (mcdr p)))))) => 3
…
```