CSC 520, Spring 2020

Principles of Programming Languages

Michelle Strout

THE UNIVERSITY OF ARIZONA

Version: 1

March 2, 2020

Plan

- **Announcements**
  - HW5 is due Friday
  - The uscheme-trace version of the interpreter DOES have "record" syntactic sugar.
- **Last time**
  - Continuations as "gotos with arguments"
  - Example: Handling missing values (association list)
  - Example: Structuring a search (SAT solver)
- **Today**
  - Scheme Semantics
    - Stores
    - Lambdas evaluate to closures
    - Application

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

New Syntax, Values, Environments, and Evaluation Rules

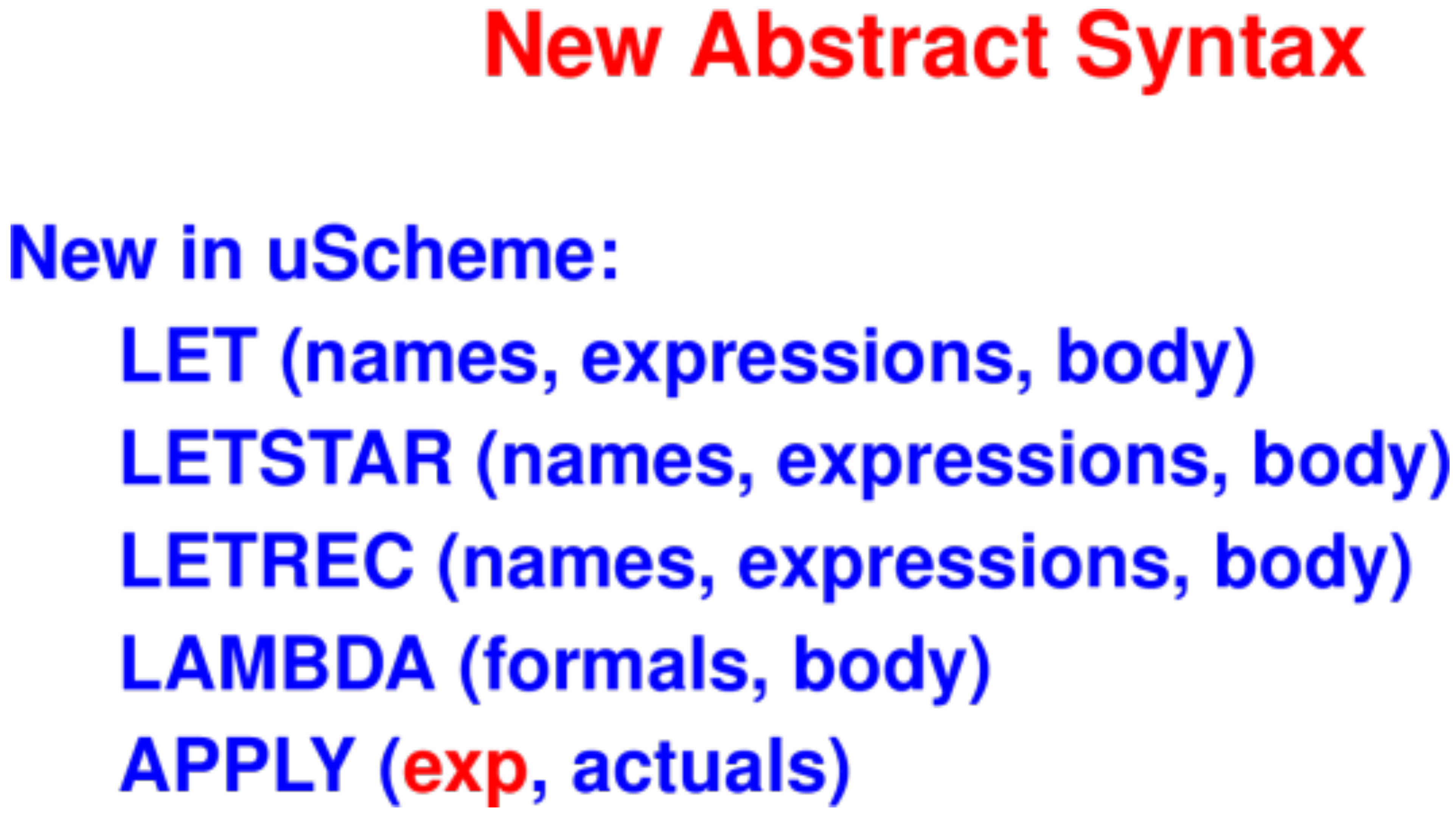
- **First four of five questions**

  1. What is the abstract syntax? Syntax categories?
  2. What are the values?
  3. What environments are there? What are names mapped to?
  4. How are terms evaluated?
  5. What's in the initial basis? Primitives and otherwise, what is built in?

- **Key changes from Impcore**
  - New constructs: let, lambda, application of more than just named functions
  - New values: cons cells and functions (closures)
  - A single kind of environment
  - New evaluation rules

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

New constructs



New Abstract Syntax

New in uScheme:

- LET (names, expressions, body)
- LETSTAR (names, expressions, body)
- LETREC (names, expressions, body)
- LAMBDA (formals, body)
- APPLY (exp, actuals)

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

New Values

- **Cons cells**
- **Functions (closures)**
- **Other values?**

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

A single kind of environment

- Environment maps names to mutable locations, NOT values
- A store maps locations to values
- Environments get copied (in closures)

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

New Evaluation Judgment

Judgment $\langle e, \rho, \sigma \rangle \Downarrow \langle v, \sigma' \rangle$

- Mappings in ρ never change
- ρ maps a name to a mutable location
- σ is the store (contents of every location)

Intuition of the compiler writer:

- ρ models the compiler's "symbol table"
- σ models the contents of registers and memory

Classic semantic technique

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

New Evaluation Rules

$$\frac{x \in \operatorname{dom} \rho \qquad \rho(x) \in \operatorname{dom} \sigma}{\langle \mathsf{VAR}(x), \rho, \sigma \rangle \Downarrow \langle \sigma(\rho(x)), \sigma \rangle} \quad (\textsc{Var})$$

$$\frac{x \in \operatorname{dom} \rho \qquad \rho(x) = \ell \qquad \langle e, \rho, \sigma \rangle \Downarrow \langle v, \sigma' \rangle}{\langle \mathsf{SET}(x, e), \rho, \sigma \rangle \Downarrow \langle v, \sigma'\{\ell \mapsto v\} \rangle} \quad (\textsc{Assign})$$

Semantics of Lambda

Key Issue: Values of free variables

Static scoping:

Where `lambda` occurs, “look outward” for ρ;

Capture that ρ for future reference.

$$\frac{}{\langle \mathsf{LAMBDA}(\langle x_1,\ldots,x_n\rangle,e),\rho,\sigma\rangle \Downarrow \langle (\!|\mathsf{LAMBDA}(\langle x_1,\ldots,x_n\rangle,e),\rho|\!),\sigma\rangle}\ (\textsc{MkClosure})$$

Create closure in C implementation of `eval` by

```
case LAMBDAX:
    return mkClosure(e->u.lambdax, env);
```

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

Function Application

- Which "even" is referenced with f is called?

```
(val even (lambda (x) (= 0 (mod x 2))))

(val f (lambda (y) (if (even y) 5 15)))

(val even 3)

(f 10)
```

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

Applying Closures

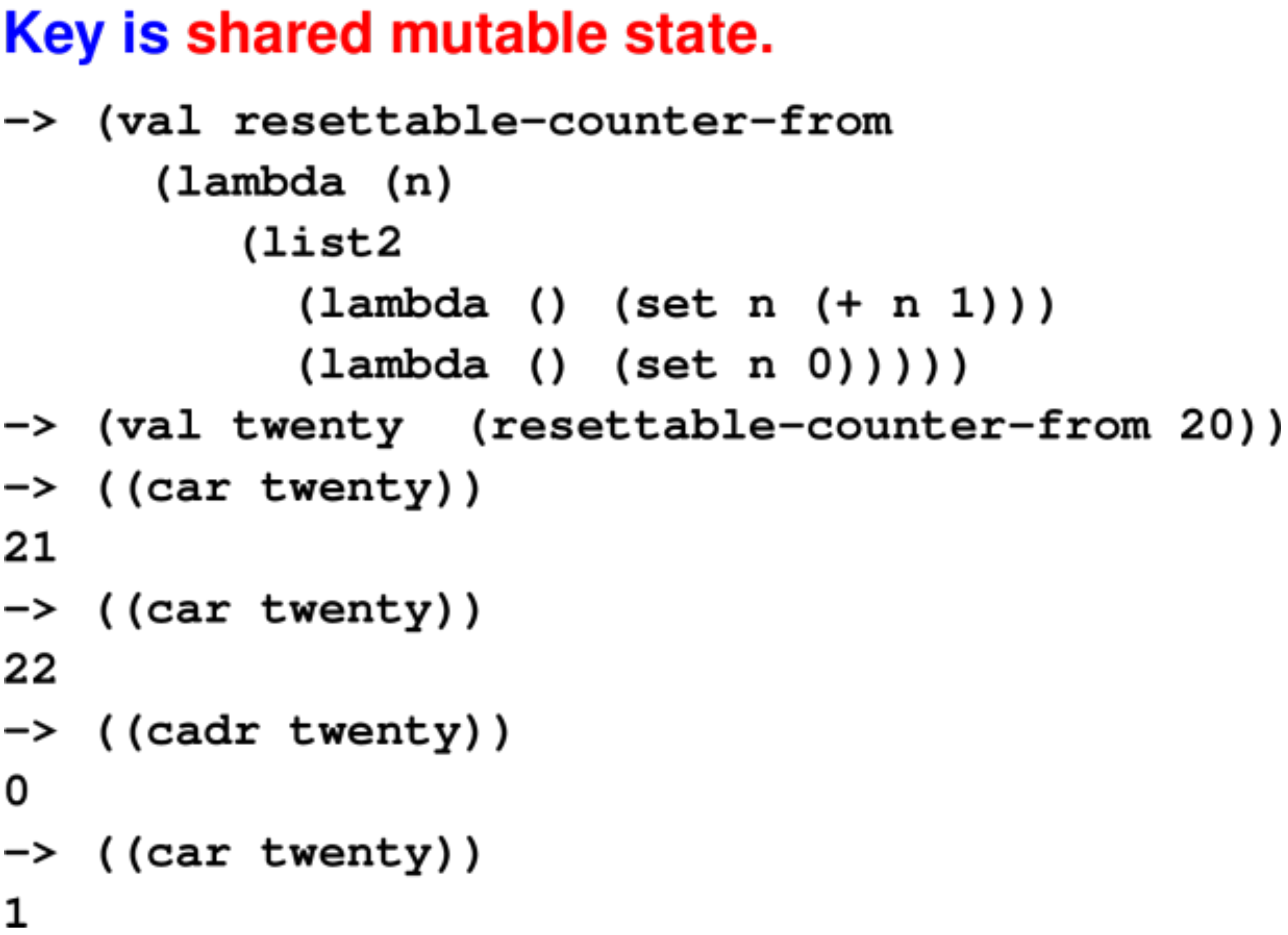
Captured environment for free variables

Arguments for bound variables (≡ formal parameters)

$$\frac{\begin{array}{c}\langle e,\rho,\sigma\rangle \Downarrow \langle (\!|\mathsf{LAMBDA}(\langle x_1,\ldots,x_n\rangle,e_c),\rho_c|\!),\sigma_0\rangle \\ \langle e_1,\rho,\sigma_0\rangle \Downarrow \langle v_1,\sigma_1\rangle \\ \vdots \\ \langle e_n,\rho,\sigma_{n-1}\rangle \Downarrow \langle v_n,\sigma_n\rangle \\ \ell_1,\ldots,\ell_n \notin \mathrm{dom}\,\sigma_n \\ \langle e_c,\rho_c\{x_1\mapsto \ell_1,\ldots,x_n\mapsto \ell_n\},\sigma_n\{\ell_1\mapsto v_1,\ldots,\ell_n\mapsto v_n\}\rangle \Downarrow \langle v,\sigma'\rangle\end{array}}{\langle \mathsf{APPLY}(e,e_1,\ldots,e_n),\rho,\sigma\rangle \Downarrow \langle v,\sigma'\rangle} \quad (\textsc{ApplyClosure})$$

```
nl = f.u.closure.lambda.formals;
return eval(f.u.closure.lambda.body,
            bindalloclist(nl, vl, f.u.closure.env));
```

Locations in Closures



Key is shared mutable state.

```
-> (val resettable-counter-from
      (lambda (n)
         (list2
           (lambda () (set n (+ n 1)))
           (lambda () (set n 0)))))
-> (val twenty  (resettable-counter-from 20))
-> ((car twenty))
21
-> ((car twenty))
22
-> ((cadr twenty))
0
-> ((car twenty))
1
```

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher

Closure Optimizations

- Keep closures on the stack
- Share closures
- Eliminate closures (when functions don't escape)

Slide Content Credits: Tufts Comp105 by Norman Ramsey and Kathleen Fisher