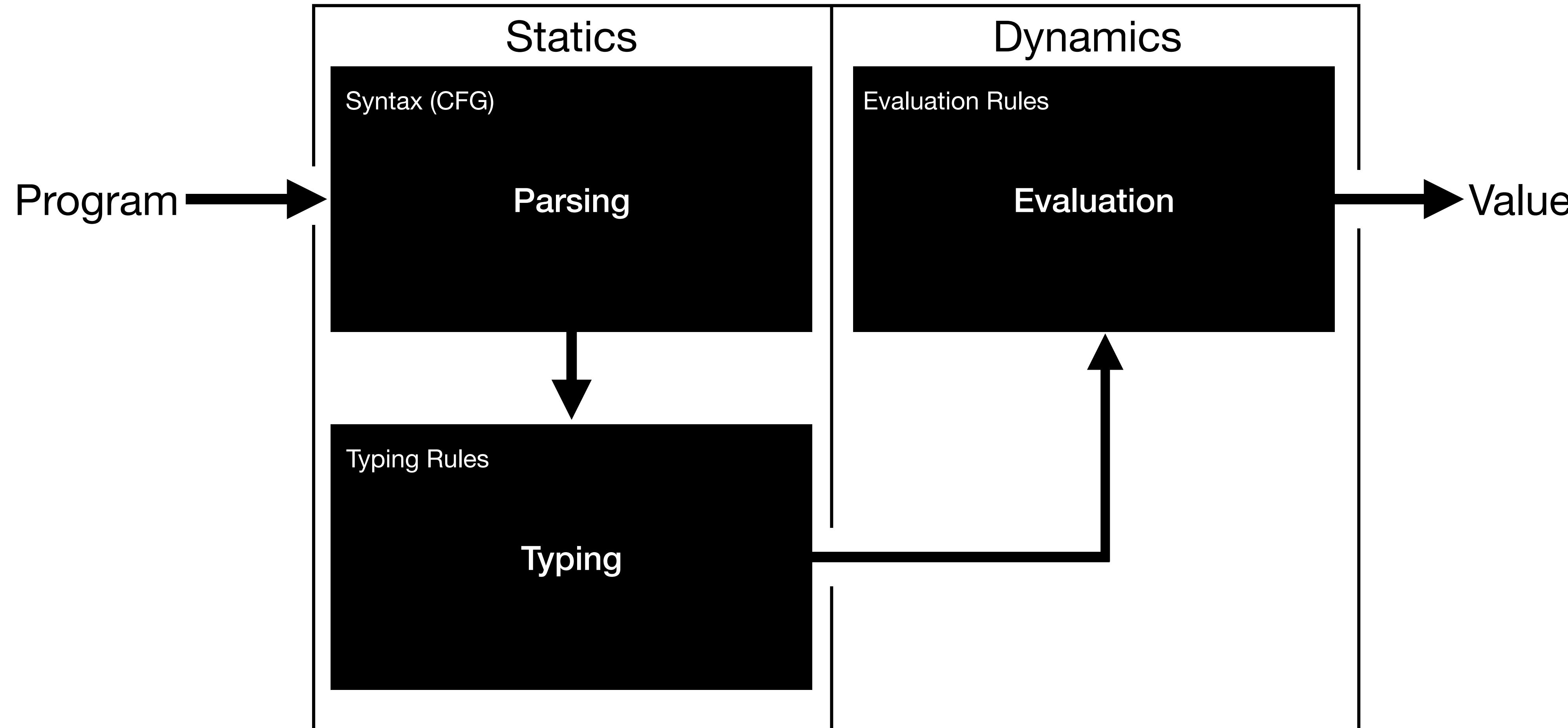


Object-Oriented Programming

Harley Eades III

Programming Language



Part 1: Base System

Base Syntax

Patterns

$$p ::= \begin{array}{l} x \\ | () \\ | T \\ | F \\ | 0 \\ | \text{succ}(x) \end{array}$$

Nats

$$n ::= \begin{array}{l} 0 \\ | \text{succ}(n) \end{array}$$

Terms

$$t ::= \begin{array}{l} T \\ | F \\ | 0 \\ | \text{succ}(t) \\ | x \\ | \text{fun } x : T \rightarrow \{t\} \end{array}$$

$$t_1 t_2$$

$$()$$

$$\text{match } t \{ \mid p_1 \rightarrow t_1 \mid \dots \mid p_i \rightarrow t_i \}$$

Values

$$v ::= \begin{array}{l} T \\ | F \\ | \text{fun } x : T \rightarrow \{t\} \\ | () \\ | n \end{array}$$

Types

$$T ::= \begin{array}{l} \text{Bool} \\ | \text{Nat} \\ | () \\ | T_1 \rightarrow T_2 \end{array}$$

Statics: Booleans

$$\frac{\Gamma \vdash t : \text{Bool} \quad \Gamma \vdash t_2 : T \quad \Gamma \vdash t_3 : T}{\Gamma \vdash \text{match } t \{ \mid \text{T} \rightarrow t_2 \mid \text{F} \rightarrow t_3 \} : T} \text{ If}$$

$$\frac{}{\Gamma \vdash \text{T} : \text{Bool}}^{\text{T}}$$
$$\frac{}{\Gamma \vdash \text{F} : \text{Bool}}^{\text{F}}$$

Statics: Nats

$$\frac{}{\Gamma \vdash 0 : \text{Nat}}^0 \quad \frac{\Gamma \vdash t : \text{Nat}}{\Gamma \vdash \text{succ}(t) : \text{Nat}}^{\text{succ}}$$
$$\frac{\Gamma \vdash t : \text{Nat} \quad \Gamma \vdash t_2 : T \quad \Gamma, x : \text{Nat} \vdash t_3 : T}{\Gamma \vdash \text{match } t \{ \mid 0 \rightarrow t_2 \mid \text{succ}(x) \rightarrow t_3 \} : T}^{\text{matchNat}}$$

Statics: Functions

$$\frac{}{\Gamma_1, x : T, \Gamma_2 \vdash x : T} \text{Var}$$

$$\frac{\Gamma \vdash t_1 : T_1 \rightarrow T_2 \quad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1 t_2 : T_2} \text{App}$$

$$\frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \text{fun } x : T_1 \rightarrow \{t\} : T_1 \rightarrow T_2} \text{Fun}$$

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \text{match } t_1 \{ \mid x \rightarrow t_2 \} : T} \text{Let}$$

Statics: Unit

$$\frac{}{\Gamma \vdash () : ()} \text{Unit}$$

$$\frac{\Gamma \vdash t_1 : () \quad \Gamma \vdash t_2 : T}{\Gamma \vdash \text{match } t_1 \{ | () \rightarrow t_2 \} : T} \text{Match}$$

Call-by-Value Dynamics: Match

$$\frac{}{\text{match } v \{ \mid x \rightarrow t \} \rightsquigarrow [v/x]t} \text{let}^\beta$$

$$\frac{}{\text{match } () \{ \mid () \rightarrow t \} \rightsquigarrow t} \text{unit}^\beta$$

$$\frac{}{\text{match } 0 \{ \mid 0 \rightarrow t_1 \mid \text{succ}(x) \rightarrow t_2 \} \rightsquigarrow t_1} \text{nat}^\beta_1$$

$$\frac{}{\text{match } \text{succ}(n) \{ \mid 0 \rightarrow t_1 \mid \text{succ}(x) \rightarrow t_2 \} \rightsquigarrow [n/x]t_2} \text{nat}^\beta_2$$

$$\frac{}{\text{match } T \{ \mid T \rightarrow t_1 \mid F \rightarrow t_2 \} \rightsquigarrow t_1} \text{if}^\beta_1$$

$$\frac{}{\text{match } F \{ \mid T \rightarrow t_1 \mid F \rightarrow t_2 \} \rightsquigarrow t_2} \text{if}^\beta_2$$

Call-by-Value Dynamics: Match

$$\frac{t \rightsquigarrow t'}{\text{match } t \{ \mid p_1 \rightarrow t_1 \mid \dots \mid p_i \rightarrow t_i \} \rightsquigarrow \text{match } t' \{ \mid p_1 \rightarrow t_1 \mid \dots \mid p_i \rightarrow t_i \}}^{\text{match}}$$

Call-by-Value Dynamics: Functions

$$\frac{t_1 \rightsquigarrow t'_1}{t_1 t_2 \rightsquigarrow t'_1 t_2} \text{ App1}$$

$$\frac{t_2 \rightsquigarrow t'_2}{v_1 t_2 \rightsquigarrow v_1 t'_2} \text{ App2}$$

$$\frac{}{(\text{fun } x : T \rightarrow \{t\}) v \rightsquigarrow [v/x]t} \beta$$

Part 2:Unit

Example: Side Effects

$$\frac{\Gamma \vdash t : \text{String}}{\Gamma \vdash \text{print } t : ()} \text{Print}$$

Outputting a string to the screen doesn't return a value, and so we can model this by returning the unit.

Statics: Unit

$$\frac{}{\Gamma \vdash () : ()} \text{Unit}$$

$$\frac{\Gamma \vdash t_1 : () \quad \Gamma \vdash t_2 : T}{\Gamma \vdash \text{match } t_1 \{ | () \rightarrow t_2 \} : T} \text{Match}$$

Sequencing

$$\frac{\Gamma \vdash t_1 : () \quad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1; t_2 : T} \text{ Seq}$$

```
let r = ref 7  
r := succ(!r); !r  
# 8 : Nat
```

Sequencing

$$\frac{\Gamma \vdash t_1 : () \quad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1; t_2 : T} \text{ Seq}$$

$t_1; t_2 = \text{match } t_1 \{ | () \rightarrow t_2 \}$

let $r = \text{ref } 7$
 $r := \text{succ}(!r); !r$

8 : Nat

Sequencing is match!

Call-by-Value Dynamics: Unit

$$\frac{}{\text{match } ()\{ \mid () \rightarrow t_2 \} \rightsquigarrow t_2} \text{Unit}\beta$$

$$\frac{t_1 \rightsquigarrow t'_1}{\text{match } t_1\{ \mid () \rightarrow t_2 \} \rightsquigarrow \text{match } t'_1\{ \mid () \rightarrow t_2 \}} \text{Match}$$

Part 3:Pairs

New Syntactic Forms: Adding Pairs

<u>Terms</u>	<u>Values</u>	<u>Types</u>
$t ::= T$	$v ::= T$	$T ::= \text{Bool}$
F	F	(T_1, T_2)
if t_1 then t_2 else t_3	$\text{fun } x : T \rightarrow \{t\}$	$T_1 \rightarrow T_2$
x	(v_1, v_2)	
$\text{fun } x : T \rightarrow \{t\}$		
$t_1 t_2$		
(t_1, t_2)		
$t.1$		
$t.2$		
let $x = t_1$ in t_2		

Example Programs

```
let twist = fun p : (Bool, Bool) → {(p.2,p.1)}  
in twist(T, F)
```

```
let second = fun p : (Bool → Bool, Bool) → {if p.2 then p.1 T else F}  
in second (fun x : Bool → {if x then F else T}, T)
```

Statics: Booleans

$$\frac{\Gamma \vdash t_1 : \text{Bool} \quad \Gamma \vdash t_2 : T \quad \Gamma \vdash t_3 : T}{\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T} \text{ If}$$

$$\frac{}{\Gamma \vdash T : \text{Bool}}^T$$
$$\frac{}{\Gamma \vdash F : \text{Bool}}^F$$

Statics: Functions

$$\frac{}{\Gamma_1, x : T, \Gamma_2 \vdash x : T} \text{Var}$$

$$\frac{\Gamma \vdash t_1 : T_1 \rightarrow T_2 \quad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1 t_2 : T_2} \text{App}$$

$$\frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \text{fun } x : T_1 \rightarrow \{t\} : T_1 \rightarrow T_2} \text{Fun}$$

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \text{let } x = t_1 \text{ in } t_2 : T_2} \text{Let}$$

Statics: Pairs

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma \vdash t_2 : T_2}{\Gamma \vdash (t_1, t_2) : (T_1, T_2)} \text{ Pair}$$

$$\frac{\Gamma \vdash t : (T_1, T_2)}{\Gamma \vdash t.1 : T_1} \text{ First}$$

$$\frac{\Gamma \vdash t : (T_1, T_2)}{\Gamma \vdash t.2 : T_2} \text{ Second}$$

Call-by-Value Dynamics: Bools

$$\frac{t_1 \rightsquigarrow t'_1}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \rightsquigarrow \text{if } t'_1 \text{ then } t_2 \text{ else } t_3} \text{ If}$$

$$\frac{\overline{\text{if T then } t_2 \text{ else } t_3 \rightsquigarrow t_2}^{\text{IfT}}}{\text{if F then } t_2 \text{ else } t_3 \rightsquigarrow t_3} \text{ IfF}$$

Call-by-Value Dynamics: Functions

$$\frac{t_1 \rightsquigarrow t'_1}{t_1 t_2 \rightsquigarrow t'_1 t_2} \text{App}^1$$

$$\frac{t_2 \rightsquigarrow t'_2}{v_1 t_2 \rightsquigarrow v_1 t'_2} \text{App}^2$$

$$\frac{t_1 \rightsquigarrow t'_1}{\text{let } x = t_1 \text{ in } t_2 \rightsquigarrow \text{let } x = t'_1 \text{ in } t_2} \text{Let}$$

$$\frac{}{\text{let } x = v \text{ in } t \rightsquigarrow [v/x]t} \text{Let}^\beta$$

$$\frac{}{(\text{fun } x : T \rightarrow \{t\}) v \rightsquigarrow [v/x]t} \beta$$

Call-by-Value Dynamics: Pairs

$$\frac{t_1 \rightsquigarrow t'_1}{(t_1, t_2) \rightsquigarrow (t'_1, t_2)} \text{Pair1}$$

$$\frac{t \rightsquigarrow t'}{t.1 \rightsquigarrow t'.1} \text{Proj1}$$

$$\frac{}{(v_1, v_2).1 \rightsquigarrow v_1} \text{Pair}\beta_1$$

$$\frac{t_2 \rightsquigarrow t'_2}{(v_1, t_2) \rightsquigarrow (v_1, t'_2)} \text{Pair2}$$

$$\frac{t \rightsquigarrow t'}{t.2 \rightsquigarrow t'.2} \text{Proj2}$$

$$\frac{}{(v_1, v_2).2 \rightsquigarrow v_2} \text{Pair}\beta_2$$

Part 4:Tuples

New Syntactic Forms: Adding Tuples

Terms

$t ::=$

- | T
- | F
- | if t_1 then t_2 else t_3
- | x
- | fun $x : T \rightarrow \{t\}$
- | $t_1 t_2$
- | (t_1, \dots, t_i)
- | match $t_1\{(x_1, \dots, x_i) \rightarrow t_2\}$
- | let $x = t_1$ in t_2

Values

$v ::=$

- | T
- | F
- | fun $x : T \rightarrow \{t\}$
- | (v_1, \dots, v_i)

Types

$T ::=$

- | Bool
- | (T_1, \dots, T_i)
- | $T_1 \rightarrow T_2$

Example: Tuple

(T, F, T, F, F)

```
fun (p : (Bool, Bool, Bool)) → {  
    match p {  
        (x, y, z) → if x  
                    then if y  
                        then z  
                        else False  
                    else False  
    }  
}
```

Statics: Booleans

$$\frac{\Gamma \vdash t_1 : \text{Bool} \quad \Gamma \vdash t_2 : T \quad \Gamma \vdash t_3 : T}{\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T} \text{ If}$$

$$\frac{}{\Gamma \vdash T : \text{Bool}}^T$$
$$\frac{}{\Gamma \vdash F : \text{Bool}}^F$$

Statics: Functions

$$\frac{}{\Gamma_1, x : T, \Gamma_2 \vdash x : T} \text{Var}$$

$$\frac{\Gamma \vdash t_1 : T_1 \rightarrow T_2 \quad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1 t_2 : T_2} \text{App}$$

$$\frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \text{fun } x : T_1 \rightarrow \{t\} : T_1 \rightarrow T_2} \text{Fun}$$

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \text{let } x = t_1 \text{ in } t_2 : T} \text{Let}$$

Statics: Tuples

$$\frac{\Gamma \vdash t_1 : T_1 \dots \Gamma \vdash t_i : T_i}{\Gamma \vdash (t_1, \dots, t_i) : (T_1, \dots, T_i)} \text{ Tuple}$$

$$\frac{\Gamma \vdash t_1 : (T_1, \dots, T_i) \quad \Gamma, x_1 : T_1, \dots, x_i : T_i \vdash t_2 : T}{\Gamma \vdash \text{match } t_1 \{(x_1, \dots, x_i) \rightarrow t_2\} : T} \text{ Match}$$

Call-by-Value Dynamics: Bools

$$\frac{t_1 \rightsquigarrow t'_1}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \rightsquigarrow \text{if } t'_1 \text{ then } t_2 \text{ else } t_3} \text{ If}$$

$$\frac{\overline{\text{if T then } t_2 \text{ else } t_3 \rightsquigarrow t_2}^{\text{IfT}}}{\text{if F then } t_2 \text{ else } t_3 \rightsquigarrow t_3} \text{ IfF}$$

Call-by-Value Dynamics: Functions

$$\frac{t_1 \rightsquigarrow t'_1}{t_1 t_2 \rightsquigarrow t'_1 t_2} \text{App}^1$$

$$\frac{t_2 \rightsquigarrow t'_2}{v_1 t_2 \rightsquigarrow v_1 t'_2} \text{App}^2$$

$$\frac{t_1 \rightsquigarrow t'_1}{\text{let } x = t_1 \text{ in } t_2 \rightsquigarrow \text{let } x = t'_1 \text{ in } t_2} \text{Let}$$

$$\frac{}{\text{let } x = v \text{ in } t \rightsquigarrow [v/x]t} \text{Let}^\beta$$

$$\frac{}{(\text{fun } x : T \rightarrow \{t\}) v \rightsquigarrow [v/x]t} \beta$$

Call-by-Value Dynamics: Tuples

$$\frac{t_{i+1} \rightsquigarrow t'_{i+1}}{(v_1, \dots, v_i, t_{i+1}, \dots, t_j) \rightsquigarrow (v_1, \dots, v_i, t'_{i+1}, \dots, t_j)} \text{ Tuple}$$

Call-by-Value Dynamics: Tuples

$$\frac{t_1 \rightsquigarrow t'_1}{\text{match } t_1\{(x_1, \dots, x_i) \rightarrow t_2\} \rightsquigarrow \text{match } t'_1\{(x_1, \dots, x_i) \rightarrow t_2\}} \text{ Match}$$

$$\frac{}{\text{match } (v_1, \dots, v_i)\{(x_1, \dots, x_i) \rightarrow t_2\} \rightsquigarrow [v_1/x_1] \cdots [v_i/x_i] t_2} \text{ Tuple}^\beta$$

Part 5:Records

New Syntactic Forms: Adding Records

Suppose we have a set of labels \mathcal{L}

Terms

$$\begin{aligned} t ::= & \top \\ | & \text{F} \\ | & \text{if } t_1 \text{ then } t_2 \text{ else } t_3 \\ | & x \\ | & \text{fun } x : T \rightarrow \{t\} \\ | & t_1 t_2 \\ | & (l_1 = t_1, \dots, l_i = t_i) \\ | & t.l \\ | & \text{let } x = t_1 \text{ in } t_2 \end{aligned}$$

Values

$$\begin{aligned} v ::= & \top \\ | & \text{F} \\ | & \text{fun } x : T \rightarrow \{t\} \\ | & (l_1 = v_1, \dots, l_i = v_i) \end{aligned}$$

Types

$$\begin{aligned} T ::= & \text{Bool} \\ | & (l_1 : T_1, \dots, l_i : T_i) \\ | & T_1 \rightarrow T_2 \end{aligned}$$

Example: Records

$(x = 2, y = 5) : (x : \text{Int}, y : \text{Int})$

$(\text{desc} = \text{"brake rotor"}, \text{partno} = 3947, \text{cost} = 250) : (\text{desc} : \text{String}, \text{partno} : \text{Int}, \text{cost} : \text{Float})$

Statics: Booleans

$$\frac{\Gamma \vdash t_1 : \text{Bool} \quad \Gamma \vdash t_2 : T \quad \Gamma \vdash t_3 : T}{\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T} \text{ If}$$

$$\frac{}{\Gamma \vdash T : \text{Bool}}^T$$

$$\frac{}{\Gamma \vdash F : \text{Bool}}^F$$

Statics: Functions

$$\frac{}{\Gamma_1, x : T, \Gamma_2 \vdash x : T} \text{Var}$$

$$\frac{\Gamma \vdash t_1 : T_1 \rightarrow T_2 \quad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1 t_2 : T_2} \text{App}$$

$$\frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \text{fun } x : T_1 \rightarrow \{t\} : T_1 \rightarrow T_2} \text{Fun}$$

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \text{let } x = t_1 \text{ in } t_2 : T} \text{Let}$$

Statics: Records

$$\frac{\Gamma \vdash t_1 : T_1 \dots \Gamma \vdash t_i : T_i}{\Gamma \vdash (l_1 = t_1, \dots, l_i = t_i) : (l_1 : T_1, \dots, l_i : T_i)} \text{Record}$$

$$\frac{\Gamma \vdash t : (l_1 : T_1, \dots, l_i : T_i)}{\Gamma \vdash t . l_i : T_i} \text{Proj}$$

Call-by-Value Dynamics: Bools

$$\frac{t_1 \rightsquigarrow t'_1}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \rightsquigarrow \text{if } t'_1 \text{ then } t_2 \text{ else } t_3} \text{ If}$$

$$\frac{\overline{\text{if T then } t_2 \text{ else } t_3 \rightsquigarrow t_2}^{\text{IfT}}}{\text{if F then } t_2 \text{ else } t_3 \rightsquigarrow t_3} \text{ IfF}$$

Call-by-Value Dynamics: Functions

$$\frac{t_1 \rightsquigarrow t'_1}{t_1 t_2 \rightsquigarrow t'_1 t_2} \text{App}^1$$

$$\frac{t_2 \rightsquigarrow t'_2}{v_1 t_2 \rightsquigarrow v_1 t'_2} \text{App}^2$$

$$\frac{t_1 \rightsquigarrow t'_1}{\text{let } x = t_1 \text{ in } t_2 \rightsquigarrow \text{let } x = t'_1 \text{ in } t_2} \text{Let}$$

$$\frac{}{\text{let } x = v \text{ in } t \rightsquigarrow [v/x]t} \text{Let}^\beta$$

$$\frac{}{(\text{fun } x : T \rightarrow \{t\}) v \rightsquigarrow [v/x]t} \beta$$

Call-by-Value Dynamics: Records

$$\frac{t_{i+1} \rightsquigarrow t'_{i+1}}{(l_1 = v_1, \dots, l_i = v_i, l_{i+1} = t_{i+1}, \dots, l_j = t_j) \rightsquigarrow (l_1 = v_1, \dots, l_i = v_i, l_{i+1} = t'_{i+1}, \dots, l_j = t_j)} \text{ Record}$$

Call-by-Value Dynamics: Records

$$\frac{t \rightsquigarrow t'}{t . l_i \rightsquigarrow t' . l_i} \text{ Proj} \quad \frac{}{(l_1 = v_1, \dots, l_i = v_i) . l_j \rightsquigarrow v_j} \text{ Record}\beta$$

Part 6: Mutable References

Up until now, all of the languages we have studied have been **pure**.

Up until now, all of the languages we have studied have been **pure**.

pure: a programming language without **computational effects**.

Up until now, all of the languages we have studied have been **pure**.

pure: a programming language without **computational effects**.

computational effect: programs that interact or modify with the outside world

Computational Effects

- mutable references
- input/output
- networking
- non-local transfers of control
- inter-process synchronization

Computational Effects

- mutable references
- input/output
- networking
- non-local transfers of control
- inter-process synchronization

Key Concepts

- allocation (references)
- assignment operator
- explicit dereferencing
- stores (or heaps)

Allocation

Allocating a reference: ref 5 : Ref Nat

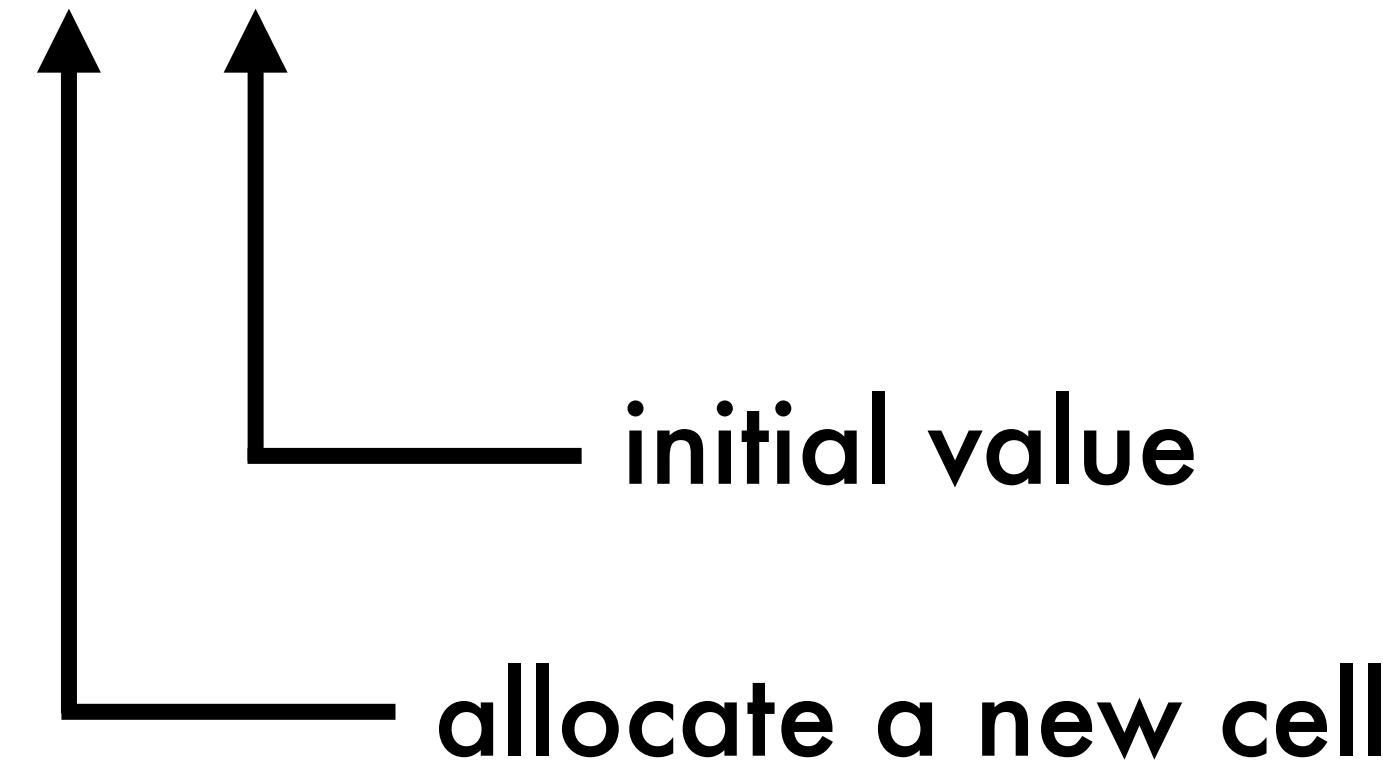
Allocation

Allocating a reference: ref 5 : Ref Nat



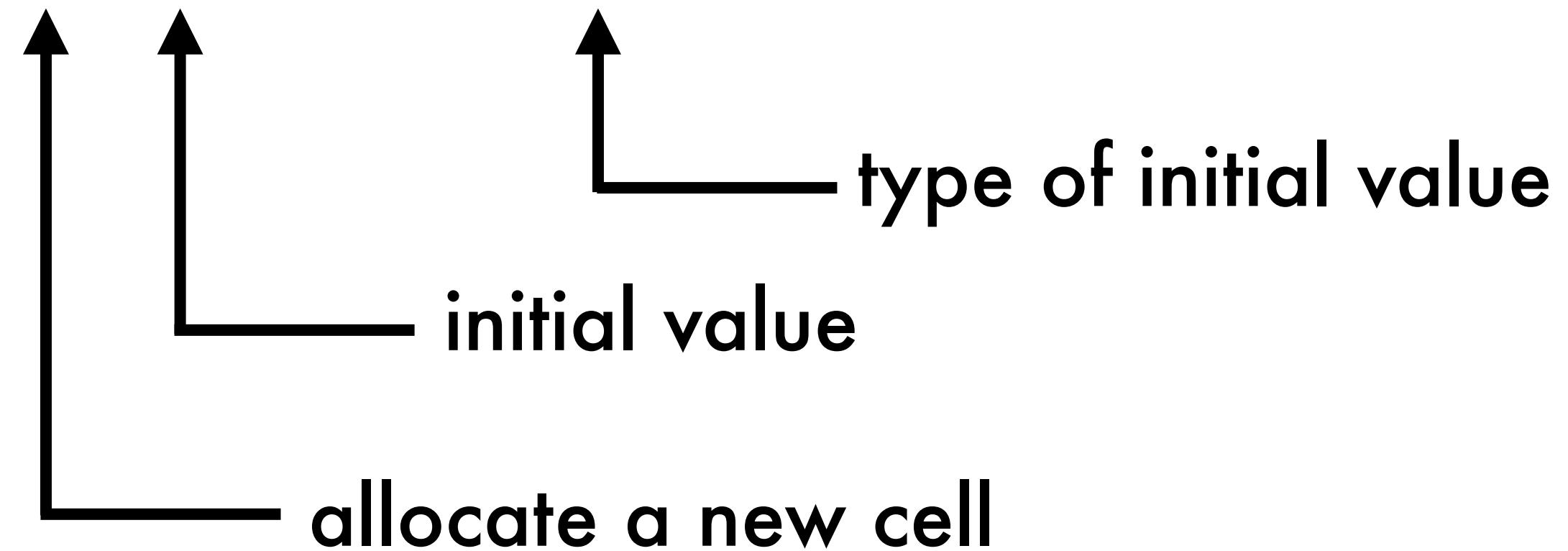
Allocation

Allocating a reference: ref 5 : Ref Nat



Allocation

Allocating a reference: ref 5 : Ref Nat

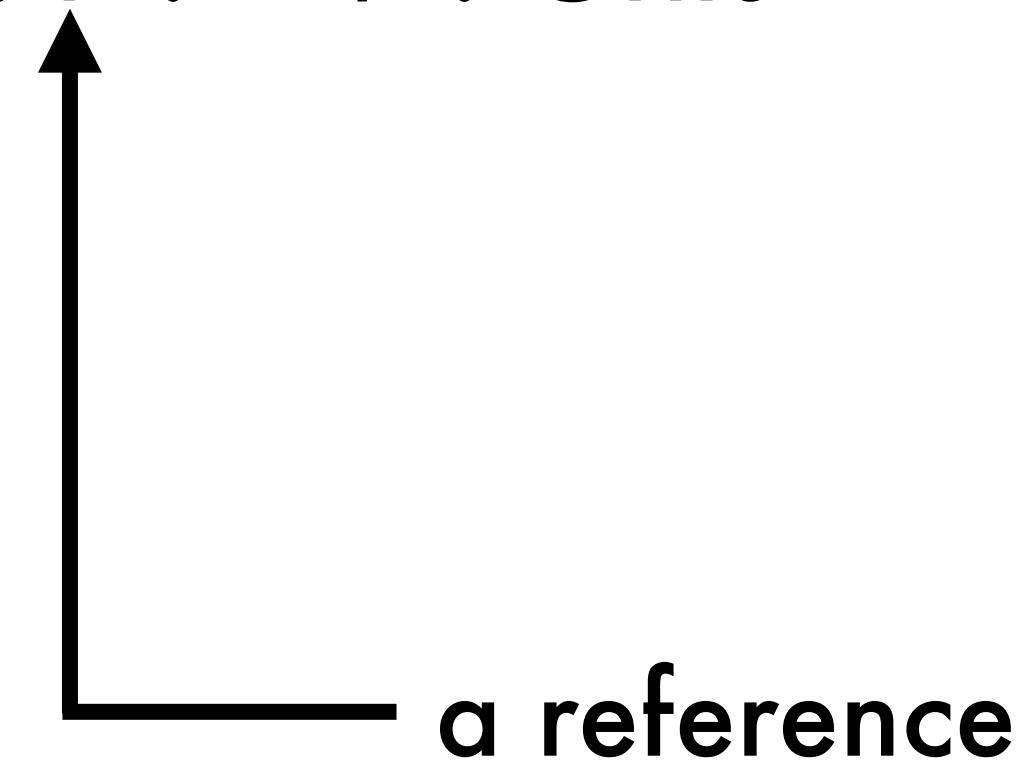


Assignment

Assignment operator: $r := 7 : \text{Unit}$

Assignment

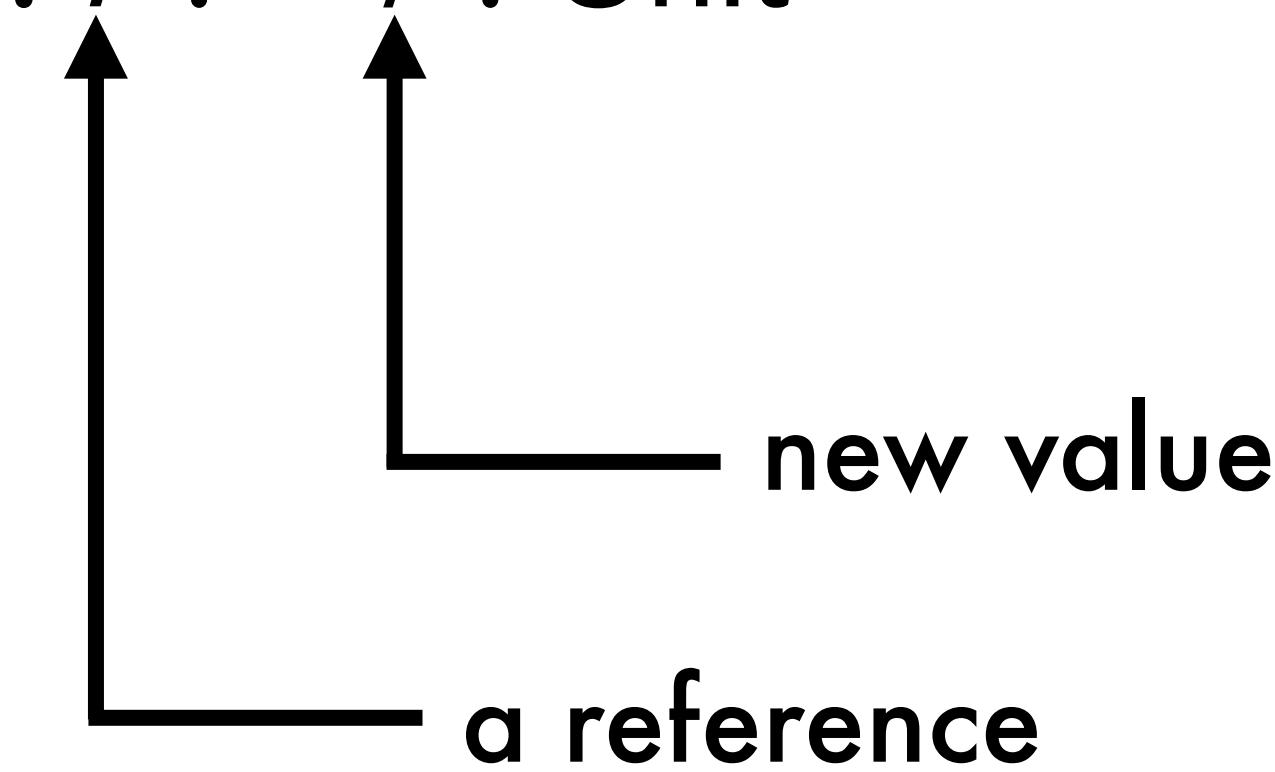
Assignment operator: $r := 7 : \text{Unit}$



a reference

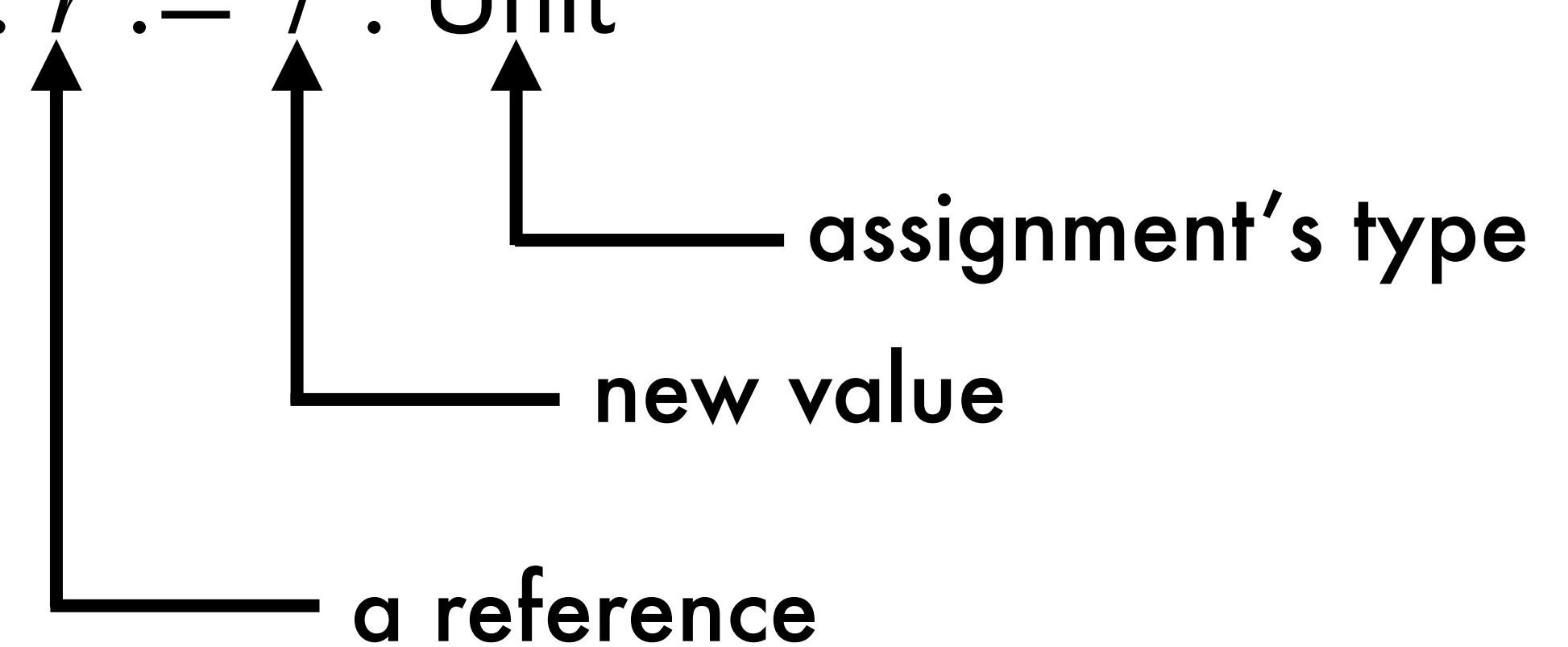
Assignment

Assignment operator: $r := 7 : \text{Unit}$



Assignment

Assignment operator: $r := 7 : \text{Unit}$



Assignment

Assignment operator: $r := 7 : \text{Unit}$

Example:

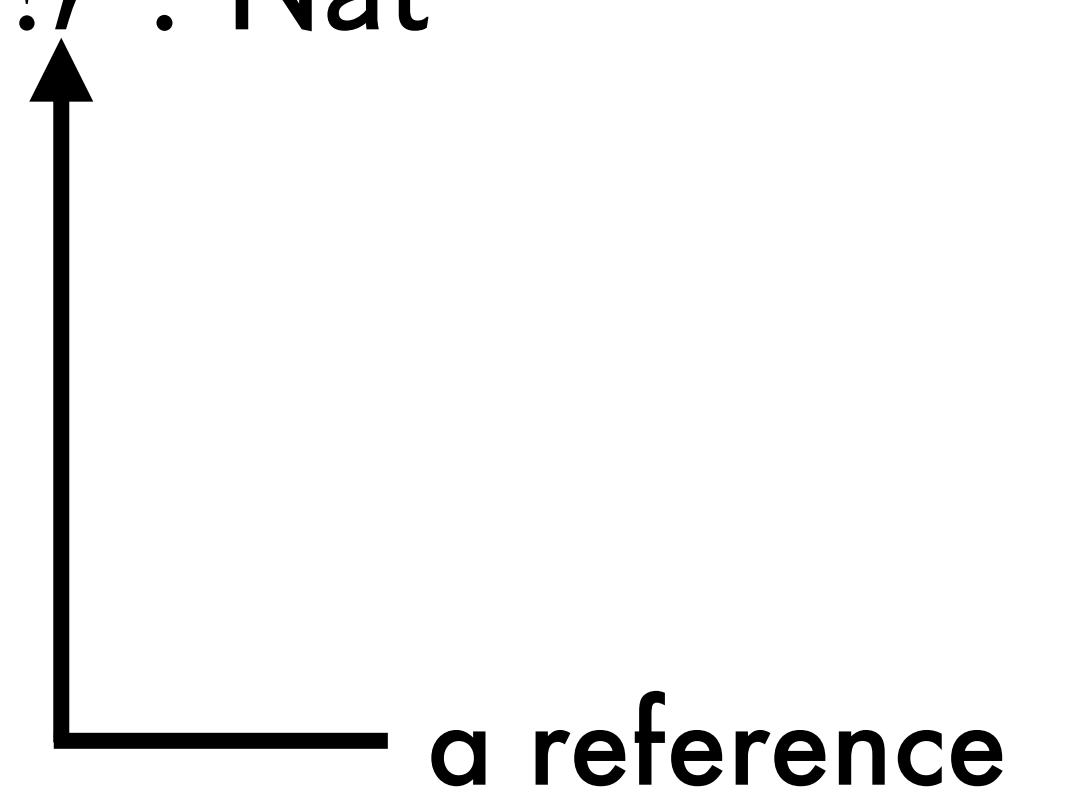
```
let r = ref 5
# r : Ref Nat
r := 7
# unit : Unit
```

Dereferencing

Dereferencing operator: $\mathbf{!r : Nat}$

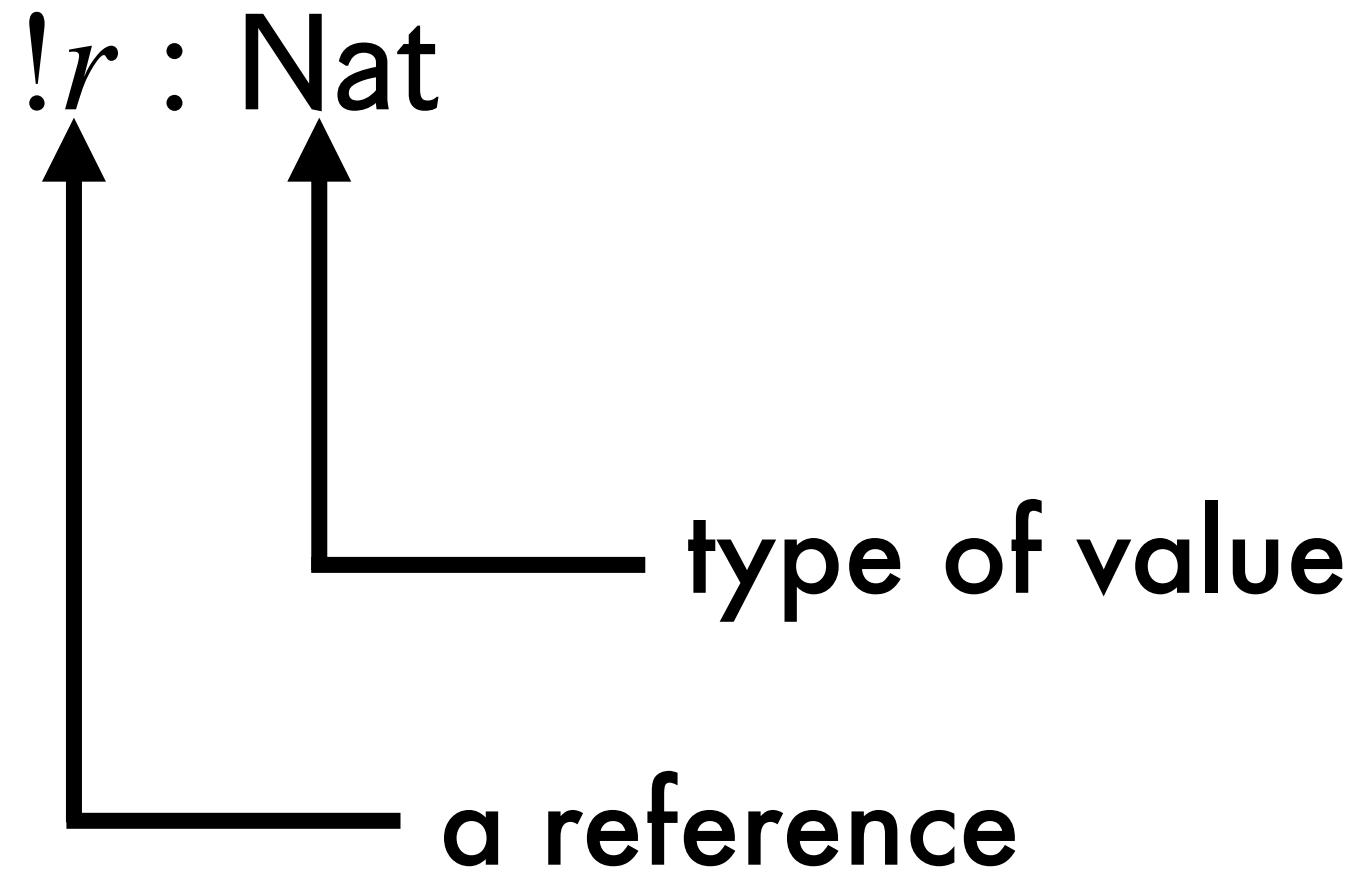
Dereferencing

Dereferencing operator: $\mathbf{!}_r : \mathbf{Nat}$



Dereferencing

Dereferencing operator: $\mathbf{!}r : \mathbf{Nat}$



Dereferencing

Dereferencing operator: $\mathbf{!}r : \mathbf{Nat}$

Example:

```
let r = ref 5
# r : Ref Nat
!r
# 5 : Nat
r := 7
# unit : Unit
!r
# 7 : Nat
```

Sequencing

$$\frac{\Gamma \vdash t_1 : () \quad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1; t_2 : T} \text{ Seq}$$

Sequencing

$$\frac{\Gamma \vdash t_1 : () \quad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1; t_2 : T} \text{ Seq}$$

```
let r = ref 7  
r := succ(!r); !r  
# 8 : Nat
```

Sequencing

$$\frac{\Gamma \vdash t_1 : () \quad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1; t_2 : T} \text{ Seq}$$

```
let r = ref 7
      r := succ(!r);
      r := succ(!r);
      r := succ(!r);
      r := succ(!r);
      !r
# 11 : Nat
```

Stores

- Locations are essentially pointers.
- Stores are sets of mappings from locations to values.
- Store typings are sets of locations with their types.
 - Think of these as "contexts" for stores, but rather than free variables and types we have locations and types.

Stores:

$$\mu ::= \emptyset \quad | \quad \mu, l = v$$

Store Typings:

$$\Sigma ::= \emptyset \quad | \quad \Sigma, l : T$$

Store Substitution:

$$[t/l]\mu = \begin{cases} \emptyset & \text{if } \mu = \emptyset \\ [t/l]\mu, l' = [t/l]v & \text{if } \mu = (\mu, l' = v) \end{cases}$$

Stores

- Stores will be the states during evaluation.
- Store typings will be used to type locations during typing.

Stores:

$$\mu ::= \emptyset$$

$$| \mu, l = v$$

Store Typings:

$$\Sigma ::= \emptyset$$

$$| \Sigma, l : T$$

Base + Mutable References

<u>Terms</u>	<u>Values</u>	<u>Types</u>	<u>Stores</u>	<u>Store Typings</u>
$t ::= \dots$ $\text{ref } t$ $!t$ $t_1 := t_2$ l	$v ::= \dots$ l	$T ::= \dots$ $\text{Ref } T$	$\mu ::= \emptyset$ $\mu, l = v$	$\Sigma ::= \emptyset$ $\Sigma, l : T$

Base + Mutable References

Existing typing rules are updated to the judgment

$$\Gamma \mid \Sigma \vdash t : T$$

existing rules don't change except to carry the Σ along to each premise.

$$\frac{\Sigma(l) = T}{\Gamma \mid \Sigma \vdash l : \text{Ref } T}^{\text{loc}} \quad \frac{\Gamma \mid \Sigma \vdash t_1 : \text{Ref } T \quad \Gamma \mid \Sigma \vdash t_2 : T}{\Gamma \vdash t_1 := t_2 : \text{Unit}}^{\text{assign}}$$

$$\frac{\Gamma \mid \Sigma \vdash t : T}{\Gamma \vdash \text{ref } t : \text{Ref } T}^{\text{ref}}$$

$$\frac{\Gamma \mid \Sigma \vdash t : \text{Ref } T}{\Gamma \vdash !t : T}^{\text{deref}}$$

Base + Mutable References

Existing evaluation rules are updated to the judgment

$$[\mu_1 \mid t_1] \rightsquigarrow [\mu_2 \mid t_2]$$

existing rules don't change except to carry the μ along to each premise; where we replace every $t_1 \rightsquigarrow t_2$ with $[\mu_1 \mid t_1] \rightsquigarrow [\mu_2 \mid t_2]$.

$$\frac{l \notin \text{dom}(\mu)}{[\mu \mid \text{ref } v] \rightsquigarrow [\mu, l = v \mid l]} \text{ ref}_\beta$$

$$\frac{[\mu_1 \mid t_1] \rightsquigarrow [\mu_2 \mid t_2]}{[\mu_1 \mid \text{ref } t_1] \rightsquigarrow [\mu_2 \mid \text{ref } t_2]} \text{ ref}$$

$$\frac{\mu(l) = v}{[\mu \mid !l] \rightsquigarrow [\mu \mid v]} \text{ deref}_\beta$$

$$\frac{[\mu_1 \mid t_1] \rightsquigarrow [\mu_2 \mid t_2]}{[\mu_1 \mid !t_1] \rightsquigarrow [\mu_2 \mid !t_2]} \text{ deref}$$

Base + Mutable References

Existing evaluation rules are updated to the judgment

$$[\mu_1 \mid t_1] \rightsquigarrow [\mu_2 \mid t_2]$$

existing rules don't change except to carry the μ along to each premise; where we replace every $t_1 \rightsquigarrow t_2$ with $[\mu_1 \mid t_1] \rightsquigarrow [\mu_2 \mid t_2]$.

$$\frac{}{[\mu \mid l := v] \rightsquigarrow [[v/l]\mu \mid ()]} \text{assign}\beta$$

$$\frac{[\mu_1 \mid t_1] \rightsquigarrow [\mu_2 \mid t'_1]}{[\mu_1 \mid t_1 := t_2] \rightsquigarrow [\mu_2 \mid t'_1 := t_2]} \text{assign1}$$

$$\frac{[\mu_1 \mid t_2] \rightsquigarrow [\mu_2 \mid t'_2]}{[\mu_1 \mid v_1 := t_2] \rightsquigarrow [\mu_2 \mid v_1 := t'_2]} \text{assign2}$$

Part 7: Subtyping

System: Base + Records

Motivating Example

What type does this program have?

$$(\text{fun } (r : (x : \text{Nat}) \rightarrow \{r.x\}) (x = 0, y = \text{succ } 0)$$

Motivating Example

What type does this program have?

$$\frac{(\text{fun } (r : (x : \text{Nat}) \rightarrow \{r.x\}) (x = 0, y = \text{succ } 0))}{(x : \text{Nat}) \rightarrow \text{Nat}}$$

Motivating Example

What type does this program have?

$$\frac{(\text{fun } (r : (x : \text{Nat}) \rightarrow \{r.x\}) (x = 0, y = \text{succ } 0))}{(x : \text{Nat}) \rightarrow \text{Nat} \qquad (x : \text{Nat}, y : \text{Nat})}$$

Motivating Example

What type does this program have?

$$\frac{(\text{fun } (r : (x : \text{Nat}) \rightarrow \{r.x\}) (x = 0, y = \text{succ } 0))}{(x : \text{Nat}) \rightarrow \text{Nat} \quad (x : \text{Nat}, y : \text{Nat})}$$



it's not typeable!!

Motivating Example

What type does this program have?

$$\frac{(\text{fun } (r : (x : \text{Nat}) \rightarrow \{r.x\}) (x = 0, y = \text{succ } 0))}{(x : \text{Nat}) \rightarrow \text{Nat} \quad (x : \text{Nat}, y : \text{Nat})}$$

😱 it's not typeable!!

$$\frac{\begin{array}{c} \text{identical} \\ \Gamma \vdash t_1 : T_1 \rightarrow T_2 \quad \Gamma \vdash t_2 : T_1 \end{array}}{\Gamma \vdash t_1 t_2 : T_2} \text{App}$$

Motivating Example

What type does this program have?

$$\frac{(\text{fun } (r : (x : \text{Nat}) \rightarrow \{r.x\}) (x = 0, y = \text{succ } 0))}{(x : \text{Nat}) \rightarrow \text{Nat} \quad (x : \text{Nat}, y : \text{Nat})}$$

different

$$\frac{\Gamma \vdash t_1 : T_1 \rightarrow T_2 \quad \Gamma \vdash t_2 : T'_1 \quad T_1 <: T'_1}{\Gamma \vdash t_1 t_2 : T_2} \text{ App}$$

$$\frac{\{l'_1 : T'_1, \dots, l'_j : T'_j\} \subseteq \{l_1 : T_1, \dots, l_i : T_i\}}{(l_1 : T_1, \dots, l_i : T_i) <: (l'_1 : T'_1, \dots, l'_j : T'_j)} \text{ recSub}$$

Motivating Example

With

Subtyping

Increases the set of typeable programs by generalizing the types of the programs that flow into another.

$$\frac{\Gamma \vdash t_1 : T_1 \rightarrow T_2 \quad \Gamma \vdash t_2 : T'_1 \quad T_1 <: T'_1}{\Gamma \vdash t_1 t_2 : T_2} \text{App}$$

$$\frac{\vdash T'_1, \dots, l'_j : T'_j}{\vdash T'_1, \dots, l'_j : T'_j} \text{recSub}$$

Subtyping

principle of safe substitution

S is a subtype of T , written $S <: T$, means any term of type S can safely be used in a context where a term of type T is expected.

Subtyping

subset semantics

S is a subtype of T , written $S <: T$, every value described by S is also described by T .

Subtyping

subsumption

every element t of T_1 is also an element of T_2

$$\frac{\Gamma \vdash t : T_1 \quad T_1 <: T_2}{\Gamma \vdash t : T_2} \text{ sub}$$

Subtyping

$$\frac{\Gamma \vdash (x = 0, y = \text{succ}(0)) : (x : \text{Nat}, y : \text{Nat}) \quad (x : \text{Nat}, y : \text{Nat}) <: (x : \text{Nat})}{\Gamma \vdash (x = 0, y = \text{succ}(0)) : (x : \text{Nat})} \text{ sub}$$

Base + Subtyping

The only syntax that changes is the addition of a Top type.

Terms

$t ::= \dots$

Values

$v ::= \dots$

Types

$T ::= \dots$

| Top

Base + Subtyping

Subtyping Rules

$$\frac{}{T <: T} \text{Refl}$$

$$\frac{T_1 <: T_2 \quad T_2 <: T_3}{T_1 <: T_3} \text{Trans}$$

$$\frac{}{T <: \text{Top}} \text{Top}$$

$$\frac{T'_1 <: T_1 \quad T_2 <: T'_2}{T_1 \rightarrow T_2 <: T'_1 \rightarrow T'_2} \text{Arrow}$$

Base + Records + Subtyping

Subtyping Rules

$$\frac{}{(l_i : T_i)^{i \in \{1 \dots n+k\}} <: (l_i : T_i)^{i \in \{1 \dots n\}}} \text{RecWidth}$$

$$\frac{\forall i \in \{1 \dots n\} . T_i <: T'_i}{(l_i : T_i)^{i \in \{1 \dots n\}} <: (l_i : T'_i)^{i \in \{1 \dots n\}}} \text{RecDepth}$$

$$\frac{(l_i : T_i)^{i \in \{1 \dots n\}} \text{ is a permutation of } (l_j : T'_j)^{j \in \{1 \dots n\}}}{(l_i : T_i)^{i \in \{1 \dots n\}} <: (l_j : T'_j)^{j \in \{1 \dots n\}}} \text{RecDepth}$$

Base + Subtyping

Typing rules are all the same, except the addition of the subsumption rule.

$$\frac{\Gamma \vdash t : T_1 \quad T_1 <: T_2}{\Gamma \vdash t : T_2} \text{Sub}$$