Binding and Substitution

$e ::= \ldots \mid x \mid \text{let } x = e \text{ in } e$

everything from before — variables — let binding

$\text{let } x = e_1 \text{ in } e_2$ Say: "x is bound in e_2 (not e_1)"

$\text{let } x = \bar{1} \text{ in } x + \bar{2}$ A use of a var. refers to the nearest enclosing binding

Intuition (rules later):

$\text{let } x = \bar{1} \text{ in } x + \bar{2}$

$\mapsto \bar{1} + \bar{2}$

$\mapsto 3$

$\text{let } x = \bar{1} \text{ in } (\text{let } x = \bar{2} \text{ in } x + \bar{1}) + x$

$\mapsto (\text{let } x = \bar{2} \text{ in } x + \bar{1}) + \bar{1}$

$\mapsto (\bar{2} + \bar{1}) + \bar{1}$ This x is not updated

$\mapsto^* 4$ Substitution, not updates

Free vs. bound

If a var. isn't bound, it's free.

$FV(e)$ "free variables of e"

$FV(x) = \{x\}$

$FV(\bar{n}) = FV("s") = \emptyset$

$FV(e_1 + e_2) = FV(e_1 \wedge e_2) = FV(e_1) \cup FV(e_2)$

$FV(\text{let } x = e_1 \text{ in } e_2) = FV(e_1) \cup (FV(e_2) \setminus \{x\})$

let $x = y$ in $x + \bar{2}$

y is free (in this expr.; might be bound if this is part of a bigger expr.)

α-conversion: Can always (consistently) rename bound vars.

α-equivalent: expressions are the same up to α-conversion ($\equiv_\alpha$)

let $x = \bar{1}$ in $x + \bar{2} \equiv_\alpha$ let $y = \bar{1}$ in $y + \bar{2}$

let $x = y$ in $x + 2 \not\equiv_\alpha$ let $x = z$ in $x + \bar{2}$

Ex. from before:

let $x = \bar{1}$ in (let $x = \bar{2}$ in $x + \bar{1}$) + x

$\equiv_\alpha$ let $x = \bar{1}$ in (let $y = \bar{2}$ in $y + \bar{1}$) + x

Now it's clear this x isn't updated

Substitution

$[e_1/x]\, e_2$ "substitute e_1 for all free instances of x in e_2"

$[e/x]\, x \qquad = e$

$[e/x]\, y \qquad = y \qquad y \neq x$

$[e/x]\, \bar{n} \qquad = \bar{n}$

$[e/x]\, \text{"s"} \qquad = \text{"s"}$

$[e/x]\,(e_1 + e_2) \qquad = [e/x]e_1 + [e/x]e_2$

" ^ $\qquad$ = " ^ "

$[e/x]\,(\text{let } x = e_1 \text{ in } e_2) \qquad = \text{let } x = [e/x]e_1 \text{ in } e_2$

$[e/x]\,(\text{let } y = e_1 \text{ in } e_2) \qquad = \text{let } y = [e/x]e_1 \text{ in } [e/x]e_2 \qquad y \neq x$ and $y \notin FV(e)$

why?

$[e/x]\,(\text{let } x = \bar{1} \text{ in } x + \bar{2}) \neq \text{let } x = 1 \text{ in } e + 2$

$\equiv_\alpha [e/x]\,(\text{let } y = 1 \text{ in } y + \bar{2})$

$[x+\bar{2}/y]\,(\text{let } x = \bar{1} \text{ in } y + \bar{2}) \neq \text{let } x = \bar{1} \text{ in } x + \bar{2} + \bar{2}$

this x is supposed to be free

now is bound ("captured")

What if $y \notin FV(e)$? α-convert

$[x+2/y]\,(\text{let } x = \bar{1} \text{ in } y + \bar{2})$

$\equiv_\alpha [x+2/y]\,(\text{let } z = \bar{1} \text{ in } y + \bar{2})$

$= \qquad \text{let } z = \bar{1} \text{ in } x + \bar{2} + \bar{2}$ ✓

Dynamics

2 versions:

"call-by-value" (strict)

$$\frac{e_1 \mapsto e_1'}{\texttt{let } x = e_1 \texttt{ in } e_2 \mapsto \texttt{let } x = e_1' \texttt{ in } e_2}\ (\text{StepSearchLet}) \qquad \frac{v \text{ val}}{\texttt{let } x = v \texttt{ in } e_2 \mapsto [v/x]e_2}\ (\text{StepLet})$$

"Call-by-name"

$$\frac{}{\texttt{let } x = e_1 \texttt{ in } e_2 \mapsto [e_1/x]e_2}\ (\text{Step Let (BN)})$$

Statics

$\Gamma \vdash e : \tau$ "under context Γ, e has type τ"

context: maps vars to types

Write a context like this: $x_1 : \tau_1, x_2 : \tau_2, \ldots, x_n : \tau_n$

$\Gamma, x : \tau$ - extend Γ with $x : \tau$ (implies $x \notin dom(\Gamma)$)

Order doesn't matter, so if $x : \tau \in \Gamma$, we can write Γ as $\Gamma', x : \tau$
(for now)

Empty context: $\emptyset$ or $\bullet$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}\ (\text{Type Var}) \quad \left(\text{sometimes written: } \frac{}{\Gamma, x : \tau \vdash x : \tau}\right) \quad \frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash \texttt{let } x = e_1 \texttt{ in } e_2 : \tau_2}\ (\text{Type Let})$$

$$\frac{}{\Gamma \vdash \bar{n} : \texttt{int}}\ (\text{TypeNum}) \qquad \frac{}{\Gamma \vdash \texttt{"s"} : \texttt{string}}\ (\text{TypeString}) \qquad \frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 + e_2 : \texttt{int}}\ (\text{TypeAdd})$$

$$\frac{\Gamma \vdash e_1 : \texttt{string} \quad \Gamma \vdash e_2 : \texttt{string}}{\Gamma \vdash e_1 \hat{} e_2 : \texttt{string}}\ (\text{Type Cat}) \qquad \frac{\Gamma \vdash e : \texttt{string}}{\Gamma \vdash |e| : \texttt{int}}\ (\text{Type Len})$$

Structural Properties

Weakening: If $\Gamma \vdash e : \tau$ and $x \notin dom(\Gamma)$, then $\Gamma, x:\tau' \vdash e:\tau$.

Proof: By induction on the derivation of $\Gamma \vdash e:\tau$.

TypeVar. Then $e = y$ and $\Gamma(y) = \tau$. ~~Pa~~ $(\Gamma, x:\tau')(y) = \tau$, since $y \neq x$.
Apply TypeVar.

TypeNum. Then $e = \bar{n}$. Apply TypeNum.
TypeString. Similar

TypeAdd. Then $e = e_1 + e_2$, and $\tau = int$, and $\Gamma \vdash e_1 : int$ and $\Gamma \vdash e_2 : int$.
By induction, $\Gamma, x:\tau' \vdash e_1 : int$ and $\Gamma, x:\tau' \vdash e_2 : int$. Apply TypeAdd.
TypeCat, TypeLen. Similar.

Substitution: If $\Gamma, x:\tau' \vdash e:\tau$ and $\Gamma \vdash e':\tau'$ then $\Gamma \vdash [e'/x]e : \tau$.
Pf: By induction on the derivation of $\Gamma, x:\tau' \vdash e:\tau$.

TypeAdd. Then $e = e_1 + e_2$ and $[e'/x]e = [e'/x]e_1 + [e'/x]e_2$.
By IH, $\Gamma \vdash [e'/x]e_1 : int$ and $\Gamma \vdash [e'/x]e_2 : int$. Apply TypeAdd.

TypeVar. $\dfrac{\Gamma(y) = \tau}{\Gamma \vdash y : \tau}$ $e = y$ and $\Gamma = \Gamma', x:\tau'$.

Case 1: $x = y$. Then $[e'/x]e = e'$ and $\tau = \tau'$. By assumption, $\Gamma \vdash e' : \tau'$.
Case 2: $x \neq y$. Then $[e'/x]y = y$. ~~By TypeVar, Γ~~ Since $x \neq y$, $\Gamma'(y) = \tau$.
Apply TypeVar.

TypeLet. $\dfrac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, y:\tau_1 \vdash e_2 : \tau}{\Gamma \vdash \text{let } y = e_1 \text{ in } e_2 : \tau}$ Then $e = \text{let } y = e_1 \text{ in } e_2$
and $\Gamma, x:\tau' \vdash e_1 : \tau_1$ and $\Gamma, x:\tau', y:\tau_1 \vdash e_2 : \tau$.

~~Case 1: $x = y$. Then $[e'/x]e = \text{let } x = [e'/x]e_1 \text{ in } e_2$.~~
~~By induction, $\Gamma \vdash [e'/x]e_1 : \tau_1$~~

By α-conversion, assume $x \neq y$ and $y \notin FV(e')$.

Then $[e'/x]e = \text{let } y = [e'/x]e_1 \text{ in } [e'/x]e_2$.

We have $\Gamma, x:\tau' \vdash e_1 : \tau_1$ and $\Gamma, x:\tau', y:\tau_1 \vdash e_2 \ \tau_2$.

By IH, $\Gamma \vdash [e'/x]e_1 : \tau_1$. By weakening, $\Gamma, y:\tau_1 \vdash e' : \tau'$.

By IH, $\Gamma, y:\tau_1 \vdash [e'/x]e_2 : \tau_2$ (also had to swap the order of x and y)

By TypeLet, $\Gamma \vdash [e'/x]e : \tau_2$.

<u>Preservation</u>: If $\bullet \vdash e : \tau$ and $e \mapsto e'$ then $\bullet \vdash e' \ \tau$. (Note: only evaluate "closed" programs, i.e. no free vars.)

Most cases unchanged.

Case StepLet (or StepLetCBN). Then $e = \text{let } x = e_1 \text{ in } e_2$
$e' = [e_1/x]e_2$.

By inversion, $\bullet \vdash e_1 : \tau_1$ and $\bullet, x \ \tau_1 \vdash e_2 : \tau$.

By substitution, $\bullet \vdash [e_1/x]e_2 : \tau$.

<u>Progress</u>: If $\bullet \vdash e : \tau$ then e val or $e \mapsto e'$

Case TypeVar. Then $e = x$ and $\bullet(x) = \tau$. But this is a contradiction!

Case TypeLet. Then $e = \text{let } x = e_1 \text{ in } e_2$ and $\bullet \vdash e_1 : \tau_1$ and $x \ \tau_1 \vdash e_2 : \tau$.

By IH, e_1 val or $e_1 \mapsto e_1'$.

- e_1 val. Apply StepLet.
- $e_1 \mapsto e_1'$. Apply StepSearchLet

(if using CBN, just apply StepLet CBN no matter what!)