Australian National University

COMP3610/6361
Principles of Programming Languages

Peter Höfner

Aug 1, 2023

Section 5

Functions

Functions, Methods, Procedures, . . .

- so far IMP was really minimalistic
- the most important ‘add-on’ are functions
- this requires variables and other concepts

Examples

```
add_one :: Int -> Int
add_one n = n + 1
```

```
public int add_one (int x) {
  return (x+1);
}
```

```
<script type="text/vbscript">
function addone(x)
    addone = x+1
end function
</script>
```

Introductory Examples: C♯

In C♯, what is the output of the following?

```
delegate int IntThunk();
class C {
  public static void Main() {
    IntThunk[] funcs = new IntThunk[11];
    for (int i = 0; i <= 10; i++)
    {
        funcs[i] = delegate() { return i; } ;
    }
    foreach (IntThunk f in funcs)
    {
        System.Console.WriteLine(f());
    }
  }
}
```

In my opinion, the design was wrong.

Functions – Examples

We want include the following expressions:

$$(\textbf{fn}\ x : \mathsf{int} \Rightarrow x + 1)$$

$$(\textbf{fn}\ x : \mathsf{int} \Rightarrow x + 1)\ 7$$

$$(\textbf{fn}\ y : \mathsf{int} \Rightarrow (\textbf{fn}\ x : \mathsf{int} \Rightarrow x + y))$$

$$(\textbf{fn}\ y : \mathsf{int} \Rightarrow (\textbf{fn}\ x : \mathsf{int} \Rightarrow x + y))\ 1$$

$$(\textbf{fn}\ x : \mathsf{int} \to \mathsf{int} \Rightarrow (\textbf{fn}\ y : \mathsf{int} \Rightarrow x\ (x\ y)))$$

$$(\textbf{fn}\ x : \mathsf{int} \to \mathsf{int} \Rightarrow (\textbf{fn}\ y : \mathsf{int} \Rightarrow x\ (x\ y)))\ (\textbf{fn}\ x : \mathsf{int} \Rightarrow x + 1)$$

$$\big((\textbf{fn}\ x : \mathsf{int} \to \mathsf{int} \Rightarrow (\textbf{fn}\ y : \mathsf{int} \Rightarrow x\ (x\ y)))\ (\textbf{fn}\ z : \mathsf{int} \Rightarrow z + 1)\big)\ 7$$

Functions – Syntax

We extend our syntax:

Variables $x \in \mathbb{X}$ for a set $\mathbb{X} = \{x, y, z, \dots\}$ (countable)

Expressions

$$E ::= \dots \mid (\textbf{fn}\ x : T \Rightarrow E) \mid E\ E \mid x$$

Types

$$T ::= \text{int} \mid \text{bool} \mid \text{unit} \mid T \to T$$
$$T_{loc} ::= \text{intref}$$

Variable Shadowing

$$(\textbf{fn}\ x : \mathsf{int} \Rightarrow (\textbf{fn}\ x : \mathsf{int} \Rightarrow x + 1))$$

Alpha conversion

In expressions $(\textbf{fn}\ x : T \Rightarrow E)$, variable x is a binder

- inside E, any x (not being a binder themselves and not inside another $(\textbf{fn}\ x : T' \Rightarrow \ldots)$) mean the same
- it is the formal parameter of this function
- outside $(\textbf{fn}\ x : T \Rightarrow E)$, it does not matter which variable we use – in fact, we should not be able to tell
  For example, $(\textbf{fn}\ x : \text{int} \Rightarrow x + 2)$ should be the same as $(\textbf{fn}\ y : \text{int} \Rightarrow y + 2)$

Binders are known from many areas of mathematics/logics.

Alpha conversion: free and bound variables

An occurrence x in an expression E is *free* if it is not inside any ($\textbf{fn}\ x : T \Rightarrow \ldots$).
For example:

$$17$$

$$x + y$$

$$(\textbf{fn}\ x : \text{int} \Rightarrow x + 2)$$

$$(\textbf{fn}\ x : \text{int} \Rightarrow x + z)$$

$$\textbf{if}\ y\ \textbf{then}\ 2 + x\ \textbf{else}\ ((\textbf{fn}\ x : \text{int} \Rightarrow x + 2)\ z)$$

Alpha Conversion – Binding Examples

$$(\textbf{fn}\ x : \mathsf{int} \Rightarrow x + 2)$$

$$(\textbf{fn}\ x : \mathsf{int} \Rightarrow x + z)$$

$$(\textbf{fn}\ y : \mathsf{int} \Rightarrow y + z)$$

$$(\textbf{fn}\ z : \mathsf{int} \Rightarrow z + z)$$

$$(\textbf{fn}\ x : \mathsf{int} \Rightarrow (\textbf{fn}\ x : \mathsf{int} \Rightarrow x + 2))$$

Alpha Conversion – Convention

- we want to allow to replace binder x (and all occurrences of x bound by that x) by another binder y
- *if* it does not change the binding graph

For example

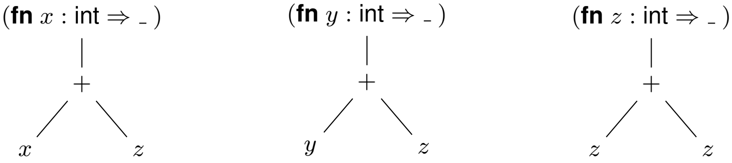
$$(\textbf{fn}\ x : \mathsf{int} \Rightarrow x + z) \ = \ (\textbf{fn}\ y : \mathsf{int} \Rightarrow y + z) \ \neq \ (\textbf{fn}\ z : \mathsf{int} \Rightarrow z + z)$$

- called 'working up to alpha conversion'
- extend abstract syntax trees by pointers

Syntax Trees up to Alpha Conversion

$$(\textbf{fn}\ x : \mathsf{int} \Rightarrow x + z) \ = \ (\textbf{fn}\ y : \mathsf{int} \Rightarrow y + z) \ \neq \ (\textbf{fn}\ z : \mathsf{int} \Rightarrow z + z)$$

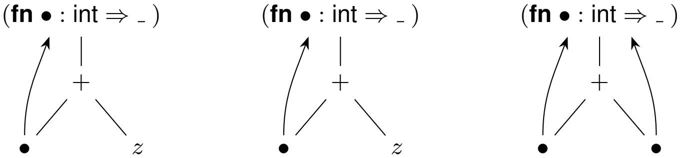
Standard abstract syntax trees



Syntax Trees up to Alpha Conversion II

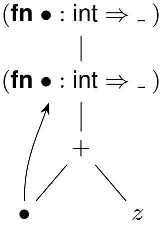
$$(\textbf{fn}\ x : \mathsf{int} \Rightarrow x + z) \ = \ (\textbf{fn}\ y : \mathsf{int} \Rightarrow y + z) \ \neq \ (\textbf{fn}\ z : \mathsf{int} \Rightarrow z + z)$$

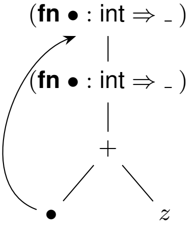
Add pointers



Syntax Trees up to Alpha Conversion III

$$
\begin{aligned}
& (\textbf{fn}\ x : \mathsf{int} \Rightarrow (\textbf{fn}\ x : \mathsf{int} \Rightarrow x + 2)) \\
= & (\textbf{fn}\ y : \mathsf{int} \Rightarrow (\textbf{fn}\ z : \mathsf{int} \Rightarrow z + 2)) \quad \neq \quad (\textbf{fn}\ z : \mathsf{int} \Rightarrow (\textbf{fn}\ y : \mathsf{int} \Rightarrow z + 2))
\end{aligned}
$$

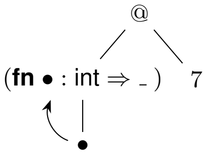


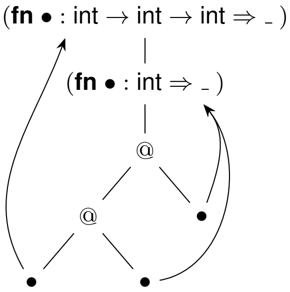


Syntax Trees up to Alpha Conversion IV

Application and function type

$(\textbf{fn}\ x : \mathsf{int} \Rightarrow x)\ 7$ $\qquad$ $(\textbf{fn}\ z : \mathsf{int} \to \mathsf{int} \to \mathsf{int} \Rightarrow (\textbf{fn}\ y : \mathsf{int} \Rightarrow z\ y\ y))$

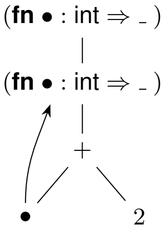


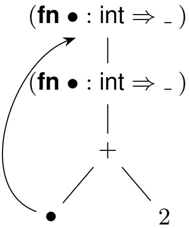


De Bruijn indices

- these pointers are known as *De Bruijn indices*
- each occurrence of a bound variable is represented by the number of **fn**-nodes you have to pass

$$(\textbf{fn}\ \bullet : \text{int} \Rightarrow (\textbf{fn}\ \bullet : \text{int} \Rightarrow v_0 + 2)) \quad \neq \quad (\textbf{fn}\ \bullet : \text{int} \Rightarrow (\textbf{fn}\ \bullet : \text{int} \Rightarrow v_1 + 2))$$





Free Variables

- *free variables* of an expression E are the set of variables for which there is an occurrence of x free in E

$$\begin{aligned}\mathsf{fv}(x) &= \{x\} \\ \mathsf{fv}(E_1\ op\ E_2) &= \mathsf{fv}(E_1) \cup \mathsf{fv}(E_2) \\ \mathsf{fv}((\textbf{fn}\ x : T \Rightarrow E)) &= \mathsf{fv}(E) - \{x\}\end{aligned}$$

- an expression E is closed if $\mathsf{fv}(E) = \emptyset$
- For a set $\mathbb{E}$ of expressions $\mathsf{fv}(\mathbb{E}) = \bigcup_{E \in \mathbb{E}} \mathsf{fv}(E)$
- these definitions are alpha-invariant
  (all forthcoming definitions should be)

Substitution – Examples

- semantics of functions will involve substitution (replacement)
- $\{E/x\}\, E'$ denotes the expression E' where all *free* occurrences of x are substituted by E

Examples

$$\{3/x\}\,(x \geq x) = (3 \geq 3)$$

$$\{3/x\}\,((\textbf{fn}\ x : \text{int} \Rightarrow x + y)\ x) = (\textbf{fn}\ x : \text{int} \Rightarrow x + y)\ 3$$

$$\{y + 2/x\}\,(\textbf{fn}\ y : \text{int} \Rightarrow x + y) = (\textbf{fn}\ z : \text{int} \Rightarrow (y + 2) + z)$$

Substitution

Definition

$$\{E/z\}\, x \stackrel{\text{def}}{=} \begin{cases} E & \text{if } x = z \\ x & \text{otherwise} \end{cases}$$

$$\{E/z\}\, (\textbf{fn}\ x : T \Rightarrow E_1) \stackrel{\text{def}}{=} (\textbf{fn}\ x : T \Rightarrow (\{E/z\}\, E_1)) \quad \text{if } x \neq z \text{ and } x \notin \text{fv}(E) (^*)$$

$$\{E/z\}\, (E_1\ E_2) \stackrel{\text{def}}{=} (\{E/z\}\, E_1) \quad (\{E/z\}\, E_2)$$

$$\ldots$$

if $(^*)$ is false, apply alpha conversion to generate a variant of $(\textbf{fn}\ x : T \Rightarrow E_1)$ to make $(^*)$ true

Australian National University

Substitution – Example

Substitution – Example Again

$$
\begin{aligned}
& \{y + 2/x\}\,(\textbf{fn}\ y : \mathsf{int} \Rightarrow x + y) \\
={} & \{y + 2/x\}\,(\textbf{fn}\ z : \mathsf{int} \Rightarrow x + z) \\
={} & (\textbf{fn}\ z : \mathsf{int} \Rightarrow \{y + 2/x\}\,(x + z)) \\
={} & (\textbf{fn}\ z : \mathsf{int} \Rightarrow \{y + 2/x\}\,x + \{y + 2/x\}\,z) \\
={} & (\textbf{fn}\ z : \mathsf{int} \Rightarrow (y + 2) + z)
\end{aligned}
$$

Simultaneous Substitution

- a *substitution* σ is a *finite* partial function from variables to expressions
- notation: $\{E_1/x_1, \ldots, E_k/x_k\}$ instead of $\{x_1 \mapsto E_1, \ldots, x_k \mapsto E_k\}$
- the formal definition is straight forward

Australian National University

Definition Substitution [for completeness]

Let σ be $\{E_1/x_1, \ldots, E_k/x_k\}$.
Moreover, dom$(\sigma) = \{x_1, \ldots, x_k\}$ and ran$(\sigma) = \{E_1, \ldots, E_k\}$.

$$
\begin{array}{lcl}
\sigma\ x & = & \begin{cases} E_i & \text{if } x = x_i \text{ (and } x_i \in \text{dom}(\sigma)\text{)} \\ x & \text{otherwise} \end{cases} \\
\sigma\ (\textbf{fn}\ x : T \Rightarrow E) & = & (\textbf{fn}\ x : T \Rightarrow (\sigma\ E)) \qquad \text{if } x \notin \text{dom}(\sigma) \text{ and } x \notin \text{fv}(\text{ran}(\sigma))\ (^*) \\
\sigma\ (E_1\ E_2) & = & (\sigma\ E_1)\ (\sigma\ E_2) \\
\sigma\ n & = & n \\
\sigma\ (E_1\ op\ E_2) & = & (\sigma\ E_1)\ op\ (\sigma\ E_2) \\
\sigma\ (\textbf{if}\ E_1\ \textbf{then}\ E_2\ \textbf{else}\ E_3) & = & \textbf{if}\ (\sigma\ E_1)\ \textbf{then}\ (\sigma\ E_2)\ \textbf{else}\ (\sigma\ E_3) \\
\sigma\ b & = & b \\
\sigma\ \textbf{skip} & = & \textbf{skip} \\
\sigma\ (l := E) & = & l := (\sigma\ E) \\
\sigma\ (\ !l) & = & \ !l \\
\sigma\ (E_1\ ;\ E_2) & = & (\sigma\ E_1)\ ;\ (\sigma\ E_2) \\
\sigma\ (\textbf{while}\ E_1\ \textbf{do}\ E_2) & = & \textbf{while}\ (\sigma\ E_1)\ \textbf{do}\ (\sigma\ E_2)
\end{array}
$$

Function Behaviour

- we are now ready to define the semantics of functions
- there are some choices to be made
  - call-by-value
  - call-by-name
  - call-by-need

Australian National University

Function Behaviour

Consider the expression

$$E = (\textbf{fn}\ x : \mathsf{unit} \Rightarrow (l := 1)\ ;\ x)\ (l := 2)$$

What is the transition relation

$$\langle E\,,\ \{l \mapsto 0\}\rangle \longrightarrow^* \langle \textbf{skip}\,,\ \{l \mapsto \textbf{???}\}\rangle$$

Choice 1: Call-by-Value

Idea: reduce left-hand-side of application to an **fn**-term;
then reduce argument to a value;
then replace all occurrences of the formal parameter in the **fn**-term by that value.

$$E = (\textbf{fn}\ x : \mathsf{unit} \Rightarrow (l := 1)\ ;\ x)\ (l := 2)$$

$$\begin{array}{ll} & \langle E\,,\ \{l \mapsto 0\}\rangle \\ \longrightarrow & \langle (\textbf{fn}\ x : \mathsf{unit} \Rightarrow (l := 1)\ ;\ x)\ \textbf{skip}\,,\ \{l \mapsto 2\}\rangle \\ \longrightarrow & \langle (l := 1)\ ;\ \textbf{skip}\,,\ \{l \mapsto 2\}\rangle \\ \longrightarrow & \langle \textbf{skip}\ ;\ \textbf{skip}\,,\ \{l \mapsto 1\}\rangle \\ \longrightarrow & \langle \textbf{skip}\,,\ \{l \mapsto 1\}\rangle \end{array}$$

Call-by-Value – Semantics

Values

$v ::= b \mid n \mid \textbf{skip} \mid (\textbf{fn}\ x : T \Rightarrow E)$

SOS rules

all sos rules we used so far, plus the following

(app1) $$\frac{\langle E_1\ ,\ s\rangle \longrightarrow \langle E_1'\ ,\ s'\rangle}{\langle E_1\ E_2\ ,\ s\rangle \longrightarrow \langle E_1'\ E_2\ ,\ s'\rangle}$$

(app2) $$\frac{\langle E_2\ ,\ s\rangle \longrightarrow \langle E_2'\ ,\ s'\rangle}{\langle v\ E_2\ ,\ s\rangle \longrightarrow \langle v\ E_2'\ ,\ s'\rangle}$$

(fn) $$\langle (\textbf{fn}\ x : T \Rightarrow E)\ v\ ,\ s\rangle \longrightarrow \langle \{v/x\}\ E\ ,\ s\rangle$$

Call-by-Value – Example I

$$
\begin{array}{rl}
 & \langle(\textbf{fn}\ x : \mathsf{int} \Rightarrow (\textbf{fn}\ y : \mathsf{int} \Rightarrow x + y))\ (3+4)\ 5\ ,\ s\rangle \\
= & \langle\big((\textbf{fn}\ x : \mathsf{int} \Rightarrow (\textbf{fn}\ y : \mathsf{int} \Rightarrow x + y))\ (3+4)\big)\ 5\ ,\ s\rangle \\
\longrightarrow & \langle\big((\textbf{fn}\ x : \mathsf{int} \Rightarrow (\textbf{fn}\ y : \mathsf{int} \Rightarrow x + y))\ 7\big)\ 5\ ,\ s\rangle \\
\longrightarrow & \langle\big(\{7/x\}\ (\textbf{fn}\ y : \mathsf{int} \Rightarrow x + y)\big)\ 5\ ,\ s\rangle \\
= & \langle(\textbf{fn}\ y : \mathsf{int} \Rightarrow 7 + y)\ 5\ ,\ s\rangle \\
\longrightarrow & \langle\{5/y\}\,7 + y\ ,\ s\rangle \\
= & \langle 7 + 5\ ,\ s\rangle \\
\longrightarrow & \langle 12\ ,\ s\rangle
\end{array}
$$

Call-by-Value – Example II

$$(\textbf{fn}\ f : \mathsf{int} \to \mathsf{int} \Rightarrow f\ 3)\ (\textbf{fn}\ x : \mathsf{int} \Rightarrow (1 + 2) + x) \quad \longrightarrow^{*} \quad ???$$

Choice 2: Call-by-Name

Idea: reduce left-hand-side of application to an **fn**-term;
then replace all occurrences of the formal parameter in the **fn**-term by that argument.

$$E = (\textbf{fn}\ x : \mathsf{unit} \Rightarrow (l := 1)\ ;\ x)\ (l := 2)$$

$$
\begin{aligned}
& \langle E\,,\ \{l \mapsto 0\}\rangle \\
\longrightarrow\ & \langle (l := 1)\ ;\ (l := 2)\,,\ \{l \mapsto 0\}\rangle \\
\longrightarrow\ & \langle \textbf{skip}\ ;\ (l := 2)\,,\ \{l \mapsto 1\}\rangle \\
\longrightarrow\ & \langle l := 2\,,\ \{l \mapsto 1\}\rangle \\
\longrightarrow\ & \langle \textbf{skip}\,,\ \{l \mapsto 2\}\rangle
\end{aligned}
$$

Call-by-Name – Semantics

SOS rules

(CBN-app) $$\frac{\langle E_1\ ,\ s\rangle \longrightarrow \langle E_1'\ ,\ s'\rangle}{\langle E_1\ E_2\ ,\ s\rangle \longrightarrow \langle E_1'\ E_2\ ,\ s'\rangle}$$

(CBN-fn) $$\langle(\textbf{fn}\ x : T \Rightarrow E_1)\ E_2\ ,\ s\rangle \longrightarrow \langle\{E_2/x\}\,E_1\ ,\ s\rangle$$

No evaluation unless needed

$$\begin{aligned} & \langle(\textbf{fn}\ x : \text{unit} \Rightarrow \textbf{skip})\ (l := 2)\ ,\ \{l \mapsto 0\}\rangle \\ \longrightarrow\ & \langle\{l := 2/x\}\,\textbf{skip}\ ,\ \{l \mapsto 0\}\rangle \\ =\ & \langle\textbf{skip}\ ,\ \{l \mapsto 0\}\rangle \end{aligned}$$

but if it is needed, repeated evaluation possible.

Choice 3: Full Beta

Idea: allow reductions on left-hand-side and right-hand-side;
any time if left-hand-side is an **fn**-term;
replace all occurrences of the formal parameter in the **fn**-term by that argument; allow reductions inside functions

$$\langle(\textbf{fn}\ x : \text{int} \Rightarrow 2 + 2)\ ,\ s\rangle \longrightarrow \langle(\textbf{fn}\ x : \text{int} \Rightarrow 4)\ ,\ s\rangle$$

Full Beta – Semantics

Values

$v ::= b \mid n \mid \textbf{skip} \mid (\textbf{fn}\ x : T \Rightarrow E)$

SOS rules

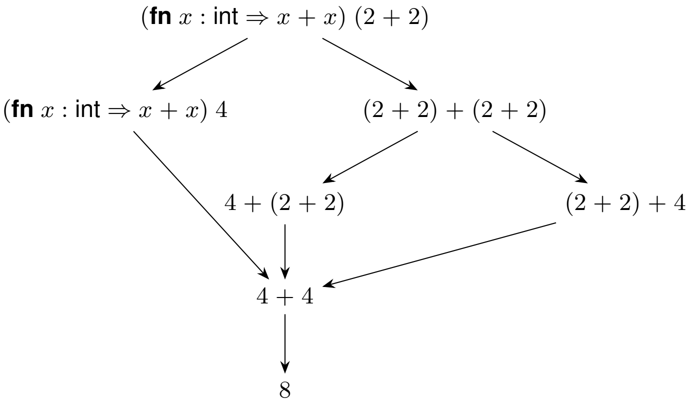
(beta-app1) $$\frac{\langle E_1\ ,\ s\rangle \longrightarrow \langle E_1'\ ,\ s'\rangle}{\langle E_1\ E_2\ ,\ s\rangle \longrightarrow \langle E_1'\ E_2\ ,\ s'\rangle}$$

(beta-app2) $$\frac{\langle E_2\ ,\ s\rangle \longrightarrow \langle E_2'\ ,\ s'\rangle}{\langle E_1\ E_2\ ,\ s\rangle \longrightarrow \langle E_1\ E_2'\ ,\ s'\rangle}$$

(beta-fn1) $$\langle (\textbf{fn}\ x : T \Rightarrow E_1)\ E_2\ ,\ s\rangle \longrightarrow \langle \{E_2/x\}\ E_1\ ,\ s\rangle$$

(beta-fn2) $$\frac{\langle E\ ,\ s\rangle \longrightarrow \langle E'\ ,\ s'\rangle}{\langle (\textbf{fn}\ x : T \Rightarrow E)\ ,\ s\rangle \longrightarrow \langle (\textbf{fn}\ x : T \Rightarrow E')\ ,\ s'\rangle}$$

Full Beta – Example



Choice 4: Normal-Order Reduction

Idea: leftmost, outermost variant of full beta.