**Harvard School of Engineering and Applied Sciences — CS 152: Programming Languages**

# Dynamic types, Lambda calculus machines
# Section and Practice Problems

Apr 21–22, 2016

---

## 1 Dynamic types and contracts

(a) To make sure you understand the operational semantics of dynamic types and exceptions, show the execution of the following program under the semantics of Section 1 of the Lecture 22 notes.

$$\begin{array}{l}\mathsf{let}\ f = \lambda x.\, 42 + x\ \mathsf{in}\\ \mathsf{let}\ g = \lambda y.\, (y\ \mathsf{true}) + 42\ \mathsf{in}\\ g\ f\end{array}$$

**Answer:**

$$\begin{array}{rl} & \mathit{let}\ f = \lambda x.\, 42 + x\ \mathit{in}\ \mathit{let}\ g = \lambda y.\, (y\ \mathsf{true}) + 42\ \mathit{in}\ g\ f\\ \longrightarrow & \mathit{let}\ g = \lambda y.\, (y\ \mathsf{true}) + 42\ \mathit{in}\ g\ (\lambda x.\, 42 + x)\\ \longrightarrow & (\lambda y.\, (y\ \mathsf{true}) + 42)\ (\lambda x.\, 42 + x)\\ \longrightarrow & ((\lambda x.\, 42 + x)\ \mathsf{true}) + 42\\ \longrightarrow & (42 + \mathsf{true}) + 42\\ \longrightarrow & \mathsf{Err} + 42\\ \longrightarrow & \mathsf{Err}\end{array}$$

(b) Modify the program from question (a) by adding appropriate error handlers (i.e., expressions of the form try $e_1$ catch $x.\ e_2$ to catch the type error and return the integer $42$ as the final result of the program. There are multiple places in the program where you can insert an error handler to achieve the desired result. Show three variations and their executions. (Note that the semantics for the execution of your programs is from Section 2 of the Lecture 22 notes.)

**Answer:** *Here is a version where we add an error handler in the body of function $f$.*

$$\begin{array}{l}\mathit{let}\ f = \lambda x.\, (\mathsf{try}\ (42 + x)\ \mathsf{catch}\ z.\, 0)\ \mathit{in}\\ \mathit{let}\ g = \lambda y.\, (y\ \mathsf{true}) + 42\ \mathit{in}\\ g\ f\end{array}$$

$$\begin{array}{rl} & \mathit{let}\ f = \lambda x.\, (\mathsf{try}\ (42 + x)\ \mathsf{catch}\ z.\, 0)\ \mathit{in}\ \mathit{let}\ g = \lambda y.\, (y\ \mathsf{true}) + 42\ \mathit{in}\ g\ f\\ \longrightarrow & \mathit{let}\ g = \lambda y.\, (y\ \mathsf{true}) + 42\ \mathit{in}\ g\ (\lambda x.\, (\mathsf{try}\ (42 + x)\ \mathsf{catch}\ z.\, 0))\\ \longrightarrow & (\lambda y.\, (y\ \mathsf{true}) + 42)\ (\lambda x.\, (\mathsf{try}\ (42 + x)\ \mathsf{catch}\ z.\, 0))\\ \longrightarrow & ((\lambda x.\, (\mathsf{try}\ (42 + x)\ \mathsf{catch}\ z.\, 0))\ \mathsf{true}) + 42\\ \longrightarrow & (\mathsf{try}\ (42 + \mathsf{true})\ \mathsf{catch}\ z.\, 0) + 42\\ \longrightarrow & (\mathsf{try}\ (\mathsf{Err}\ 1)\ \mathsf{catch}\ z.\, 0) + 42\end{array}$$

$\longrightarrow 0 + 42$
$\longrightarrow 42$

*Here's another version, where we add an error handler in the body of function $g$.*

$$\begin{array}{l}\textit{let } f = \lambda x.\, 42 + x \textit{ in} \\ \textit{let } g = \lambda y.\, (\mathsf{try}\ (y\ \mathsf{true})\ \mathsf{catch}\ z.\ 0) + 42 \textit{ in} \\ g\ f\end{array}$$

$\textit{let } f = \lambda x.\, 42 + x \textit{ in let } g = \lambda y.\, (\mathsf{try}\ (y\ \mathsf{true})\ \mathsf{catch}\ z.\ 0) + 42 \textit{ in } g\ f$
$\longrightarrow \textit{let } g = \lambda y.\, (\mathsf{try}\ (y\ \mathsf{true})\ \mathsf{catch}\ z.\ 0) + 42 \textit{ in } g\ (\lambda x.\, 42 + x)$
$\longrightarrow (\lambda y.\, (\mathsf{try}\ (y\ \mathsf{true})\ \mathsf{catch}\ z.\ 0) + 42)\ (\lambda x.\, 42 + x)$
$\longrightarrow (\mathsf{try}\ (\lambda x.\, 42 + x)\ \mathsf{true})\ \mathsf{catch}\ z.\ 0) + 42$
$\longrightarrow (\mathsf{try}\ (42 + \mathsf{true})\ \mathsf{catch}\ z.\ 0) + 42$
$\longrightarrow (\mathsf{try}\ (\mathsf{Err}\ 1)\ \mathsf{catch}\ z.\ 0) + 42$
$\longrightarrow 0 + 42$
$\longrightarrow 42$

*Finally, here is a version where we put the error handling code at the top level.*

$$\begin{array}{l}\textit{let } f = \lambda x.\, 42 + x \textit{ in} \\ \textit{let } g = \lambda y.\, (y\ \mathsf{true}) + 42 \textit{ in} \\ \mathsf{try}\ g\ f\ \mathsf{catch}\ z.\ 42\end{array}$$

$\textit{let } f = \lambda x.\, 42 + x \textit{ in let } g = \lambda y.\, (y\ \mathsf{true}) + 42 \textit{ in } \mathsf{try}\ g\ f\ \mathsf{catch}\ z.\ 42$
$\longrightarrow \textit{let } g = \lambda y.\, (y\ \mathsf{true}) + 42 \textit{ in } \mathsf{try}\ g\ (\lambda x.\, 42 + x)\ \mathsf{catch}\ z.\ 42$
$\longrightarrow \mathsf{try}\ (\lambda y.\, (y\ \mathsf{true}) + 42)\ (\lambda x.\, 42 + x)\ \mathsf{catch}\ z.\ 42$
$\longrightarrow \mathsf{try}\ (((\lambda x.\, 42 + x)\ \mathsf{true}) + 42)\ \mathsf{catch}\ z.\ 42$
$\longrightarrow \mathsf{try}\ ((42 + \mathsf{true}) + 42)\ \mathsf{catch}\ z.\ 42$
$\longrightarrow \mathsf{try}\ ((\mathsf{Err}\ 1) + 42)\ \mathsf{catch}\ z.\ 42$
$\longrightarrow \mathsf{try}\ (\mathsf{Err}\ 1)\ \mathsf{catch}\ z.\ 42$
$\longrightarrow 42$

(c) Modify the program from question (a) by adding appropriate dynamic type checks to raise the error as early as possible. When does your program detect the error?

**Answer:** *This version of the code adds dynamic type checks on all arguments and on results of function applications.*

$$\begin{array}{l}\textit{let } f = \lambda x.\, \mathsf{if}\ (\mathsf{is\_int?}\ \ x)\ \mathsf{then}\ 42 + x\ \mathsf{else}\ \mathsf{raise}\ 3 \textit{ in} \\ \textit{let } g = \lambda y.\, \mathsf{if}\ (\mathsf{is\_fun?}\ \ y)\ \mathsf{then} \\ \qquad\qquad \textit{let } y' = (y\ \mathsf{true}) \textit{ in } \mathsf{if}\ (\mathsf{is\_int?}\ \ y')\ \mathsf{then}\ y' + 42\ \mathsf{else}\ \mathsf{raise}\ 3 \\ \qquad \mathsf{else}\end{array}$$

raise 3 *in*
*let* $a = g\ f$ *in* if (is_int? $a$) then $a$ else raise 3

*The execution detects the error as soon as function $f$ is invoked, i.e.,* is_int? $x$ *evaluates to false when $x$ is replaced with* true.

(d) Modify the program from question (a) by adding contracts that specify the types of the input and output of $f$ and $g$. Show the execution of the modified program.

**Answer:**

*let* $f$ = monitor($\lambda x.\ 42 + x$, is_int? $\longmapsto$ is_int? ) *in*
*let* $g$ = monitor($\lambda y.\ (y\ \text{true}) + 42$, (is_bool? $\longmapsto$ is_int? ) $\longmapsto$ is_int? ) *in*
$g\ f$

(e) To make sure you understand the operational semantics of contracts, show the execution of the following program.

let $f$ = monitor($\lambda x.\ x + 1$, is_int? $\longmapsto$ is_int? ) in
let $g$ = monitor($\lambda x.\ x\ (42) + 42$, (is_int? $\longmapsto$ is_int? ) $\longmapsto$ is_int? ) in
let $h$ = monitor($\lambda x.\ \lambda y.\ y\ 42$, is_int? $\longmapsto$ ((is_int? $\longmapsto$ is_bool? ) $\longmapsto$ is_bool? )) in
$h\ (g\ f)\ (\lambda z.\ z + 42)$

**Answer:** *This answer is not yet done...*

## 2 Lambda calculus machines

Consider the following program in the language from the lecture notes (Lecture 23):

$$((((\lambda f.\ \lambda g.\ \lambda x.\ \lambda y.\ f\ x + g\ y)\ \lambda a.\ 0)\ \lambda b.\ 42)\ 43)\ 44$$

Show its evaluation trace under CC, SCC, CK and CEK semantics.

**Answer:**

*The CC machine:*

$$
\begin{array}{lll}
 & \langle(((((\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y)\;\lambda a.\,0)\;\lambda b.\,42)\;43)\;44 & ,[\cdot]\rangle \\
\longrightarrow & \langle((((\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y)\;\lambda a.\,0)\;\lambda b.\,42)\;43 & ,[\cdot]\;44\rangle \\
\longrightarrow & \langle(((\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y)\;\lambda a.\,0)\;\lambda b.\,42 & ,([\cdot]\;43)\;44\rangle \\
\longrightarrow & \langle((\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y)\;\lambda a.\,0 & ,(([\cdot]\;\lambda b.\,42)\;43)\;44\rangle \\
\longrightarrow & \langle\lambda g.\,\lambda x.\,\lambda y.\,(\lambda a.\,0)\;x + g\; y & ,(([\cdot]\;\lambda b.\,42)\;43)\;44\rangle \\
\longrightarrow & \langle(\lambda g.\,\lambda x.\,\lambda y.\,(\lambda a.\,0)\;x + g\; y)\;\lambda b.\,42 & ,([\cdot]\;43)\;44\rangle \\
\longrightarrow & \langle\lambda x.\,\lambda y.\,(\lambda a.\,0)\;x + (\lambda b.\,42)\;y & ,([\cdot]\;43)\;44\rangle \\
\longrightarrow & \langle(\lambda x.\,\lambda y.\,(\lambda a.\,0)\;x + (\lambda b.\,42)\;y)\;43 & ,44\rangle \\
\longrightarrow & \langle\lambda y.\,(\lambda a.\,0)\;43 + (\lambda b.\,42)\;y & ,[\cdot]\;44\rangle \\
\longrightarrow & \langle(\lambda y.\,(\lambda a.\,0)\;43 + (\lambda b.\,42)\;y)\;44 & ,[\cdot]\rangle \\
\longrightarrow & \langle(\lambda a.\,0)\;43 + (\lambda b.\,42)\;44 & ,[\cdot]\rangle \\
\longrightarrow & \langle(\lambda a.\,0)\;43 & ,[\cdot] + (\lambda b.\,42)\;44\rangle \\
\longrightarrow & \langle 0 & ,[\cdot] + (\lambda b.\,42)\;44\rangle \\
\longrightarrow & \langle 0 + (\lambda b.\,42)\;44 & [\cdot]\rangle \\
\longrightarrow & \langle(\lambda b.\,42)\;44 & 0 + [\cdot]\rangle \\
\longrightarrow & \langle 42 & 0 + [\cdot]\rangle \\
\longrightarrow & \langle 0 + 42 & [\cdot]\rangle \\
\longrightarrow & \langle 42 & [\cdot]\rangle
\end{array}
$$

*The SCC machine:*

$$
\begin{array}{lll}
 & \langle(((((\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y)\;\lambda a.\,0)\;\lambda b.\,42)\;43)\;44 & ,[\cdot]\rangle \\
\longrightarrow & \langle((((\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y)\;\lambda a.\,0)\;\lambda b.\,42)\;43 & ,[\cdot]\;44\rangle \\
\longrightarrow & \langle(((\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y)\;\lambda a.\,0)\;\lambda b.\,42 & ,([\cdot]\;43)\;44\rangle \\
\longrightarrow & \langle((\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y)\;\lambda a.\,0 & ,(([\cdot]\;\lambda b.\,42)\;43)\;44\rangle \\
\longrightarrow & \langle(\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y)\;\lambda a.\,0 & ,(((([\cdot]\;\lambda a.\,0)\;\lambda b.\,42)\;43)\;44\rangle \\
\longrightarrow & \langle\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y & ,((((\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y)\;[\cdot])\;\lambda b.\,42)\;43)\;44\rangle \\
\longrightarrow & \langle\lambda a.\,0 & ,(((([\cdot]\;\lambda b.\,42)\;43)\;44\rangle \\
\longrightarrow & \langle\lambda g.\,\lambda x.\,\lambda y.\,(\lambda a.\,0)\;x + g\; y & ,((((\lambda g.\,\lambda x.\,\lambda y.\,(\lambda a.\,0)\;x + g\; y)\;[\cdot])\;43)\;44\rangle \\
\longrightarrow & \langle\lambda b.\,42 & ,(((\lambda x.\,\lambda y.\,(\lambda a.\,0)\;x + (\lambda b.\,42)\;y)\;[\cdot])\;44\rangle \\
\longrightarrow & \langle\lambda x.\,\lambda y.\,(\lambda a.\,0)\;x + (\lambda b.\,42)\;y & ,([\cdot]\;44\rangle \\
\longrightarrow & \langle 43 & ,(\lambda y.\,(\lambda a.\,0)\;43 + (\lambda b.\,42)\;y)\;[\cdot]\rangle \\
\longrightarrow & \langle\lambda y.\,(\lambda a.\,0)\;43 + (\lambda b.\,42)\;y & ,[\cdot]\rangle \\
\longrightarrow & \langle 44 & ,[\cdot] + (\lambda b.\,42)\;44\rangle \\
\longrightarrow & \langle(\lambda a.\,0)\;43 + (\lambda b.\,42)\;44 & ,([\cdot]\;43) + (\lambda b.\,42)\;44\rangle \\
\longrightarrow & \langle(\lambda a.\,0)\;43 & ,((\lambda a.\,0)\;[\cdot]) + (\lambda b.\,42)\;44\rangle \\
\longrightarrow & \langle\lambda a.\,0 & ,[\cdot] + (\lambda b.\,42)\;44\rangle \\
\longrightarrow & \langle 43 & ,0 + [\cdot]\rangle \\
\longrightarrow & \langle 0 & ,0 + ([\cdot]\;44)\rangle \\
\longrightarrow & \langle(\lambda b.\,42)\;44 & ,0 + ((\lambda b.\,42)\;[\cdot])\rangle \\
\longrightarrow & \langle\lambda b.\,42 & ,0 + [\cdot]\rangle \\
\longrightarrow & \langle 44 & ,[\cdot]\rangle \\
\longrightarrow & \langle 42 & \\
\longrightarrow & \langle 42 &
\end{array}
$$

*The SK machine (which is exactly analogous to the SCC machine, except the contexts are represented as continuations):*

$$
\begin{array}{ll}
 & \langle(((((\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y)\;\lambda a.\,0)\;\lambda b.\,42)\;43)\;44 \quad , \mathbf{mt}\rangle \\
\longrightarrow & \langle((((\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y)\;\lambda a.\,0)\;\lambda b.\,42)\;43 \quad , \langle\mathbf{arg}, 44, \mathbf{mt}\rangle\rangle \\
\longrightarrow & \langle(((\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y)\;\lambda a.\,0)\;\lambda b.\,42 \quad , \langle\mathbf{arg}, 43, \langle\mathbf{arg}, 44, \mathbf{mt}\rangle\rangle\rangle \\
\longrightarrow & \langle((\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y)\;\lambda a.\,0 \quad , \langle\mathbf{arg}, \lambda b.\,42, \langle\mathbf{arg}, 43, \langle\mathbf{arg}, 44, \mathbf{mt}\rangle\rangle\rangle\rangle \\
\longrightarrow & \langle(\lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y \quad , \langle\mathbf{arg}, \lambda a.\,0, \langle\mathbf{arg}, \lambda b.\,42, \langle\mathbf{arg}, 43, \langle\mathbf{arg}, 44, \mathbf{mt}\rangle\rangle\rangle\rangle\rangle \\
\longrightarrow & \langle\lambda a.\,0 \quad , \langle\mathbf{fun}, \lambda f.\,\lambda g.\,\lambda x.\,\lambda y.\; f\; x + g\; y, \langle\mathbf{arg}, \lambda b.\,42, \langle\mathbf{arg}, 43, \langle\mathbf{arg}, 44, \mathbf{mt}\rangle\rangle\rangle \\
\longrightarrow & \langle\lambda g.\,\lambda x.\,\lambda y.\,(\lambda a.\,0)\; x + g\; y \quad , \langle\mathbf{arg}, \lambda b.\,42, \langle\mathbf{arg}, 43, \langle\mathbf{arg}, 44, \mathbf{mt}\rangle\rangle\rangle \\
\longrightarrow & \langle\lambda b.\,42 \quad , \langle\mathbf{fun}, \lambda g.\,\lambda x.\,\lambda y.\,(\lambda a.\,0)\; x + g\; y, \langle\mathbf{arg}, 43, \langle\mathbf{arg}, 44, \mathbf{mt}\rangle\rangle\rangle \\
\longrightarrow & \langle\lambda x.\,\lambda y.\,(\lambda a.\,0)\; x + (\lambda b.\,42)\; y \quad , \langle\mathbf{arg}, 43, \langle\mathbf{arg}, 44, \mathbf{mt}\rangle\rangle \\
\longrightarrow & \langle 43 \quad , \langle\mathbf{fun}, \lambda x.\,\lambda y.\,(\lambda a.\,0)\; x + (\lambda b.\,42)\; y, \langle\mathbf{arg}, 44, \mathbf{mt}\rangle\rangle \\
\longrightarrow & \langle\lambda y.\,(\lambda a.\,0)\; 43 + (\lambda b.\,42)\; y \quad , \langle\mathbf{arg}, 44, \mathbf{mt}\rangle\rangle \\
\longrightarrow & \langle 44 \quad , \langle\mathbf{fun}, \lambda y.\,(\lambda a.\,0)\; 43 + (\lambda b.\,42)\; y, \mathbf{mt}\rangle\rangle \\
\longrightarrow & \langle(\lambda a.\,0)\; 43 + (\lambda b.\,42)\; 44 \quad , \mathbf{mt}\rangle\rangle \\
\longrightarrow & \langle(\lambda a.\,0)\; 43 \quad , \langle +, \langle\rangle, \langle(\lambda b.\,42)\; 44\rangle, \mathbf{mt}\rangle\rangle \\
\longrightarrow & \langle\lambda a.\,0 \quad , \langle\mathbf{arg}, 43, \langle +, \langle\rangle, \langle(\lambda b.\,42)\; 44\rangle, \mathbf{mt}\rangle\rangle \\
\longrightarrow & \langle 43 \quad , \langle\mathbf{fun}, \lambda a.\,0, \langle +, \langle\rangle, \langle(\lambda b.\,42)\; 44\rangle, \mathbf{mt}\rangle\rangle \\
\longrightarrow & \langle 0 \quad , \langle +, \langle\rangle, \langle(\lambda b.\,42)\; 44\rangle, \mathbf{mt}\rangle\rangle \\
\longrightarrow & \langle(\lambda b.\,42)\; 44 \quad , \langle +, \langle 0\rangle, \langle\rangle, \mathbf{mt}\rangle\rangle \\
\longrightarrow & \langle\lambda b.\,42 \quad , \langle\mathbf{arg}, 44, \langle +, \langle 0\rangle, \langle\rangle, \mathbf{mt}\rangle\rangle\rangle \\
\longrightarrow & \langle 44 \quad , \langle\mathbf{fun}, \lambda b.\,42, \langle +, \langle 0\rangle, \langle\rangle, \mathbf{mt}\rangle\rangle\rangle \\
\longrightarrow & \langle 42 \quad , \langle +, \langle 0\rangle, \langle\rangle, \mathbf{mt}\rangle\rangle\rangle \\
\longrightarrow & \langle 42 \quad , \mathbf{mt}\rangle
\end{array}
$$

*The answer for the CEK machine is not yet done...*