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

Dynamic Types, Concurrency, Type and effect system
Section and Practice Problems

Apr 24–27, 2018

---

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 Part 2 (Exception handling) of the Lecture 22 notes.)

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

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

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

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

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

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

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

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

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

(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.*

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

The execution detects the error as soon as function f is invoked, i.e., $\mathsf{is\_int?}\ \ x$ *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:

$$
\begin{array}{l}
\textit{let } f = \mathsf{monitor}(\lambda x.\ 42 + x, \mathsf{is\_int?} \longmapsto \mathsf{is\_int?}\ )\ \textit{in} \\
\textit{let } g = \mathsf{monitor}(\lambda y.\ (y\ \mathsf{true}) + 42, (\mathsf{is\_bool?} \longmapsto \mathsf{is\_int?}\ ) \longmapsto \mathsf{is\_int?}\ )\ \textit{in} \\
g\ f
\end{array}
$$

2 Concurrency

(a) Consider the following program.

$$(3 + 7)\ ||\ ((\lambda x.\ x + 1)\ 2 + 5)$$

Show an execution sequence for this program (i.e., give a sequence of expressions such that $e_0 \longrightarrow e_1 \longrightarrow ... \longrightarrow e_n$ where $e_0 = (3 + 7)\ ||\ ((\lambda x.\ x + 1)\ 2 + 5)$ and e_n is a value.

Answer: *This evaluation evaluates the left expression all the way to a value, and then the right.*

$$
\begin{array}{rl}
(3 + 7)\ ||\ ((\lambda x.\ x + 1)\ 2 + 5) & \longrightarrow 10\ ||\ ((\lambda x.\ x + 1)\ 2 + 5) \\
& \longrightarrow 10\ ||\ ((\lambda x.\ x + 1)\ 7) \\
& \longrightarrow 10\ ||\ 7 + 1 \\
& \longrightarrow 10\ ||\ 8 \\
& \longrightarrow (10, 8)
\end{array}
$$

Now give a different execution sequence for this program.

Answer: *This evaluation evaluates the right expression all the way to a value, and then the left.*

$$\begin{aligned}(3+7) \;||\; ((\lambda x.\, x+1)\; 2+5) &\longrightarrow 3+7 \;||\; ((\lambda x.\, x+1)\; 7) \\ &\longrightarrow 3+7 \;||\; 7+1 \\ &\longrightarrow 3+7 \;||\; 8 \\ &\longrightarrow 10 \;||\; 8 \\ &\longrightarrow (10, 8)\end{aligned}$$

How many different execution sequences of this program are there?

Answer: *The left expression just needs a single step to evaluate to a value, the right expression takes 3 steps. There are 4 different possible interleavings (i.e., for $i \in 0..3$ there is an interleaving where the left expression takes its single step after the right expression has evaluated i steps).*

(b) Consider the following program.

$$\begin{aligned}&\mathsf{let\ foo} = \mathsf{ref}\ 2\ \mathsf{in} \\ &\mathsf{let}\ y = (\mathsf{foo} := \mathsf{!foo} + \mathsf{!foo} \;||\; \mathsf{foo} := 1)\ \mathsf{in} \\ &\mathsf{!foo}\end{aligned}$$

What are the possible final values of the program?

Answer: *The left expression (*foo := !foo + !foo*) reads* foo *twice, and then updates it, and the right expression (*foo := 1*) just updates* foo*.*

There are four possible interleavings, which we show below, as well as the final value.

!foo *(=2)* !foo *(=2)* foo := ... *(=4)*	foo := 1

Final value: 1

!foo *(=2)* !foo *(=2)* foo := ... *(=4)*	foo := 1

Final value: 4

!foo *(=2)* !foo *(=1)* foo := ... *(=3)*	foo := 1

Final value: 3

!foo *(=1)* !foo *(=1)* foo := ... *(=2)*	foo := 1

Final value: 2

3 Type and effect system

Recall the type and effect system to ensure determinacy, covered in Lecture 25.

(a) Consider the program (from class) of a bank balance, where the bank balance is in the region A.

$$\mathsf{let\ bal} = \mathsf{ref}_A\ 0\ \mathsf{in}\ (\mathsf{let}\ y = (\mathsf{bal} := \mathsf{!bal} + 25 \;||\; \mathsf{bal} := \mathsf{!bal} + 50)\ \mathsf{in}\ \mathsf{!bal})$$

Try to produce a typing derivation for this program (using the type-and-effect typing rules from lecture). Where do the typing rules fail? Why?

Answer: *The typing derivation fails when we try to type the concurrent expression* bal := !bal + 25 || bal := !bal + 50*. Specifically, the read and write effect for both the left and write expression is $\{A\}$, the set containing the region A. Thus, the write set of* bal := !bal + 25 *intersects with the write set of* bal := !bal + 50*, and so we can't satisfy the premise.*

(b) Write a program that allocates two locations (in different regions) and reads and writes from both of them. Moreover, make sure that your program is well-typed according to the type-and-effect system.

Is your program deterministic?

Answer: *Here's one. It is well-typed and deterministic! (Indeed, all well-typed programs are deterministic.)*

$$
\begin{array}{l}
\textit{let}\ \mathsf{a} = \textit{ref}_A\ 0\ \textit{in} \\
\textit{let}\ \mathsf{b} = \textit{ref}_B\ 0\ \textit{in} \\
\textit{let}\ y = (\mathsf{a} := !\mathsf{a} + 25\ ||\ \mathsf{b} := !\mathsf{b} + 50)\ \textit{in} \\
!\mathsf{a} + !\mathsf{b}
\end{array}
$$