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

Dynamic Types
Section and Practice Problems

Week 13: Tue April 18–Fri April 21, 2023

---

1 Dynamic types and contracts

This material will be covered in Lecture on Tuesday April 30; we include it here so that later sections can be focused on comprehensive review.

(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 25 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 $\mathsf{try}\ e_1\ \mathsf{catch}\ x.\, e_2$ 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.*

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

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

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

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

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

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

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

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

(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.*

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

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:

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