

Suppose the expression e eventually signals an exception. Then the computation proceeds as follows:

$$\begin{aligned}
 & (\text{catch } e \text{ with } (\lambda x c. \text{if0 } x \text{ (c inf) (throw } x))) \\
 = & (\text{catch } F[(/ \text{1} \text{0}]) \text{ with } (\lambda x c. \text{if0 } x \text{ (c inf) (throw } x))) \\
 = & (\text{catch } F[(\text{throw } \text{0}]) \text{ with } (\lambda x c. \text{if0 } x \text{ (c inf) (throw } x))) \\
 = & (\lambda x c. \text{if0 } x \text{ (c inf) (throw } x)) \text{0} (\lambda y. F[y]) \\
 = & (\text{if0 } \text{0} ((\lambda y. F[y]) \text{ inf) (throw } \text{0})) \\
 = & F[\text{inf}]
 \end{aligned}$$

In other words, the computation of e is resumed exactly where the division-by-zero error is discovered. The handler, however, is no longer available. Hence, if $F[\text{inf}]$ signals the exception again, the evaluation is stopped.

Exercise 8.22. Design a recursive function D that consumes a function f and some value v . The function D applies f to v and resumes this computation for all exceptions with `inf` until it produces a value. In particular, if $(f v)$ repeatedly raises an exception, the call to D never terminates.

Let us formulate the syntax and semantics of Control ISWIM, a programming language that gives programs complete power over the control flow.

$ \begin{array}{l} M = \dots \\ \quad (\text{throw } b) \\ \quad (\text{catch } M \text{ with } \lambda X. \lambda Y. M) \end{array} $	same as plain ISWIM
--	---------------------

Like Handler ISWIM, Control ISWIM extends ISWIM with a `catch` and a `throw` expression. Indeed, the only difference is that, in Control ISWIM, the former requires a handler of two arguments instead of one; otherwise, the syntax of Control ISWIM is just that of Handler ISWIM. And just like for Handler ISWIM, we deal with the handler sub-expressions also as if it were a plain λ expression, especially as far as substitution is concerned.

$ \begin{aligned} & (\text{catch } M \text{ with } (\lambda X_1. \lambda Y. M_1))[X_2 \leftarrow M_2] \\ = & (\text{catch } M[X_2 \leftarrow M_2] \text{ with } (\lambda X_1. \lambda Y. M_1)[X_2 \leftarrow M_2]) \end{aligned} $

The calculus for Control ISWIM is a seemingly simple variation of the one for Handler ISWIM. We start with its definition of F contexts.

$F = [] \mid (V F) \mid (F M) \mid (o^m V \dots V F M \dots M)$

Below we list the new notions of reduction. Since `throw` and `catch` collaborate, the **throw** relation is integrated into **cntrl**, a revision of **catch**. All others look are adapted to the syntax of Control ISWIM.

$$\begin{array}{c}
\mathbf{c} = \beta_v \cup \delta \cup \delta_{\text{err}} \cup \mathbf{return} \cup \mathbf{cntrl} \\
\\
(\text{catch } F[(\text{throw } b)] \text{ with } \lambda XZ.M) \quad \mathbf{cntrl} \quad (\lambda XZ.M) b \ (\lambda Y.F[Y]) \\
\quad \text{if } Y \notin \mathcal{FV}(F[{}^1 5]) \\
(\text{catch } V \text{ with } \lambda X.M) \quad \mathbf{return} \quad V \\
\quad (o^m b_1 \dots b_m) \quad \delta_{\text{err}} \quad (\text{throw } b) \\
\quad \text{if } \delta(o^m, b_1, \dots, b_m) = \text{err}_b \\
(o^m b_1 \dots b_{i-1} (\lambda X.N) V_{i+1} \dots V_m) \quad \delta_{\text{err}} \quad (\text{throw } {}^1 i) \\
\quad (b V) \quad \delta_{\text{err}} \quad (\text{throw } b)
\end{array}$$

As usual, \rightarrow_c is the compatible closure of \mathbf{c} ; \rightarrow_c^* is the transitive-reflexive closure of \rightarrow_c ; and $=_c$ is its reflexive-transitive-symmetric closure.

The evaluation function for Control ISWIM has the same range as the one for Handler ISWIM but its definition is somewhat unusual.

$$\begin{array}{l}
eval_c : M \longrightarrow A_e \\
eval_c(M) = \begin{cases} b & \text{if } M =_c b \\ \text{function} & \text{if } M =_c \lambda X.N \\ \text{err}_b & \text{if } M =_c F[(\text{throw } b)] \end{cases}
\end{array}$$

Specifically, it requires a special case to signal errors because $(\text{throw } b)$ without a surrounding catch creates a new kind of irreducible expression.

Theorem 8.17 [Consistency for Control ISWIM]: The relation $eval_c$ is a partial function.

Exercise 8.23. Prove theorem 8.17.

Standard Reduction for Control ISWIM. Following our usual program, we next adapt the standard reduction theorem of Handler ISWIM to Control ISWIM. Doing so requires an adaptation of two definitions: the one for the set of contexts in which standard reduction steps take place and the standard reduction function.

Both adaptations are straightforward. As far as contexts are concerned, there is only change required. Since the syntax of catch expressions in Control ISWIM demands a handler function of two parameters, the contexts for evaluating protected bodies must do so, too.

$$\begin{array}{l}
E = \dots \quad \text{same as plain ISWIM} \\
| \quad (\text{catch } E \text{ with } \lambda X.\lambda Y.M)
\end{array}$$

The standard reduction function needs to grab the current continuation and immediately supply it to the exception handler when an exception is signaled. Otherwise, Control ISWIM's standard reduction function is adapted from Handler ISWIM's.

$$\tilde{c} = \beta_v \cup \delta \cup \delta_{\text{err}} \cup \mathbf{return}$$

$$\begin{array}{l} E[M] \quad \quad \quad \longmapsto_c \quad E[N] \\ \text{if } M \tilde{c} N \end{array}$$

$$E \left[\begin{array}{l} (\text{catch } F[(\text{throw } b)]) \\ \text{with } \lambda X Y.M \end{array} \right] \longmapsto_c E[(\lambda X Y.M) b (\lambda Z.F[Z])]$$

The definition of the standard reduction function for Control ISWIM now looks just like the plain evaluator for Control ISWIM, except that it uses the standard reduction function.

$$\text{eval}_c^{\tilde{c}}(M) = \begin{cases} b & \text{if } M \longmapsto_c b \\ \text{function} & \text{if } M \longmapsto_c \lambda X.N \\ \text{err}_b & \text{if } M \longmapsto_c F[(\text{throw } b)] \end{cases}$$

It is now time to re-state the two correctness theorems, starting with the unique evaluation lemma.

Lemma 8.18 [Unique Evaluation Contexts]: For every closed Control ISWIM expression M , one of the following is true:

- M is a value or for some b , $M = (\text{throw } b)$.
- There exist a unique standard context E and a standard re-
dex M_{rh} such that $M = E[M_{\text{rh}}]$ where

$$\begin{array}{l} M_{\text{rh}} = (V \ V) \\ \quad | \quad (o^m \ V \ \dots \ V) \\ \quad | \quad (\text{catch } V \ \text{with } \lambda X \ Y.M) \end{array}$$

- There exist a unique standard context E and context F such that $M = E[(\text{catch } F[(\text{throw } b)] \ \text{with } \lambda X \ Y.M)]$.
- There exists a non-empty F such that $M = F[(\text{throw } b)]$.

Note the subtle but important differences between this theorem and the corresponding theorem for Handler ISWIM (theorem 8.13, page 135). In particular, it is no longer possible to partition an expression into a throw expression