Decorated semantics for an imperative language with exceptions

Dominique Duval,
with Jean-Guillaume Dumas, Burak Ekici,
Damien Pous and Jean-Claude Reynaud

Work in progress

GdT Plume, ENS Lyon, 21 mars 2016

The language IMP-EX

Syntax

Arithmetic expressions:

$$a \ ::= \ 0 \mid 1 \mid -1 \mid 2 \mid -2 \mid \dots \mid \ell_1 \mid \ell_2 \mid \cdots \mid a+a \mid a-a \mid a \times a$$

Boolean expressions:

$$b \ ::= \ \mathtt{true} \mid \mathtt{false} \mid \neg b \mid b \wedge b \mid b \vee b \mid a = a \mid a > a$$

Commands:

$$c \ ::= \ \begin{cases} \mathtt{skip} \mid c; c \mid \ell_i := a \mid \\ \mathtt{if}(b)\mathtt{then}(c)\mathtt{else}(c) \mid \mathtt{while}(b)\mathtt{do}(c) \mid \\ \mathtt{throw}(exn_i) \mid \mathtt{try}(c)\mathtt{catch}(exn_i \Rightarrow c) \end{cases}$$

Programs:

$$pg \ ::= \ c; \mathtt{return}(a) \mid c; \mathtt{return}(b)$$

Semantics

Denotational: in the category of sets and partial functions

Operational: small-step, big-step

Predicate transformer semantics, ...

Theorem *"All semantics for IMP-EX coincide".*

Aims and tools

Aims.

- Design a "kind of" equational logic $\mathcal{L}$, close to the syntax, for reasoning about imperative programs with exceptions.
- Translate the syntax of IMP-EX into the logic $\mathcal{L}$.
- Prove properties of programs of IMP-EX in the logic $\mathcal{L}$.
- Implement this proof system in Coq.

Tools.

- [Moggi 1989] "effects as monads".
  Terms of type B with a parameter of type A
  are not interpreted by morphisms from A to B
  but by morphisms from A to $T(B)$ for some monad T.
- Here, more generally, "effects as functors".
  Terms of type B with a parameter of type A
  are not interpreted by morphisms from A to B
  but by morphisms from $H(A)$ to $H(B)$ for some functor H.

Outline

Logic and categories

- Syntax and equational semantics: a theory $\mathcal{T}h$
  (a category with a congruence $\equiv$)
  generated by a signature and equations.
- Denotational semantics: a model $M : \mathcal{T}h \to \mathcal{C}$
  (a functor mapping $\equiv$ to $=$)
  where $\mathcal{C}$ is "given by mathematics"
  (e.g., $\mathcal{C} = \mathcal{S}et$ or $\mathcal{C} = \mathcal{P}art$).

$$\begin{array}{ccc}
\begin{tikzcd} \text{theory} \arrow[d, "\text{model}"'] \\ \text{category} \end{tikzcd}
&
\begin{tikzcd} \mathcal{T}h \arrow[d, "M"'] \\ \mathcal{C} \end{tikzcd}
&
\begin{tikzcd} 1+2 \arrow[d, mapsto] & \equiv & 3 \arrow[d, mapsto] \\ 1+2 & = & 3 \end{tikzcd}
\end{array}$$

Soundness: granted

Remark: usually *structured* categories and functors

Decorated logic: theories and models

Simply "enlarge" the previous diagram

$$\begin{tikzcd}
\mathcal{T}h_0 \arrow[r, "\subseteq"] \arrow[d, "M_0"'] & \mathcal{T}h_1 \arrow[r, "\subseteq"] \arrow[dd, "M_1"'] & \cdots \arrow[r, "\subseteq"] & \mathcal{T}h_n \arrow[dddd, "M_n"] \\
\mathcal{C}_0 \arrow[drr, "H_1"'] & & & \\
& \mathcal{C}_1 \arrow[dr, "H_2"'] & & \\
& & \cdots \arrow[dr, "H_n"'] & \\
& & & \mathcal{C}_n
\end{tikzcd}$$

where the functor $\mathcal{T}h_{i-1} \subseteq \mathcal{T}h_i$

- is the identity on objects
- preserves $\equiv$ and is "$\equiv$-faithful": for all $f, g : X \to Y$ in $\mathcal{T}h_{i-1}$

$$f \equiv g \text{ in } \mathcal{T}h_{i-1} \iff f \equiv g \text{ in } \mathcal{T}h_i$$

Decoration of terms (notation): $f^{(d)}$ iff $f \in \mathcal{T}h_d$

conversions: $f^{(d)} \implies f^{(d+1)}$

Soundness: if each H_i is faithful

Full image

The full image of a functor $H : \mathcal{C}_{i-1} \to \mathcal{C}_i$ is the category $\overline{im}(H)$ with:

- the same objects as $\mathcal{C}_{i-1}$
- an arrow $f : X \to Y$ for each $f : H(X) \to H(Y)$ in $\mathcal{C}_i$.

$$
\begin{tikzcd}
\mathcal{T}h_{i-1} \arrow[r, "\subseteq"] \arrow[d, "M_{i-1}"'] & \mathcal{T}h_i \arrow[d] \arrow[dd, bend left, "M_i"] \\
\mathcal{C}_{i-1} \arrow[r, "\overline{H}"] \arrow[dr, "H"'] & \overline{im}(H) \arrow[d] \\
& \mathcal{C}_i
\end{tikzcd}
\qquad
\begin{tikzcd}
(\underline{X} \to \underline{Y}) \arrow[r, mapsto] \arrow[d, mapsto] & (\underline{X} \to \underline{Y}) \arrow[d, mapsto] \\
(X \to Y) \arrow[r, mapsto] & (X \to Y) \arrow[d, mapsto] \\
& (HX \to HY)
\end{tikzcd}
$$

Soundness: if $\overline{H}$ is faithful

Kleisli category

The Kleisli category of a monad $T : \mathcal{C} \to \mathcal{C}$ is the category $\mathcal{C}_T$ with:

- the same objects as $\mathcal{C}$
- an arrow $f : X \to Y$ for each $f : X \to T(Y)$ in $\mathcal{C}$.

$$
\begin{tikzcd}
\mathcal{T}h_{i-1} \arrow[r, "\subseteq"] \arrow[d, "M_{i-1}"'] & \mathcal{T}h_i \arrow[r, "\subseteq"] \arrow[d, "M_i"'] & \mathcal{T}h_{i+1} \arrow[d] \arrow[dd, bend left, "M_{i+1}"] \\
\mathcal{C} \arrow[r] \arrow[rrd, "T"'] & \mathcal{C}_T \arrow[r] & \overline{im}(T) \arrow[d] \\
 & & \mathcal{C}
\end{tikzcd}
\qquad
\begin{tikzcd}
(\underline{X} \to \underline{Y}) \arrow[r, mapsto] \arrow[d, mapsto] & (\underline{X} \to \underline{Y}) \arrow[r, mapsto] \arrow[d, mapsto] & (\underline{X} \to \underline{Y}) \arrow[d, mapsto] \\
(X \to Y) \arrow[r, mapsto] & (X \to Y) \arrow[r, mapsto] \arrow[d, mapsto, dotted] & (X \to Y) \arrow[d, mapsto] \\
 & (X \to TY) \arrow[r, mapsto, dotted] & (TX \to TY)
\end{tikzcd}
$$

Soundness: if each component of the unit $\eta : Id \Rightarrow T$ is mono

Decorated logic: decorated equations

Notation: $\boxed{f \bullet g = g \circ f}$ when $\bullet \xrightarrow{f} \bullet \xrightarrow{g} \bullet$

In each theory:

- a congruence $\equiv$:
  - equivalence relation between parallel terms
  - compatible with composition

    $$g_1 \equiv g_2 \implies f \bullet g_1 \bullet h \equiv f \bullet g_2 \bullet h$$

- a *weak* congruence (or several):
  - extends $\equiv$
  - preorder relation between parallel terms
  - "sometimes" symmetry
  - "sometimes" substitution

    $$g_1 \equiv g_2 \implies f \bullet g_1 \equiv f \bullet g_2$$

  - "sometimes" replacement

    $$g_1 \equiv g_2 \implies g_1 \bullet h \equiv g_2 \bullet h$$

Outline

The language XS-IMP

Syntax

Expressions:

$$
\begin{aligned}
a &::= 0 \mid 1 \mid -1 \mid 2 \mid -2 \mid \ldots \mid \ell \mid s(a) \mid p(a) \\
b &::= \mathtt{true} \mid \mathtt{false} \mid \neg b \mid a = 0 \mid a > 0 \\
e &::= a \mid b
\end{aligned}
$$

Commands:

$$c ::= \mathtt{skip} \mid c; c \mid \ell := a$$

Programs:

$$pg ::= c; \mathtt{return}(e)$$

Restrictions (easy to remove):

- only one location ℓ
- no binary operation on expressions

Later:

- exceptions, conditionals, loops

Decorated logic for states

Comonad $\boxed{D(X) = S \times X}$

$$
\begin{tikzcd}
\mathcal{T}h_0 \arrow[r, "\subseteq"] & \mathcal{T}h_1 \arrow[r, "\subseteq"] & \mathcal{T}h_2
\end{tikzcd}
$$

$$
\begin{tikzcd}
(X \to Y) \arrow[r, mapsto] \arrow[d, mapsto] & (X \to Y) \arrow[r, mapsto] \arrow[d, mapsto] & (X \to Y) \arrow[d, mapsto] \\
(X \to Y) \arrow[r, mapsto] & (X \to Y) \arrow[r, mapsto] \arrow[d, mapsto, dotted] & (X \to Y) \arrow[d, mapsto] \\
 & (S \times X \to Y) \arrow[r, mapsto, dotted] & (S \times X \to S \times Y)
\end{tikzcd}
$$

Weak equations $\boxed{f_1 \sim_{st} f_2 : X \to Y}$

interpreted as: $f_1 \bullet \varepsilon_Y = f_2 \bullet \varepsilon_Y : S \times X \to Y$

$$
\begin{tikzcd}
S \times X \arrow[r, shift left, "f_1"] \arrow[r, shift right, "f_2"'] & S \times Y \arrow[r, "\varepsilon_Y"] & Y
\end{tikzcd}
$$

$\sim_{st}$ satisfies substitution and pure replacement:

$$g_1 \sim_{st} g_2 \implies f \bullet g_1 \bullet h^{(0)} \sim_{st} f \bullet g_2 \bullet h^{(0)}$$

Pure operations and equations

The pure theory $\mathcal{T}h_0$ contains:

- sorts $\mathbb{1}$, A, B
- operations $0, 1, -1, \ldots : \mathbb{1} \to A$, $s, p : A \to A$, $\mathtt{true}, \mathtt{false} : \mathbb{1} \to B$, $\mathtt{not} : B \to B$, $\mathtt{null?}, \mathtt{pos?} : A \to B$
- equations $s(0) \equiv 1$, $p(0) \equiv -1$,..., $s \bullet p \equiv id_A$, $p \bullet s \equiv id_A$, $\mathtt{true} \bullet \mathtt{not} \equiv \mathtt{false}$, ...

$M_0 : \mathcal{T}h_0 \to \mathcal{S}et$ interprets A as the set A of integers,
B as the set B of truth values, etc

Operations and equations for states

In $\mathcal{S}et$: a set of states S with (here) $S \cong A$, denoted $\boxed{x} \leftrightarrow x$
Then $\mathcal{T}h_1$ and $\mathcal{T}h_2$ are generated from $\mathcal{T}h_0$ by
two operations:

$\mathtt{lookup}^{(1)} : \mathbb{1} \to A$	$\mathtt{update}^{(2)} : A \to \mathbb{1}$
$\mathtt{lookup} : S \to A$ $\mathtt{lookup} : \boxed{x} \mapsto x$	$\mathtt{update} : S \times A \to S$ $\mathtt{update} : (\boxed{x}, y) \mapsto \boxed{y}$

one weak equation:

$\mathtt{update} \bullet \mathtt{lookup} \sim_{st} id_A$
$\mathtt{update} \bullet \mathtt{lookup} = \varepsilon_A$ $(\boxed{x}, y) \mapsto \boxed{y} \mapsto y$

and decorated rules...

Translation

Expressions: $\boxed{e \mapsto e^{(1)} : \mathbb{1} \to \mathit{Expr}}$ (where *Expr* is A or B)

- $0, 1, \ldots \mapsto 0^{(0)}, 1^{(0)}, \ldots, \mathtt{true}, \mathtt{false} \mapsto \mathtt{true}^{(0)}, \mathtt{false}^{(0)}$
- $s(a) \mapsto a \bullet s^{(0)}$, $p(a) \mapsto a \bullet p^{(0)}$, $\neg b \mapsto b \bullet \mathtt{not}^{(0)}$, ...
- $\ell \mapsto \mathtt{lookup}^{(1)}$

Commands: $\boxed{c \mapsto c^{(2)} : \mathbb{1} \to \mathbb{1}}$

- $\mathtt{skip} \mapsto id_{\mathbb{1}}^{(0)}$
- $c_1; c_2 \mapsto c_1 \bullet c_2$
- $\ell := a \mapsto a \bullet \mathtt{update}^{(2)}$

Programs: $\boxed{pg \mapsto pg^{(2)} : \mathbb{1} \to \mathit{Expr}}$

- $c; \mathtt{return}(e) \mapsto c \bullet e$

Forward semantics

Given a program $pg^{(2)} : \mathbb{1} \to Expr$,
find a result $rs^{(0)} : \mathbb{1} \to Expr$ such that $\boxed{pg \sim_{st} rs}$
This means that $pg : S \to S \times Expr$ and $rs : \mathbb{1} \to Expr$ satisfy:

$pg(s) = (s', rs(x))$ for some s'

$$\begin{tikzcd} S \arrow[r, "pg"] \arrow[d, "\varepsilon_{\mathbb{1}}"'] \arrow[dr, phantom, "="] & S \times Expr \arrow[d, "\varepsilon_{Expr}"] \\ \mathbb{1} \arrow[r, "u"'] & Expr \end{tikzcd}$$

This requires an initialization of the state
and the derived strong equation: for each $u^{(0)} : \mathbb{1} \to A$

$$\boxed{u \bullet \mathtt{update} \bullet \mathtt{lookup} \equiv u \bullet \mathtt{update} \bullet u}$$

Method:

- first $\equiv$ is used inductively, by replacement
- until finally $\sim_{st}$ can be used, by pure replacement

This corresponds to an operational semantics.

Forward semantics: an example

Initialization: $\ell := u^{(0)}$ for any $u^{(0)} : \mathbb{1} \to A$

The given program is

$$\ell := u;\ \ell := s(\ell);\ \mathtt{return}(p(\ell))$$

translated as:

$$pg^{(2)} = u^{(0)} \bullet \mathtt{update}^{(2)} \bullet \mathtt{lookup}^{(1)} \bullet s^{(0)} \bullet \mathtt{update}^{(2)} \bullet \mathtt{lookup}^{(1)} \bullet p^{(0)}$$

$$\begin{tikzcd}
\mathbb{1} \arrow[r, "u"] \arrow[d, equal] & A \arrow[r, "\mathtt{update}"] \arrow[dr, phantom, "\equiv"] & \mathbb{1} \arrow[r, "\mathtt{lookup}"] & A \arrow[r, "s"] \arrow[d, equal] & A \arrow[r, "\mathtt{update}"] & \mathbb{1} \arrow[r, "\mathtt{lookup}"] & A \arrow[r, "p"] & A \\
\mathbb{1} \arrow[r, "u"] & A \arrow[r, "\mathtt{update}"] & \mathbb{1} \arrow[r, "u"] \arrow[d, equal] & A \arrow[r, "s"] & A \arrow[r, "\mathtt{update}"] \arrow[d, phantom, "\equiv"] & \mathbb{1} \arrow[r, "\mathtt{lookup}"] & A \arrow[r, "p"] \arrow[d, equal] & A \\
\mathbb{1} \arrow[r, "u"] & A \arrow[r, "\mathtt{update}"] & \mathbb{1} \arrow[r, "u"] & A \arrow[r, "s"] & A \arrow[r, "\mathtt{update}"] & \mathbb{1} \arrow[r, "u\bullet s"] \arrow[d, equal] & A \arrow[r, "p"] \arrow[d, phantom, "\equiv"] & A \arrow[d, equal] \\
\mathbb{1} \arrow[r, "u"] \arrow[d, equal] & A \arrow[r, "\mathtt{update}"] & \mathbb{1} \arrow[r, "u"] \arrow[dr, phantom, "\sim"] & A \arrow[r, "s"] & A \arrow[r, "\mathtt{update}"] & \mathbb{1} \arrow[rr, "u"] \arrow[d, equal] & & A \\
\mathbb{1} \arrow[rrrrr, "id"] & & & & & \mathbb{1} \arrow[rr, "u"] & & A
\end{tikzcd}$$

Conclusion: $pg^{(2)} \sim_{st} rs^{(0)}$ where $rs^{(0)} = u$. The result is u

Backward semantics

Given a program $pg = c; \mathtt{return}(post) : \mathbb{1} \to Expr$,
find an expression $pre : \mathbb{1} \to Expr$ such that $\boxed{pg \sim_{st} \mathtt{return}(pre)}$
This means that c, $post$ and pre satisfy:

$$post(c(s)) = pre(s)$$

$$\begin{tikzcd} S \arrow[r, "c^{(2)}"] \arrow[d, "pre^{(1)}"'] \arrow[dr, phantom, "="] & S \arrow[d, "post^{(1)}"] \\ Expr \arrow[r, "id^{(0)}"'] & Expr \end{tikzcd}$$

This requires only the weak equation:

$$\boxed{\mathtt{update} \bullet \mathtt{lookup} \sim_{st} id_A}$$

Method:

- $\sim_{st}$ is used inductively, by substitution and pure replacement
- until finally $\equiv$ is used for simplifying pure terms

When $Expr = B$ this corresponds to a weakest precondition semantics (here with a restricted language for conditions)

Backward semantics: an example

The given program is

$$\ell := s(\ell);\ \ell := s(\ell);\ \mathtt{return}(p(\ell))$$

translated as:

$$pg^{(2)} = \mathtt{lookup}^{(1)} \bullet s^{(0)} \bullet \mathtt{update}^{(2)} \bullet \mathtt{lookup}^{(1)} \bullet s^{(0)} \bullet \mathtt{update}^{(2)} \bullet \mathtt{lookup}^{(1)} \bullet p^{(0)}$$

$$\begin{tikzcd}
\mathbb{1} \arrow[r, "\mathtt{lookup}"] & A \arrow[r, "s"] & A \arrow[r, "\mathtt{update}"] & \mathbb{1} \arrow[r, "\mathtt{lookup}"] & A \arrow[r, "s"] & A \arrow[r, "\mathtt{update}"] \arrow[d, equal] & \mathbb{1} \arrow[r, "\mathtt{lookup}"] & A \arrow[r, "p"] \arrow[d, equal] & A \\
\mathbb{1} \arrow[r, "\mathtt{lookup}"] & A \arrow[r, "s"] & A \arrow[r, "\mathtt{update}"] \arrow[d, equal] & \mathbb{1} \arrow[r, "\mathtt{lookup}"] & A \arrow[r, "s"] \arrow[d, equal] & A \arrow[rr, "\overset{\sim}{id}"] & & A \arrow[r, "p"] & A \\
\mathbb{1} \arrow[r, "\mathtt{lookup}"] & A \arrow[r, "s"] \arrow[d, equal] & A \arrow[rr, "\overset{\sim}{id}"] & & A \arrow[r, "s"] & A \arrow[rr, "id"] & & A \arrow[r, "p"] & A \arrow[d, equal] \\
\mathbb{1} \arrow[r, "\mathtt{lookup}"] & A \arrow[rrrrrrr, "\overset{\equiv}{s}"] & & & & & & & A
\end{tikzcd}$$

Conclusion: $pg^{(2)} \sim_{st} \mathtt{lookup} \bullet s^{(0)}$. The "pre-expression" is $s(\ell)$

Outline

The language XS-IMP-EX

Syntax

Expressions:

as in XS-IMP

Commands:

$$c \ ::= \ \texttt{skip} \mid c; c \mid \ell := a \mid \texttt{throw} \mid \texttt{try}(c)\texttt{catch}(c)$$

Programs:

$$pg \ ::= \ c; \texttt{return}(e)$$

Restriction (easy to remove):

- only one exception name (thus, omitted)

Decorated logic for exceptions (only)

Monad $\boxed{T(X) = X + E}$

$$
\begin{tikzcd}
\mathcal{T}h_0 \arrow[r, "\subseteq"] & \mathcal{T}h_1 \arrow[r, "\subseteq"] & \mathcal{T}h_2 \\
(X \to Y) \arrow[r, mapsto] \arrow[d, mapsto] & (X \to Y) \arrow[r, mapsto] \arrow[d, mapsto] & (X \to Y) \arrow[d, mapsto] \\
(X \to Y) \arrow[r, mapsto] & (X \to Y) \arrow[r, mapsto] \arrow[d, mapsto, dotted] & (X \to Y) \arrow[d, mapsto] \\
 & (X \to Y+E) \arrow[r, mapsto, dotted] & (X+E \to Y+E)
\end{tikzcd}
$$

Weak equations $\boxed{f_1 \sim_{ex} f_2 : X \to Y}$

interpreted as: $\eta_X \bullet f_1 = \eta_X \bullet f_2 : X \to Y+E$

$$
\begin{tikzcd}
X \arrow[r, "\eta_X"] & X+E \arrow[r, shift left, "f_1"] \arrow[r, shift right, "f_2"'] & Y+E
\end{tikzcd}
$$

$\sim_{ex}$ satisfies replacement and pure substitution:

$$g_1 \sim_{ex} g_2 \implies f^{(0)} \bullet g_1 \bullet h \sim_{ex} f^{(0)} \bullet g_2 \bullet h$$

Operations and equations for exceptions (only)

In $\mathcal{S}et$: a set of exceptions E with (here) $\mathbb{1} \cong E$, denoted $\star \leftrightarrow \circledast$
Then $\mathcal{T}h_1$ and $\mathcal{T}h_2$ are generated from $\mathcal{T}h_0$ by
two operations:

$\mathtt{tag}^{(1)} : \mathbb{1} \to \mathbb{0}$	$\mathtt{untag}^{(2)} : \mathbb{0} \to \mathbb{1}$
$\mathtt{tag} : \mathbb{1} \to E$ $\mathtt{tag} : \star \mapsto \circledast$	$\mathtt{untag} : E \to \mathbb{1} + E$ $\mathtt{untag} : \circledast \mapsto \star$

one weak equation:

$\mathtt{tag} \bullet \mathtt{untag} \sim_{ex} id_{\mathbb{1}}$
$\mathtt{tag} \bullet \mathtt{untag} = \eta_{\mathbb{1}}$ $\star \mapsto \circledast \mapsto \star$ and $\circledast \mapsto \circledast \mapsto \star$

and decorated rules...

Decorated logic for states and exceptions

Duality is broken!

Functor $\boxed{T(D(X)) = S \times X + S \times E}$

$$\mathcal{T}h_0 \xrightarrow{\subseteq} \mathcal{T}h_1 \xrightarrow{\subseteq} \mathcal{T}h_2 \xrightarrow{\subseteq} \mathcal{T}h_3 \xrightarrow{\subseteq} \mathcal{T}h_4$$

$$\begin{tikzcd}
(X \to Y) \arrow[r, mapsto] \arrow[d, mapsto] & (X \to Y) \arrow[r, mapsto] \arrow[d, mapsto] & (X \to Y) \arrow[d, mapsto] \\
(X \to Y) \arrow[r, mapsto] & (X \to Y) \arrow[r, mapsto] \arrow[d, mapsto] & (X \to Y) \arrow[d, mapsto] \\
 & (S \times X \to S \times Y) \arrow[r, mapsto] & (S \times X \to S \times Y) \arrow[d, mapsto] \\
 & & (S \times X + S \times E \to S \times Y + S \times E)
\end{tikzcd}$$

pure *state* *exception*

Operations for states and exceptions: summary

$$
\begin{tikzcd}
\cdots & \texttt{lookup} & \texttt{update} & \texttt{tag} & \texttt{untag} \\
\mathcal{T}h_0 \arrow[r, "\subseteq"] \arrow[d] & \mathcal{T}h_1 \arrow[r, "\subseteq"] & \mathcal{T}h_2 \arrow[r, "\subseteq"] \arrow[dd] & \mathcal{T}h_3 \arrow[r, "\subseteq"] & \mathcal{T}h_4 \arrow[ddd] \\
\mathcal{S}et \arrow[drr, "S\times-"'] & & & & \\
& & \mathcal{S}et \arrow[drr, "-+SE"'] & & \\
& & & & \mathcal{S}et \\
\mathit{pure} & & \mathit{state} & & \mathit{exception}
\end{tikzcd}
$$

- $f^{(1)}$: may use the state
- $f^{(2)}$: may use and modify the state
- $f^{(3)}$: may use and modify the state,
  may raise exceptions and must propagate exceptions
- $f^{(4)}$: may use and modify the state,
  may raise exceptions and must propagate exceptions,
  may recover from exceptions

Decorated equations for states and exceptions

Weak equations

$$SX \xrightarrow{\eta_{SX}} SX+SE \underset{f_2}{\overset{f_1}{\rightrightarrows}} SY+SE \xrightarrow{\varepsilon_Y+SE} Y+SE$$

- $\boxed{f_1 \sim_{ex} f_2 : X \to Y}$

  interpreted as: $\eta_{SX} \bullet f_1 = \eta_{SX} \bullet f_2$

  $$g_1 \sim_{ex} g_2 \implies f^{(2)} \bullet g_1 \bullet h \sim_{ex} f^{(2)} \bullet g_2 \bullet h$$

- $\boxed{f_1 \sim_{st} f_2 : X \to Y}$

  interpreted as: $f_1 \bullet (\varepsilon_Y+SE) = f_2 \bullet (\varepsilon_Y+SE)$

  $$g_1 \sim_{st} g_2 \implies f \bullet g_1 \bullet h^{(0)} \sim_{st} f \bullet g_2 \bullet h^{(0)}$$

- $\boxed{f_1 \sim_{st,ex} f_2 : X \to Y}$

  interpreted as: $\eta_{SX} \bullet f_1 \bullet (\varepsilon_Y+SE) = \eta_{SX} \bullet f_2 \bullet (\varepsilon_Y+SE)$

  $$g_1 \sim_{st,ex} g_2 \implies f^{(2)} \bullet g_1 \bullet h^{(0)} \sim_{st,ex} f^{(2)} \bullet g_2 \bullet h^{(0)}$$

Equations for states and exceptions: summary

$$
\begin{tikzcd}
\mathcal{T}h_0 \arrow[r, "\subseteq"] & \mathcal{T}h_1 \arrow[r, "\subseteq"] & \mathcal{T}h_2 \arrow[r, "\subseteq"] & \mathcal{T}h_3 \arrow[r, "\subseteq"] & \mathcal{T}h_4 \\
\equiv \arrow[r, dash] & \equiv \arrow[r, dash] \arrow[ddr, dash] & \equiv \arrow[r, dash] & \equiv \arrow[r, dash] \arrow[dr, dash] & \equiv \\
 & & & & \sim_{ex} \\
 & & \sim_{st} \arrow[r, dash] & \sim_{st} \arrow[r, dash] \arrow[dr, dash] & \sim_{st} \\
 & & & & \sim_{st,ex} \\
\mathit{pure} & & \mathit{state} & & \mathit{exception}
\end{tikzcd}
$$

Translation

Expressions: $\boxed{e \mapsto e^{(1)} : \mathbb{1} \to Expr}$
as for XS-IMP

Commands: $\boxed{c \mapsto c^{(3)} : \mathbb{1} \to \mathbb{1}}$ *(really (3), not (4))*

- $\mathtt{skip}$, $c_1; c_2$, $\ell := a$:
  as for XS-IMP
- $\mathtt{throw} \mapsto \mathtt{tag}^{(3)} \bullet [\,]^{(0)}_{\mathbb{1}}$
  pretends that the exception has type $\mathbb{1}$, instead of $\mathbb{0}$
- $\mathtt{try}(c_1)\mathtt{catch}(c_2) \mapsto (\downarrow (c_1 \bullet [id_{\mathbb{1}} | \mathtt{untag}^{(4)} \bullet c_2]\,)^{(3)}$
  (next slide)

Programs: $\boxed{pg \mapsto pg^{(3)} : \mathbb{1} \to Expr}$ *(really (3), not (4))*
as for XS-IMP

Translation of try-catch

$$\texttt{try}(c_1)\texttt{catch}(c_2) \mapsto (\downarrow(\, c_1 \bullet [\, id_{\mathbb{1}} \mid \texttt{untag}^{(4)} \bullet c_2 \,]\,)^{(3)}$$

Uses: the decorated coproduct $\mathbb{1} = \mathbb{1} + \mathbb{0}$

$$\begin{tikzcd}
 & \mathbb{1} \arrow[d, "id"'] \arrow[drr, "id"] \arrow[drr, phantom, "\sim_{ex}"' near start] & & \\
\mathbb{1} \arrow[r, "c_1"] & \mathbb{1} \arrow[rr, "{[id|\texttt{untag}\bullet c_2]}" description] & & \mathbb{1} \\
 & \mathbb{0} \arrow[u, "\langle\,\rangle"] \arrow[r, "\texttt{untag}"'] \arrow[ur, phantom, "\equiv"] & \mathbb{1} \arrow[ur, "c_2"'] &
\end{tikzcd}$$

and the "downcast" operator $\downarrow$

$$(\downarrow(f^{(4)}))^{(3)} \text{ is such that } f \sim_{ex} \downarrow f$$

- $\downarrow f$ is the same as f on non-exceptional arguments
- $\downarrow f$ propagates exceptions while f may recover from exceptions

Rules for $\downarrow$ include: $\downarrow(f_1) \equiv \downarrow(f_2) \iff f_1 \sim_{ex} f_2$

Translation of XS-IMP-EX: summary

$$
\begin{tikzcd}
\dots & \ell \arrow[d, squiggly] & \ell := \arrow[d, squiggly] & \mathtt{throw} \arrow[d, squiggly] & \mathtt{try\text{-}catch} \arrow[dl, squiggly] \arrow[d, dotted] \\
\mathcal{T}h_0 \arrow[r, "\subseteq"] \arrow[d] & \mathcal{T}h_1 \arrow[r, "\subseteq"] & \mathcal{T}h_2 \arrow[r, "\subseteq"] \arrow[dd] & \mathcal{T}h_3 \arrow[r, "\subseteq"] & \mathcal{T}h_4 \arrow[l, dotted, bend left, "\downarrow"] \arrow[ddd] \\
\mathcal{S}et \arrow[drr, "S\times -"'] & & & & \\
& & \mathcal{S}et \arrow[drr, "-+SE"'] & & \\
& & & & \mathcal{S}et \\
\textit{pure} & & \textit{state} & & \textit{exception}
\end{tikzcd}
$$

Backward semantics

Predicate transformer semantics [Claude Marché, MPRI 2012]

Hoare triples: $\{P\}\, c\, \{Q|R\}$ is valid if:
if c is executed in a state satisfying P then:
– if c terminates normally in a state s' then s' satisfies Q
– if c terminates abruptly in a state s' then s' satisfies R

This means that $\boxed{P \sim_{st,ex} c \bullet [Q \mid \mathtt{untag} \bullet R]}$

$$\begin{tikzcd}
 & \mathbb{1} \arrow[d, "id"'] \arrow[drr, "Q^{(1)}"] \arrow[dr, phantom, "\sim_{st,ex}"] & & \\
\mathbb{1} \arrow[r, "c^{(3)}"] & \mathbb{1} \arrow[rr, "{[Q|\mathtt{untag}\bullet R]}"] \arrow[dr, phantom, "\equiv"] & & B \\
 & \mathbb{0} \arrow[u, "{[\,]}"] \arrow[r, "\mathtt{untag}^{(4)}"'] & \mathbb{1} \arrow[ur, "R^{(1)}"'] &
\end{tikzcd}$$

Outline

The language IMP-EX: syntax, revisited

Expressions:

$$\begin{aligned} a &::= 0 \mid 1 \mid -1 \mid 2 \mid -2 \mid \ldots \mid \ell \mid s(a) \mid p(a) \\ b &::= \mathtt{true} \mid \mathtt{false} \mid \neg b \mid a = 0 \mid a > 0 \\ e &::= a \mid b \end{aligned}$$

Commands:

$$c \ ::= \ \begin{cases} \mathtt{skip} \mid c; c \mid \ell := a \mid \\ \mathtt{throw} \mid \mathtt{try}(c)\mathtt{catch}(c) \mid \\ \mathtt{if}(b)\mathtt{then}(c)\mathtt{else}(c) \mid \mathtt{repeat}(c) \end{cases}$$

Programs:

$$pg \ ::= \ c; \mathtt{return}(e)$$

- As before: only one location, no binary operation on expressions, only one exception name
- In addition: $\mathtt{repeat}(c)$ “instead of” $\mathtt{while}(b)\mathtt{do}(c)$

Decorated logic for non-termination

Partiality

$$\begin{tikzcd}
\mathcal{T}h_0 \arrow[r, "\subseteq"] \arrow[d] & \mathcal{T}h_1 \arrow[d] & \\
\mathcal{S}et \arrow[r, "\subseteq"] \arrow[drr, dashed, "+\mathbb{1}"'] & \mathcal{P}art \arrow[r, dashed] & \overline{im}(+\mathbb{1}) \arrow[d, dashed] \\
& & \mathcal{S}et
\end{tikzcd}
\qquad
\begin{tikzcd}
(X \to Y) \arrow[r, mapsto] \arrow[d, mapsto] & (X \to Y) \arrow[d, mapsto] \\
(X \to Y) \arrow[r, mapsto] & (X \rightharpoonup Y)
\end{tikzcd}$$

Weak equations are inequations $\boxed{f_1 \succcurlyeq f_2 : X \to Y}$

interpreted as: $f_1 \geq f_2 : X \rightharpoonup Y$ (as partial functions)

$\mathcal{P}art$ with $\geq$ is a 2-category

$\succcurlyeq$ satisfies replacement and substitution:

$$g_1 \succcurlyeq g_2 \implies f \bullet g_1 \bullet h \succcurlyeq f \bullet g_2 \bullet h$$

Operations and equations for non-termination

$\mathcal{T}h_1$ is generated from $\mathcal{T}h_0$ by
one operation constructor:

$\mathtt{loop}(c)^{(1)} : X \to X$ for each $c^{(1)} : X \to X$
$\mathtt{loop}(c) : X \rightharpoonup X$ is the least fixed point of $f \mapsto c \bullet f$

one strong equation:

$$\boxed{\mathtt{loop}(c) \equiv c \bullet \mathtt{loop}(c)}$$

and decorated rules, including:

$$\boxed{f \equiv c \bullet f \implies f \succcurlyeq \mathtt{loop}(c)}$$

A weak congruence

The "weakest" congruence for states, exceptions and non-termination is $\preccurlyeq_{st,ex}$. For instance:

$$\boxed{f \preccurlyeq_{st,ex} u^{(0)} : X \to Y}$$

is a concise way to express the following:

$f : S\times X+S\times E \rightharpoonup S\times Y+S\times E$ and $u : X \to Y$ are such that:

if $f(s,x)$ is defined, then it returns $(s',u(x))$ for some s'.

$$\begin{tikzcd}
S\,X \arrow[r, "\varepsilon_X"] \arrow[d, "\eta_{S\,X}"'] & S\,X+S\,E \arrow[r, "f"] \arrow[d, phantom, "\curlyeqsucc"] & S\,Y+S\,E \arrow[r, "\varepsilon_Y+S\,E"] & Y+S\,E \\
X \arrow[rrr, "u"'] & & & Y \arrow[u, "\eta_Y"']
\end{tikzcd}$$

This is the kind of relation required
between a "program" f and its "result" u

Translation

Translation is obvious:

- $\mathtt{repeat}(c) \mapsto \mathtt{loop}(c)$

Example $\boxed{\mathtt{repeat}(\mathtt{throw}) \equiv \mathtt{throw}}$

$\mathtt{repeat}(\mathtt{throw})$ is translated as $r = \mathtt{loop}(\mathtt{tag} \bullet [\,]_{\mathbb{1}}\,)$

$$
\begin{tikzcd}
\mathbb{1} \arrow[rrr, "r"] \arrow[d, equal] \arrow[drr, phantom, "\equiv"] & & & \mathbb{1} \arrow[d, equal] \\
\mathbb{1} \arrow[r, "\mathtt{tag}"] & \mathbb{0} \arrow[r, "{[\,]}" description] \arrow[d, equal] \arrow[dr, phantom, "\equiv"] & \mathbb{1} \arrow[r, "r"] & \mathbb{1} \arrow[d, equal] \\
\mathbb{1} \arrow[r, "\mathtt{tag}"] & \mathbb{0} \arrow[rr, "{[\,]}" description] & & \mathbb{1}
\end{tikzcd}
$$

because r (like all commands) propagates exceptions

Decorated logic for conditions

Weak equations are conditional equations

$$\boxed{f_1 \sim f_2 : X \to Y \text{ if } b}$$

where $\sim$ is any of the previous (strong or weak) congruence and b is a boolean expression.

For replacement, conditional $\sim$ has the same properties as $\sim$.

$\mathcal{T}h_1$ is generated from $\mathcal{T}h_0$ by
two operation constructors ("conditional non-determinism"):

$\mathtt{choose}(c_1, c_2)^{(1)} : X \to Y$ for each $c_1^{(1)}, c_2^{(1)} : X \to Y$ $\downarrow_b (c)^{(0)} : X \to Y$ for each $c^{(1)} : X \to Y$
$[c_1 \vert c_2] : X + X \to Y$ $\downarrow_b ([c_1 \vert c_2]) = b \bullet [c_1 \vert c_2] : X \to Y$

Decorated logic for IMP-EX

Combine the decorated logics for:

- states
- exceptions
- non-termination
- and "conditional non-determinism"

by composing the corresponding functors
and extending the corresponding weak congruences

Outline

Conclusion

Remark.
Effects as functors, with their weak congruences, can be seen as a kind of generalization of 2-categories, with decorated categorical notions as a generalization of lax categorical notions.

To do...

- "work in progress":
  – which is the best order for composing the effects?
- Define `while` loops by:
  $\mathtt{while}(b)\mathtt{do}(c) =$
  $\mathtt{try}\,(\,\mathtt{repeat}(\mathtt{if}(b)\mathtt{then}(c)\mathtt{else}(\mathtt{throw})\,)\,)\,\mathtt{catch}(\mathtt{skip})$
  Prove that indeed such a `while` loop is the least fixed point of $f \mapsto \mathtt{if}(b)\mathtt{then}(c; f)\mathtt{else}(\mathtt{skip})$
- Complete the implementation in Coq
- Towards richer languages (C, C++, Java,...)