Relative Hilbert-Post completeness for exceptions

Dominique Duval
with J.-G. Dumas, B. Ekici, D. Pous, J.-C. Reynaud

LJK University of Grenoble-Alpes *and* ENS Lyon

November 12., 2015 — MACIS 2015, Berlin

This talk is about a completeness result

Theorem.

The decorated theory for exceptions
is relatively Hilbert-Post complete.

In the paper:

- a detailed proof of this theorem
- and the key features for its verification in Coq

In this talk:

- the framework for this theorem
- and its meaning

Outline

Framework

The general issue:

semantics
of programming languages

More precisely:

equational semantics
of programming languages with computational effects

Work in progress: IMPEX

IMPEX is a basic imperative language with exceptions:

$$c \quad ::= \quad \text{skip} \mid x := a \mid c; c \mid \text{if } b \text{ then } c \text{ else } c \mid \text{while } b \text{ do } c$$
$$\mid \text{throw } e \mid \text{try } c \text{ catch } e \Rightarrow c$$

What is the interpretation of a command?

Work in progress: IMPEX

IMPEX is a basic imperative language with exceptions:

$$c \quad ::= \quad \mathrm{skip} \mid x := a \mid c; c \mid \mathrm{if}\ b\ \mathrm{then}\ c\ \mathrm{else}\ c \mid \mathrm{while}\ b\ \mathrm{do}\ c$$

$$\mid \mathrm{throw}\ e \mid \mathrm{try}\ c\ \mathrm{catch}\ e \Rightarrow c$$

What is the interpretation of a command?

- $c : 1 \to 1$, because c has no argument and no result?

Work in progress: IMPEX

IMPEX is a basic imperative language with exceptions:

$$c \quad ::= \quad \mathrm{skip} \mid x := a \mid c; c \mid \mathrm{if}\ b\ \mathrm{then}\ c\ \mathrm{else}\ c \mid \mathrm{while}\ b\ \mathrm{do}\ c$$

$$\mid \mathrm{throw}\ e \mid \mathrm{try}\ c\ \mathrm{catch}\ e \Rightarrow c$$

What is the interpretation of a command?

- $c : 1 \to 1$, because c has no argument and no result?
- $c : S \to S$, because c may use and modify the state?

Work in progress: IMPEX

IMPEX is a basic imperative language with exceptions:

$$c \quad ::= \quad \text{skip} \mid x := a \mid c; c \mid \text{if } b \text{ then } c \text{ else } c \mid \text{while } b \text{ do } c$$

$$\mid \text{throw } e \mid \text{try } c \text{ catch } e \Rightarrow c$$

What is the interpretation of a command?

- $c : 1 \to 1$, because c has no argument and no result?
- $c : S \to S$, because c may use and modify the state?
- $c : S \rightharpoonup S$, because c may not terminate?

Work in progress: IMPEX

IMPEX is a basic imperative language with exceptions:

$$c \quad ::= \quad \mathrm{skip} \mid x := a \mid c; c \mid \mathrm{if}\ b\ \mathrm{then}\ c\ \mathrm{else}\ c \mid \mathrm{while}\ b\ \mathrm{do}\ c$$

$$\mid \mathrm{throw}\ e \mid \mathrm{try}\ c\ \mathrm{catch}\ e \Rightarrow c$$

What is the interpretation of a command?

- $c : \mathbb{1} \to \mathbb{1}$, because c has no argument and no result?
- $c : S \to S$, because c may use and modify the state?
- $c : S \rightharpoonup S$, because c may not terminate?
- $c : S \rightharpoonup S \times (\mathbb{1} + E)$, because c may raise an exception?

Work in progress: IMPEX

IMPEX is a basic imperative language with exceptions:

$$c \quad ::= \quad \mathrm{skip} \mid x := a \mid c; c \mid \mathrm{if}\ b\ \mathrm{then}\ c\ \mathrm{else}\ c \mid \mathrm{while}\ b\ \mathrm{do}\ c$$

$$\mid \mathrm{throw}\ e \mid \mathrm{try}\ c\ \mathrm{catch}\ e \Rightarrow c$$

What is the interpretation of a command?

- $c : 1 \to 1$, because c has no argument and no result?
- $c : S \to S$, because c may use and modify the state?
- $c : S \rightharpoonup S$, because c may not terminate?
- $c : S \rightharpoonup S \times (1 + E)$, because c may raise an exception?
- $c : S \times (1 + E) \rightharpoonup S \times (1 + E)$, for sequences ";" and for the `catch` part of the `try-catch` block?

Three effects for IMPEX

- State. $f : X \to Y$ stands for $f : S \times X \to S \times Y$
- Partiality. $f : X \to Y$ stands for $f : X \rightharpoonup Y$
- Exceptions. $f : X \to Y$ stands for $f : X + E \to Y + E$

Goal.
Prove equivalence of commands in a logic where $c : 1 \to 1$ (effects are "hidden", as in the syntax).

For instance, prove that:

if b is "pure" then

$$(x := a\,; x := b) \;\equiv\; (x := b)$$

or that:

$$\mathrm{while}\ b\ \mathrm{do}\ c \;\equiv\; \mathrm{try}\ (\mathrm{repeat}\ (\mathrm{if}\ b\ \mathrm{then}\ c\ \mathrm{else}\ \mathrm{throw}\ e))\ \mathrm{catch}\ e \Rightarrow \mathrm{skip}$$

where $\mathrm{repeat}\ c$ is $\mathrm{while\ true\ do}\ c$.

Goal (for IMPEX)

Prove equivalence of commands in a logic where $c : 1 \to 1$
(effects are "hidden", as in the syntax)
and implement this logic in Coq

Method.

1. Design a decorated logic for each effect.
2. Combine the three logics.

Here: a decorated logic for the exceptions effect:

A term $f : X \to Y$ is interpreted as
a function $[[f]] : [[X]] + E \to [[Y]] + E$
where E is the set of exception names.

(notation: now, [[ ]] is omitted)

Outline

Decorations and conversions

The decorated logic for exceptions is built from types, terms and equations, with

- three kinds of terms:
  - a pure term $f^{(0)} \colon X \to Y$ is interpreted as $f \colon X \to Y$
  - a propagator $f^{(1)} \colon X \to Y$ as $f \colon X \to Y + E$
  - a catcher $f^{(2)} \colon X \to Y$ as $f \colon X + E \to Y + E$

  with conversions $\dfrac{f^{(0)}}{f^{(1)}}$, $\dfrac{f^{(1)}}{f^{(2)}}$
- and two kinds of equations:
  - a strong equation $f^{(2)} \equiv g^{(2)} \colon X \to Y$ is interpreted as $f = g \colon X + E \to Y + E$
  - a weak equation $f^{(2)} \sim g^{(2)} \colon X \to Y$ is interpreted as $f \circ inl_X = g \circ inl_X \colon X \to Y + E$

  with conversions $\dfrac{f \equiv g}{f \sim g}$, $\dfrac{f^{(1)} \sim g^{(1)}}{f \equiv g}$

A decorated logic for exceptions

The logic L_E has no type of exceptions
It is generated by any pure signature and
for each exception name e (with type of parameters P_e):

- a propagator $\mathtt{tag}_e^{(1)} : P_e \to 0$
  interpreted as $\mathtt{tag}_e : P_e \to E$
  denoted $a \mapsto \boxed{a}_e$
- and a catcher $\mathtt{untag}_e^{(2)} : 0 \to P_e$
  interpreted as $\mathtt{untag}_e : E \to P_e + E$

related by weak equations:

- $\mathtt{untag}_e \circ \mathtt{tag}_e \sim id_{P_e}$
- $\mathtt{untag}_e \circ \mathtt{tag}_{e'} \sim [\,]_{P_e} \circ \mathtt{tag}_{e'}$ when $e' \neq e$

which mean that $\mathtt{untag}_e : E \to P_e + E$ satisfies:

$$\begin{cases} \boxed{a}_e \mapsto a \\ \boxed{a}_{e'} \mapsto \boxed{a}_{e'} \text{ when } e' \neq e \end{cases}$$

A conversion in the opposite direction

The conversion $\dfrac{f^{(1)}}{f^{(2)}}$ means that

each function $f : X \to Y + E$ can be extended as $f' : X + E \to Y + E$, by propagating exceptions.

In the opposite direction

each function $g : X + E \to Y + E$ can be restricted as $g \circ inl : X \to Y + E$.

This is expressed in the decorated logic by the downcast construction:

$$\frac{f^{(2)} : X \to Y}{(\downarrow f)^{(1)} : X \to Y} \quad \text{with} \quad f^{(2)} \sim (\downarrow f)^{(1)}$$

throw and try-catch

The core operations $\mathtt{tag}_e^{(1)} : P_e \to 0$ and $\mathtt{untag}_e^{(2)} : 0 \to P_e$ are used for expressing the usual constructs:

- throw:
  for each Y, $\mathtt{throw}_{e,Y}^{(1)} : P_e \to Y$ is

  $$\mathtt{throw}_{e,Y} \;=\; [\,]_Y \circ \mathtt{tag}_e$$

  it raises the exception e and pretends that it has type Y.
- try-catch:
  for each $f^{(1)} : X \to Y$ and $g^{(1)} : P_e \to Y$
  $(\mathtt{try}\ f\ \mathtt{catch}\ e \Rightarrow g)^{(1)} : X \to Y$ is

  $$\mathtt{try}\ f\ \mathtt{catch}\ e \Rightarrow g \;=\downarrow([id_Y \mid g \circ \mathtt{untag}] \circ f)$$

  it is also a propagator: the catcher $\mathtt{untag}_e^{(2)}$ is encapsulated

Outline

The framework

Decorated logic for exceptions

Relative Hilbert-Post completeness

Conclusion and references

About completeness

Fact. The decorated logic for exceptions is sound with respect to its interpretation:

Provable $\Longrightarrow$ Valid

Question. Is it complete?

For which notion of completeness?

- Semantic completeness?
  Valid $\Longrightarrow$ Provable
- Syntactic completeness?
  Every added unprovable sentence introduces an inconsistency, where inconsistency means:
  - either negation inconsistency:
    there is a sentence φ such that φ and $\neg\varphi$ are provable
  - or Hilbert-Post inconsistency:
    every sentence is provable

Here. Relative Hilbert-Post completeness

(Absolute) Hilbert-Post completeness

In a given logic:

- a theory is a set of sentences which is deductively closed
- a theory T is consistent if it does not contain all sentences
- a theory T is H-P complete if:
  - T is consistent and
  - any sentence added to T generates an inconsistent theory

So, H-P completeness is maximal consistency

Example. (H-P completeness is *very strong*)
Signature: $N, \ 0 : 1 \to N, \ s : N \to N$

- The theory generated from the axiom $s \circ s \equiv s$
  is not H-P complete
- The theory generated from $s \circ s \equiv s$ and $s \circ 0 \equiv 0$
  is H-P complete: it is made of all equations but $s \equiv id_N$

Relative Hilbert-Post completeness

In a given logic L:

- a theory T is H-P complete if:
  - T is consistent and
  - any sentence added to T generates an inconsistent theory

In a given logic L extending a sublogic L_0:

- a theory T of L is relatively H-P complete wrt L_0 if:
  - T is consistent and
  - for any sentence e of L there is a set E_0 of sentences of L_0 which is T-equivalent to e

Theorem.
In the logic L_E, under suitable assumptions [...],
the decorated theory for exceptions
is relatively H-P complete wrt the pure sublogic

Outline

Conclusion

See the paper for:

- the implementation of the logic for exceptions in Coq
- a proof of the Theorem, checked with Coq

To improve:

- weaken the assumptions in the Theorem

A question:

- Relative H-P completeness seems more interesting in practice than absolute H-P completeness: why?

Work in progress: IMPEX

- exceptions: this talk
- states: essentially dual to exceptions
- non-termination: well-known(?)
- combination of the three logics...

Some references

- A. Tarski. On some fundamental concepts in mathematics (1956).
- E. Moggi. Notions of Computation and Monads (1991).
- G.D. Plotkin, J. Power. Notions of Computation Determine Monads (2002).
- M. Pretnar. The Logic and Handling of Algebraic Effects (2010).
- C. Domínguez, D. Duval. Diagrammatic logic applied to a parameterisation process (2010).
- J.-G. Dumas, D. Duval, L. Fousse, J.-C. Reynaud. A duality between exceptions and states (2012).
- A. Bauer, M. Pretnar. Programming with algebraic effects and handlers (2015).