Decorated proofs for computational effects: States

Jean-Guillaume Dumas, Dominique Duval, Laurent Fousse, Jean-Claude Reynaud

LJK, University of Grenoble

April 1., 2012 – ACCAT 2012 – Tallinn

Outline

From computer algebra to effects

About the history of the authors:

- Computer algebra: exact computations on large integers, matrices, polynomials, field extensions,...
- Sophisticated programmation in several kinds of languages: C, C++, Axiom,...
- Questions about the languages: semantics of computational effects? (e.g., states, exceptions,...)

Effects and monads

Breaking a taboo:

$$\text{effect} \neq \text{monad}$$

Effects and monads

Breaking a taboo:

$$\text{effect} \neq \text{monad}$$

[Moggi'91]: When there is an effect:

1. a term $f : X \to Y$ should not always be interpreted as a function $[[f]] : [[X]] \to [[Y]]$
2. it should often be interpreted as a function $[[f]] : [[X]] \to T[[Y]]$ for some monad T

Effects and monads

Breaking a taboo:

$$\text{effect} \neq \text{monad}$$

[Moggi'91]: When there is an effect:

1. a term $f : X \to Y$ should not always be interpreted as a function $[[f]] : [[X]] \to [[Y]]$
2. it should often be interpreted as a function $[[f]] : [[X]] \to T[[Y]]$ for some monad T

[Plotkin & Power 2002]: The operations and equations associated with the effect are described by a Lawvere theory.

Effects and monads

Breaking a taboo:

$$\text{effect} \neq \text{monad}$$

[Moggi'91]: When there is an effect:

1. a term $f : X \to Y$ should not always be interpreted as a function $[[f]] : [[X]] \to [[Y]]$
2. it should often be interpreted as a function $[[f]] : [[X]] \to T[[Y]]$ for some monad T

[Plotkin & Power 2002]: The operations and equations associated with the effect are described by a Lawvere theory.

Example. In an imperative language

$$T[[Y]] = (S \times [[Y]])^S$$

Effects and monads

Breaking a taboo:

$$\text{effect} \neq \text{monad}$$

[Moggi'91]: When there is an effect:

1. a term $f : X \to Y$ should not always be interpreted as a function $[[f]] : [[X]] \to [[Y]]$
2. it should often be interpreted as a function $[[f]] : [[X]] \to T[[Y]]$ for some monad T

[Plotkin & Power 2002]: The operations and equations associated with the effect are described by a Lawvere theory.

Example. In an imperative language

$$T[[Y]] = (S \times [[Y]])^S$$

We agree with (1), not always with (2).
And we get operations and equations in a different way.

What is an effect?

Informally:

An effect is an apparent lack of soundness.

What is an effect?

Informally:

An effect is an apparent lack of soundness.

A lack of soundness:

$$\texttt{syntax} \;\text{– – – –}\; \times \;\text{– – – –}\; \texttt{semantics}$$

which can be "repaired":

$$\begin{tikzcd} & \bullet \arrow[dl] \arrow[dr] & \\ \texttt{syntax} & & \texttt{semantics} \end{tikzcd}$$

What is an effect?

Informally:

An effect is an apparent lack of soundness.

A lack of soundness:

$$\mathtt{syntax} \; - - - - \times - - - - \; \mathtt{semantics}$$

which can be “repaired”:

$$\begin{tikzcd} & \bullet \arrow[dl] \arrow[dr] & \\ \mathtt{syntax} & & \mathtt{semantics} \end{tikzcd}$$

Formally: [Domínguez&Duval MSCS’10]

Outline

A property of imperative languages

The annihilation lookup-update (ALU) property:

$$\boxed{X := X \text{ does not modify the state}}$$

A property of imperative languages

The annihilation lookup-update (ALU) property:

$$\boxed{X := X \text{ does not modify the state}}$$

Proof.

Let n be the value of X in the current state.

- First "X" (on the right) is evaluated as n.
- Then "$X :=$" (on the left) puts the value of X to n, without modifying the value of other locations.

Hence the state is not modified. $\square$

Towards a formalization: a specification for states

Locations (or identifiers, or variables) $X, Y, \ldots$.
The unit (or void, or singleton) type $\mathbb{1}$, with $\langle\,\rangle_A : A \to \mathbb{1}$ for each A.

Towards a formalization: a specification for states

Locations (or identifiers, or variables) $X, Y, \dots$.
The unit (or void, or singleton) type $\mathbb{1}$, with $\langle\,\rangle_A : A \to \mathbb{1}$ for each A.

For each X, a type V_X for values, two operations:

$$\begin{array}{ll} \ell_X : \mathbb{1} \to V_X & \text{(lookup)} \\ u_X : V_X \to \mathbb{1} & \text{(update)} \end{array}$$

and equations:

$$\begin{array}{ll} \ell_X \circ u_X \equiv id & \\ \ell_Y \circ u_X \equiv \ell_Y \circ \langle\,\rangle & \text{when } Y \neq X \end{array}$$

formalizing the intended semantics:

- ℓ_X returns the value of X in the current state
- $u_X(n)$ modifies the current state: the value of X becomes n, and the value of Y is not modified, for every $Y \neq X$

A property of imperative languages: proof # 1

Let Σ be the specification made of $\ell_X : \mathbb{1} \to V_X$ and $u_X : V_X \to \mathbb{1}$ such that $\ell_X \circ u_X \equiv id$ and $\ell_Y \circ u_X \equiv \ell_Y \circ \langle\,\rangle$ when $Y \neq X$.

Then Σ *satisfies the annihilation lookup-update (ALU) property:*

$$\boxed{u_X \circ \ell_X \equiv id}$$

A property of imperative languages: proof # 1

Let Σ be the specification made of $\ell_X : \mathbb{1} \to V_X$ and $u_X : V_X \to \mathbb{1}$ such that $\ell_X \circ u_X \equiv id$ and $\ell_Y \circ u_X \equiv \ell_Y \circ \langle\,\rangle$ when $Y \neq X$.

Then Σ *satisfies the annihilation lookup-update (ALU) property:*

$$\boxed{u_X \circ \ell_X \equiv id}$$

Proof. By observation: prove that $\ell_Y \circ u_X \circ \ell_X \equiv \ell_Y$ for each Y.
When $Y = X$:

$$(\text{subst})\ \frac{\ell_X \circ u_X \equiv id}{\ell_X \circ u_X \circ \ell_X \equiv \ell_X}$$

A property of imperative languages: proof # 1

Let Σ be the specification made of $\ell_X : \mathbb{1} \to V_X$ and $u_X : V_X \to \mathbb{1}$ such that $\ell_X \circ u_X \equiv id$ and $\ell_Y \circ u_X \equiv \ell_Y \circ \langle\,\rangle$ when $Y \neq X$.

Then Σ *satisfies the annihilation lookup-update (ALU) property:*

$$\boxed{u_X \circ \ell_X \equiv id}$$

Proof. By observation: prove that $\ell_Y \circ u_X \circ \ell_X \equiv \ell_Y$ for each Y.
When $Y = X$:

$$(\text{subst})\ \frac{\ell_X \circ u_X \equiv id}{\ell_X \circ u_X \circ \ell_X \equiv \ell_X}$$

When $Y \neq X$:

$$(\text{trans})\ \frac{(\text{subst})\ \dfrac{\ell_Y \circ u_X \equiv \ell_Y \circ \langle\,\rangle}{\ell_Y \circ u_X \circ \ell_X \equiv \ell_Y \circ \langle\,\rangle \circ \ell_X} \qquad (\text{repl})\ \dfrac{(\text{unit})\ \dfrac{}{\langle\,\rangle \circ \ell_X \equiv id}}{\ell_Y \circ \langle\,\rangle \circ \ell_X \equiv \ell_Y}}{\ell_Y \circ u_X \circ \ell_X \equiv \ell_Y}$$

A property of imperative languages: proof # 1

Let Σ be the specification made of $\ell_X : \mathbb{1} \to V_X$ and $u_X : V_X \to \mathbb{1}$ such that $\ell_X \circ u_X \equiv id$ and $\ell_Y \circ u_X \equiv \ell_Y \circ \langle\,\rangle$ when $Y \neq X$.

Then Σ *satisfies the annihilation lookup-update (ALU) property:*

$$\boxed{u_X \circ \ell_X \equiv id}$$

Proof. By observation: prove that $\ell_Y \circ u_X \circ \ell_X \equiv \ell_Y$ for each Y.
When $Y = X$:

$$\text{(subst)}\ \frac{\ell_X \circ u_X \equiv id}{\ell_X \circ u_X \circ \ell_X \equiv \ell_X}$$

When $Y \neq X$:

$$\text{(trans)}\ \dfrac{\text{(subst)}\ \dfrac{\ell_Y \circ u_X \equiv \ell_Y \circ \langle\,\rangle}{\ell_Y \circ u_X \circ \ell_X \equiv \ell_Y \circ \langle\,\rangle \circ \ell_X} \qquad \text{(repl)}\ \dfrac{\text{(unit)}\ \dfrac{}{\langle\,\rangle \circ \ell_X \equiv id}}{\ell_Y \circ \langle\,\rangle \circ \ell_X \equiv \ell_Y}}{\ell_Y \circ u_X \circ \ell_X \equiv \ell_Y}$$

Hence the state is not modified. $\square$

A property of imperative languages, proof # 2

The annihilation lookup-update (ALU) property:

$$\boxed{u_X \circ \ell_X \equiv id}$$

A property of imperative languages, proof # 2

The annihilation lookup-update (ALU) property:

$$\boxed{u_X \circ \ell_X \equiv id}$$

Another proof.
The (unit) rule states that id is the unique $f : \mathbb{1} \to \mathbb{1}$.

$$\text{(unit)}\ \frac{u_X \circ \ell_X : \mathbb{1} \to \mathbb{1}}{u_X \circ \ell_X \equiv id}$$

$\square$

A property of imperative languages, proof # 2

The annihilation lookup-update (ALU) property:

$$\boxed{u_X \circ \ell_X \equiv id}$$

Another proof.
The (unit) rule states that id is the unique $f : \mathbb{1} \to \mathbb{1}$.

$$\text{(unit)}\ \frac{u_X \circ \ell_X : \mathbb{1} \to \mathbb{1}}{u_X \circ \ell_X \equiv id}$$

□

BUT in the same way, we could prove for all Y:

$$\text{(unit)}\ \frac{u_X \circ \ell_Y : \mathbb{1} \to \mathbb{1}}{u_X \circ \ell_Y \equiv id}$$

which obviously is FALSE!

Questions

Two proofs of (ALU). Proof #1 is right, proof #2 is wrong.

WHY?

Questions

Two proofs of (ALU). Proof #1 is right, proof #2 is wrong.

WHY?

The (unit) rule should state that *id* is the unique $f : \mathbb{1} \to \mathbb{1}$
under the assumption that f cannot modify the state,
and it should be impossible to apply this rule to $u_X \circ \ell_Y$.

How can we formalize this fact?

Questions

Two proofs of (ALU). Proof #1 is right, proof #2 is wrong.

WHY?

The (unit) rule should state that *id* is the unique $f : \mathbb{1} \to \mathbb{1}$
under the assumption that f cannot modify the state,
and it should be impossible to apply this rule to $u_X \circ \ell_Y$.

How can we formalize this fact?

By decorating terms and equations.

Decorations: terms and equations

Terms are classified:

- $f^{(0)}$: f is pure if it cannot use nor modify the state.
- $f^{(1)}$: f is an accessor if it can use the state, not modify it.
- $f^{(2)}$: f is a modifier if it can use and modify the state.

Hierarchy rules: $\dfrac{f^{(0)}}{f^{(1)}}$, $\dfrac{f^{(1)}}{f^{(2)}}$.

Decorations: terms and equations

Terms are classified:

- $f^{(0)}$: f is pure if it cannot use nor modify the state.
- $f^{(1)}$: f is an accessor if it can use the state, not modify it.
- $f^{(2)}$: f is a modifier if it can use and modify the state.

Hierarchy rules: $\dfrac{f^{(0)}}{f^{(1)}}$, $\dfrac{f^{(1)}}{f^{(2)}}$.

Equations are classified:

- $f \equiv g$: strong equation: f and g return the same value and they have the same effect on the state.
- $f \sim g$: weak equation: f and g return the same value but they may have different effects on the state.

Hierarchy rule: $\dfrac{f \equiv g}{f \sim g}$.

Decorated rules

The rules of the logic are also decorated, for instance:

$$\text{(unit)} \quad \frac{f : \mathbb{1} \to \mathbb{1}}{f \sim id}$$

Decorated rules

The rules of the logic are also decorated, for instance:

$$(\text{unit}) \quad \frac{f : \mathbb{1} \to \mathbb{1}}{f \sim id}$$

There are new rules (which become trivial without decorations):

$$(1\text{-}\sim\text{-to-}\equiv) \quad \frac{f^{(1)} \quad g^{(1)} \quad f \sim g}{f \equiv g}$$

Decorated rules

The rules of the logic are also decorated, for instance:

$$\text{(unit)} \quad \frac{f : \mathbb{1} \to \mathbb{1}}{f \sim id}$$

There are new rules (which become trivial without decorations):

$$(1\text{-}\sim\text{-to-}\equiv) \quad \frac{f^{(1)} \quad g^{(1)} \quad f \sim g}{f \equiv g}$$

Hence there are new derived rules, like:

$$\text{(1-unit)} \quad \frac{f^{(1)} : \mathbb{1} \to \mathbb{1}}{f \equiv id}$$

Proof #2 is wrong: it cannot be properly decorated

Proof #2 of (ALU) can be decorated as follows:

$$\text{(unit)} \quad \frac{u_X \circ \ell_X : \mathbb{1} \to \mathbb{1}}{u_X \circ \ell_X \sim id}$$

which does not entail $u_X \circ \ell_X \equiv id$.

Proof #2 is wrong: it cannot be properly decorated

Proof #2 of (ALU) can be decorated as follows:

$$\text{(unit)} \quad \frac{u_X \circ \ell_X : \mathbb{1} \to \mathbb{1}}{u_X \circ \ell_X \sim id}$$

which does not entail $u_X \circ \ell_X \equiv id$.

In fact for each Y there is a proof:

$$\text{(unit)} \quad \frac{u_X \circ \ell_Y : \mathbb{1} \to \mathbb{1}}{u_X \circ \ell_Y \sim id}$$

which is right but without any interest.

Decorated rules for substitution and replacement

Strong equations form a congruence:

$$(\equiv\text{-subs})\ \frac{g_1 \equiv g_2}{g_1 \circ f \equiv g_2 \circ f} \qquad (\equiv\text{-repl})\ \frac{f_1 \equiv f_2}{g \circ f_1 \equiv g \circ f_2}$$

Decorated rules for substitution and replacement

Strong equations form a congruence:

$$(\equiv\text{-subs})\ \frac{g_1 \equiv g_2}{g_1 \circ f \equiv g_2 \circ f} \qquad (\equiv\text{-repl})\ \frac{f_1 \equiv f_2}{g \circ f_1 \equiv g \circ f_2}$$

Weak equations do not form a congruence:

$$(\sim\text{-subs})\ \frac{g_1 \sim g_2}{g_1 \circ f \sim g_2 \circ f} \qquad (0\text{-}\sim\text{-repl})\ \frac{f_1 \sim f_2 \quad g^{(0)}}{g \circ f_1 \sim g \circ f_2 : X \to Z}$$

Indeed: f_1 and f_2 may modify the state in a different way, so that $g \circ f_1$ and $g \circ f_2$ may return different values if g is not pure.

A decorated specification for states

For each X, a type V_X for values, two operations:

$$\begin{array}{ll} \ell_X^{(1)} : \mathbb{1} \to V_X & \text{(lookup) : an accessor} \\ u_X^{(2)} : V_X \to \mathbb{1} & \text{(update) : a modifier} \end{array}$$

and weak equations:

$$\begin{array}{ll} \ell_X \circ u_X \sim id & \\ \ell_Y \circ u_X \sim \ell_Y \circ \langle\,\rangle & \text{when } Y \neq X \end{array}$$

Proof #1 is right: it can be properly decorated

The annihilation lookup-update (ALU) property:

$$\boxed{u_X \circ \ell_X \equiv id}$$

Proof. By observation: prove that $\ell_Y \circ u_X \circ \ell_X \sim \ell_Y$ for each Y.
When $Y = X$:

$$(\sim\text{-subs})\ \frac{\ell_X \circ u_X \sim id}{\ell_X \circ u_X \circ \ell_X \sim \ell_X}$$

Proof #1 is right: it can be properly decorated

The annihilation lookup-update (ALU) property:

$$\boxed{u_X \circ \ell_X \equiv id}$$

Proof. By observation: prove that $\ell_Y \circ u_X \circ \ell_X \sim \ell_Y$ for each Y.
When $Y = X$:

$$(\sim\text{-subs})\ \frac{\ell_X \circ u_X \sim id}{\ell_X \circ u_X \circ \ell_X \sim \ell_X}$$

When $Y \neq X$:

$$(\sim\text{-trans})\ \dfrac{(\sim\text{-subs})\ \dfrac{\ell_Y \circ u_X \sim \ell_Y \circ \langle\,\rangle}{\ell_Y \circ u_X \circ \ell_X \sim \ell_Y \circ \langle\,\rangle \circ \ell_X} \qquad (\equiv\text{-to-}\sim)\ \dfrac{(\equiv\text{-repl})\ \dfrac{(1\text{-unit})\ \dfrac{\ell_X^{(1)}}{\langle\,\rangle \circ \ell_X \equiv id}}{\ell_Y \circ \langle\,\rangle \circ \ell_X \equiv \ell_Y}}{\ell_Y \circ \langle\,\rangle \circ \ell_X \sim \ell_Y}}{\ell_Y \circ u_X \circ \ell_X \sim \ell_Y}$$

Other properties of imperative languages

The 7 properties in [Plotkin&Power 02] can be proved similarly.
For instance the commutation update-update (CUU) property,
is proved in the paper.

Other properties of imperative languages

The 7 properties in [Plotkin&Power 02] can be proved similarly. For instance the commutation update-update (CUU) property, is proved in the paper. When $X \neq Y$:

$$\boxed{\text{The order of storing values in } X \text{ and } Y \text{ does not matter}}$$

Other properties of imperative languages

The 7 properties in [Plotkin&Power 02] can be proved similarly. For instance the commutation update-update (CUU) property, is proved in the paper. When $X \neq Y$:

$$\boxed{\text{The order of storing values in } X \text{ and } Y \text{ does not matter}}$$

which is formalized as:

$$\boxed{u_Y \circ (u_X \times id) \equiv u_X \circ (id \times u_Y) : V_X \times V_Y \to \mathbb{1}}$$

where $\times$ is the semi-pure product from [Dumas&Duval&Reynaud] *Cartesian effect categories are Freyd-categories* JSC 2011. *ACCAT'09.*

$$\begin{tikzcd}
V_X \arrow[rr, "u_X^{(2)}"] & & \mathbb{1} \\
& \equiv & \\
V_X \times V_Y \arrow[u] \arrow[d] \arrow[rr, dashed, "(u_X \times id)^{(2)}"] & & \mathbb{1} \times V_Y \arrow[u] \arrow[d] \\
& \sim & \\
V_Y \arrow[rr, "id^{(0)}"] & & V_Y
\end{tikzcd}$$

Outline

Explicit proofs

Another way to prove results about states:

1. introduce explicitly a type of states S

Explicit proofs

Another way to prove results about states:

1. introduce explicitly a type of states S
2. expand (translate) the decorations

$f^{(0)} : X \to Y$	$f : X \to Y$
$f^{(1)} : X \to Y$	$f : X \times S \to Y$
$f^{(2)} : X \to Y$	$f : X \times S \to Y \times S$
$f \equiv g : X \to Y$	$f \equiv g : X \times S \to Y \times S$
$f \sim g : X \to Y$	$\pi \circ f \equiv \pi \circ g : X \times S \to Y$

Explicit proofs

Another way to prove results about states:

1. introduce explicitly a type of states S
2. expand (translate) the decorations

$f^{(0)} : X \to Y$	$f : X \to Y$
$f^{(1)} : X \to Y$	$f : X \times S \to Y$
$f^{(2)} : X \to Y$	$f : X \times S \to Y \times S$
$f \equiv g : X \to Y$	$f \equiv g : X \times S \to Y \times S$
$f \sim g : X \to Y$	$\pi \circ f \equiv \pi \circ g : X \times S \to Y$

3. prove in the "usual" (not decorated) logic

Explicit proofs

Another way to prove results about states:

1. introduce explicitly a type of states S
2. expand (translate) the decorations

$f^{(0)} : X \to Y$	$f : X \to Y$
$f^{(1)} : X \to Y$	$f : X \times S \to Y$
$f^{(2)} : X \to Y$	$f : X \times S \to Y \times S$
$f \equiv g : X \to Y$	$f \equiv g : X \times S \to Y \times S$
$f \sim g : X \to Y$	$\pi \circ f \equiv \pi \circ g : X \times S \to Y$

3. prove in the "usual" (not decorated) logic

But the notion of effect is lost.

A span of "logics"

$$\begin{tikzcd} & \texttt{decorations} \arrow[dl] \arrow[dr] & \\ \texttt{syntax} & & \texttt{semantics} \end{tikzcd}$$

- $\texttt{decorations} \to \texttt{syntax}$:
  forget the decorations
- $\texttt{decorations} \to \texttt{semantics}$:
  expansion, with an explicit S for states

From proofs to models

The expansion:

- maps decorated proofs to “usual” explicit proofs

From proofs to models

The expansion:

- maps decorated proofs to "usual" explicit proofs
- and provides a notion of decorated model

because it can be seen as a functor F with a right adjoint:

$$\begin{tikzcd} \texttt{decorations} \arrow[r, shift left, "F"] \arrow[r, phantom, "\perp"] & \texttt{semantics} \arrow[l, shift left, "G"] \end{tikzcd}$$

$$Mod_{\mathrm{deco}}(\Sigma, G\Theta) \cong Mod_{\mathrm{expl}}(F\Sigma, \Theta)$$

For instance:

Σ is the decorated specification for states

Θ is **Set** with the distinguished set $S = \prod_X V_X$

From states to exceptions

- We can prove properties of imperative languages in a logic which respects the syntax of the language.

From states to exceptions

- We can prove properties of imperative languages in a logic which respects the syntax of the language.

- THUS, we can prove properties of exceptions in a logic which respects the syntax of exceptions.
  [Dumas&Duval&Fousse&Reynaud] *Decorated proofs for computational effects: exceptions. Submitted for publication.*

From states to exceptions

- We can prove properties of imperative languages in a logic which respects the syntax of the language.
- THUS, we can prove properties of exceptions in a logic which respects the syntax of exceptions.
  [Dumas&Duval&Fousse&Reynaud] *Decorated proofs for computational effects: exceptions. Submitted for publication.*
- This is due to the duality between states and the core part of exceptions.
  [Dumas&Duval&Fousse&Reynaud] *A duality between exceptions and states. To appear in MSCS. ACCAT'11.*

Conclusion and future work

We have designed a framework for effects which provides a denotational semantics and a proof system.

Conclusion and future work

We have designed a framework for effects which provides a denotational semantics and a proof system.

Our projects include:

- Using a proof assistant for proving decorated properties.
- Extending our framework for combining effects by composing spans.

Thank you!