Proofs About Programs Involving Computational Effects

Burak Ekici
with J-G. Dumas, D. Duval, D. Pous

LJK Grenoble, France

October 11, 2013

Motivation

- Verifying properties of programs involving computational effects such as:
  - State (Side Effect)
  - Exceptions
  - IO
  - Partiality
  - ...

  by using categorical semantics: cartesian effect categories [Dumas et al'12].
- Developing related `Coq libraries` for each effect and composing them in the end.

Motivation

- Verifying properties of programs involving computational effects such as:
  - State (Side Effect)
  - Exceptions
  - IO
  - Partiality
  - ...

  by using categorical semantics: cartesian effect categories [Dumas et al'12].
- Developing related `Coq libraries` for each effect and composing them in the end.

Specification for States Effect

Objects: Sets

- $\mathtt{Val_i}$ (values), $\mathtt{void}$ ($\{*\}$), $\mathtt{S}$ (states)

Morphisms: Functions

- $[\![\mathtt{id_i}]\!] : \mathtt{Val_i} \to \mathtt{Val_i}$, $[\![\mathtt{id_{void}}]\!] : \mathtt{void} \to \mathtt{void}$
- $[\![\mathtt{lookup_i}]\!] : \mathtt{S} \to \mathtt{Val_i}$, $[\![\mathtt{update_i}]\!] : \mathtt{Val_i} \times \mathtt{S} \to \mathtt{S}$ where $\mathtt{i}$ is a memory location (variable).

Functions are classified and decorated:

- pure: e.g., $\mathtt{id}_\mathtt{i}^{pure} : \mathtt{Val_i} \to \mathtt{Val_i}$, $\mathtt{id}_\mathtt{void}^{pure} : \mathtt{void} \to \mathtt{void}$
- accessors: e.g., $\mathtt{lookup}_\mathtt{i}^{ro} : \mathtt{void} \to \mathtt{Val_i}$
- modifiers: e.g., $\mathtt{update}_\mathtt{i}^{rw} : \mathtt{Val_i} \to \mathtt{void}$
- Hierarchy rules among functions: $\dfrac{\mathtt{f}^{pure}}{\mathtt{f}^{ro}}$, $\dfrac{\mathtt{f}^{ro}}{\mathtt{f}^{rw}}$

Equations with decorated functions are classified:

- strong equality ($\mathtt{f} == \mathtt{g}$): result + effect
- weak equality ($\mathtt{f} \sim \mathtt{g}$): result
- Hierarchy rules among equations: $\dfrac{\mathtt{f}^{rw} == \mathtt{g}^{rw}}{\mathtt{f}^{rw} \sim \mathtt{g}^{rw}}$, $\dfrac{\mathtt{f}^{ro} \sim \mathtt{g}^{ro}}{\mathtt{f}^{ro} == \mathtt{g}^{ro}}$

Specification for States Effect

Objects: Sets

- $\mathtt{Val_i}$ (values), $\mathtt{void}$ ($\{*\}$), $\mathtt{S}$ (states)

Morphisms: Functions

- $[[\mathtt{id_i}]] : \mathtt{Val_i} \to \mathtt{Val_i}$, $[[\mathtt{id_{void}}]] : \mathtt{void} \to \mathtt{void}$
- $[[\mathtt{lookup_i}]] : \mathtt{S} \to \mathtt{Val_i}$, $[[\mathtt{update_i}]] : \mathtt{Val_i} \times \mathtt{S} \to \mathtt{S}$ where $\mathtt{i}$ is a memory location (variable).

Functions are classified and decorated:

- pure: e.g., $\mathtt{id}_\mathtt{i}^{pure} : \mathtt{Val_i} \to \mathtt{Val_i}$, $\mathtt{id}_\mathtt{void}^{pure} : \mathtt{void} \to \mathtt{void}$
- accessors: e.g., $\mathtt{lookup}_\mathtt{i}^{ro} : \mathtt{void} \to \mathtt{Val_i}$
- modifiers: e.g., $\mathtt{update}_\mathtt{i}^{rw} : \mathtt{Val_i} \to \mathtt{void}$
- Hierarchy rules among functions: $\dfrac{\mathtt{f}^{pure}}{\mathtt{f}^{ro}}$, $\dfrac{\mathtt{f}^{ro}}{\mathtt{f}^{rw}}$

Equations with decorated functions are classified:

- strong equality ($\mathtt{f} == \mathtt{g}$): result + effect
- weak equality ($\mathtt{f} \sim \mathtt{g}$): result
- Hierarchy rules among equations: $\dfrac{\mathtt{f}^{rw} == \mathtt{g}^{rw}}{\mathtt{f}^{rw} \sim \mathtt{g}^{rw}}$, $\dfrac{\mathtt{f}^{ro} \sim \mathtt{g}^{ro}}{\mathtt{f}^{ro} == \mathtt{g}^{ro}}$

Specification for States Effect

Objects: Sets

- $\mathtt{Val_i}$ (values), $\mathtt{void}$ ($\{*\}$), $\mathtt{S}$ (states)

Morphisms: Functions

- $[[\mathtt{id_i}]] : \mathtt{Val_i} \to \mathtt{Val_i}$, $[[\mathtt{id_{void}}]] : \mathtt{void} \to \mathtt{void}$
- $[[\mathtt{lookup_i}]] : \mathtt{S} \to \mathtt{Val_i}$, $[[\mathtt{update_i}]] : \mathtt{Val_i} \times \mathtt{S} \to \mathtt{S}$ where $\mathtt{i}$ is a memory location (variable).

Functions are classified and decorated:

- pure: e.g., $\mathtt{id}_\mathtt{i}^{pure} : \mathtt{Val_i} \to \mathtt{Val_i}$, $\mathtt{id}_\mathtt{void}^{pure} : \mathtt{void} \to \mathtt{void}$
- accessors: e.g., $\mathtt{lookup}_\mathtt{i}^{ro} : \mathtt{void} \to \mathtt{Val_i}$
- modifiers: e.g., $\mathtt{update}_\mathtt{i}^{rw} : \mathtt{Val_i} \to \mathtt{void}$
- Hierarchy rules among functions: $\dfrac{\mathtt{f}^{pure}}{\mathtt{f}^{ro}}$, $\dfrac{\mathtt{f}^{ro}}{\mathtt{f}^{rw}}$

Equations with decorated functions are classified:

- strong equality ($\mathtt{f} == \mathtt{g}$): result + effect
- weak equality ($\mathtt{f} \sim \mathtt{g}$): result
- Hierarchy rules among equations: $\dfrac{\mathtt{f}^{rw} == \mathtt{g}^{rw}}{\mathtt{f}^{rw} \sim \mathtt{g}^{rw}}$, $\dfrac{\mathtt{f}^{ro} \sim \mathtt{g}^{ro}}{\mathtt{f}^{ro} == \mathtt{g}^{ro}}$

“Core” Operations: Update & Lookup

Let *Loc* be the set of locations. For each location $\mathtt{i} \in \mathtt{Loc}$:

- the set of values, that could be stored in it, is denoted by $\mathtt{Val_i}$
- there are two main operations and equations

the value		the stored value
$\mathtt{a} \in \mathtt{Val_i}$	$\xrightarrow{\mathtt{update_i}}$	$\boxed{a}_i$
$\mathtt{a} \in \mathtt{Val_i}$	$\xleftarrow{\mathtt{lookup_i}}$	$\boxed{a}_i$

$$\mathtt{lookup_i} : 1 \to \mathtt{Val_i}$$
$$\mathtt{update_i} : \mathtt{Val_i} \to 1$$
$$\mathtt{lookup_i} \circ \mathtt{update_i} \sim \mathtt{id_{Val_i}} \quad \text{(axiom-1)}$$
$$\mathtt{update_i} \circ \mathtt{lookup_i} == \mathtt{id_1} \quad \text{(axiom-2)}$$

$$\mathtt{lookup_i} : \mathtt{S} \to \mathtt{Val_i}$$
$$\mathtt{update_i} : \mathtt{Val_i} \times \mathtt{S} \to \mathtt{S}$$
$$\mathtt{b} \mapsto \boxed{}^{\mathtt{b}}_i \mapsto \mathtt{b} \sim \mathtt{b} \mapsto \mathtt{b}$$
$$\boxed{\mathtt{a}}_i \mapsto \mathtt{a} \mapsto \boxed{\mathtt{a}}_i == \mathtt{a} \mapsto \mathtt{a}$$

“Core” Operations: Update & Lookup

Let *Loc* be the set of locations. For each location $\mathtt{i} \in \mathtt{Loc}$:

- the set of values, that could be stored in it, is denoted by $\mathtt{Val_i}$
- there are two main operations and equations

the value		the stored value
$\mathtt{a} \in \mathtt{Val_i}$	$\xrightarrow{\mathtt{update_i}}$	$\boxed{a}_i$
$\mathtt{a} \in \mathtt{Val_i}$	$\xleftarrow{\mathtt{lookup_i}}$	$\boxed{a}_i$

$\mathtt{lookup_i} : 1 \to \mathtt{Val_i}$

$\mathtt{update_i} : \mathtt{Val_i} \to 1$

$\mathtt{lookup_i} \circ \mathtt{update_i} \sim \mathtt{id}_{\mathtt{Val_i}}$

(axiom-1)

$\mathtt{update_i} \circ \mathtt{lookup_i} == \mathtt{id}_1$

(axiom-2)

$\mathtt{lookup_i} : \mathtt{S} \to \mathtt{Val_i}$

$\mathtt{update_i} : \mathtt{Val_i} \times \mathtt{S} \to \mathtt{S}$

$\mathtt{b} \mapsto \boxed{\mathtt{a}}_i \mapsto \mathtt{b} \sim \mathtt{b} \mapsto \mathtt{b}$

$\boxed{\mathtt{a}}_i \mapsto \mathtt{a} \mapsto \boxed{\mathtt{a}}_i == \mathtt{a} \mapsto \mathtt{a}$

“Core” Operations: Update & Lookup

Let *Loc* be the set of locations. For each location $\mathtt{i} \in \mathtt{Loc}$:

- the set of values, that could be stored in it, is denoted by $\mathtt{Val_i}$
- there are two main operations and equations

the value		the stored value
$\mathtt{a} \in \mathtt{Val_i}$	$\xrightarrow{\mathtt{update_i}}$	$\boxed{a}_i$
$\mathtt{a} \in \mathtt{Val_i}$	$\xleftarrow{\mathtt{lookup_i}}$	$\boxed{a}_i$

$\mathtt{lookup_i} : 1 \to \mathtt{Val_i}$

$\mathtt{update_i} : \mathtt{Val_i} \to 1$

$\mathtt{lookup_i} \circ \mathtt{update_i} \sim \mathtt{id}_{\mathtt{Val_i}}$ (axiom-1)

$\mathtt{update_i} \circ \mathtt{lookup_i} == \mathtt{id_1}$ (axiom-2)

$\mathtt{lookup_i} : \mathtt{S} \to \mathtt{Val_i}$

$\mathtt{update_i} : \mathtt{Val_i} \times \mathtt{S} \to \mathtt{S}$

$\mathtt{b} \mapsto \boxed{\mathtt{a}}_i^{\mathtt{b}} \mapsto \mathtt{b} \sim \mathtt{b} \mapsto \mathtt{b}$

$\boxed{\mathtt{a}}_i \mapsto \mathtt{a} \mapsto \boxed{\mathtt{a}}_i == \mathtt{a} \mapsto \mathtt{a}$

Decorated Rules: Monadic Equational Logic

$$(s\text{-subs})\ \frac{f^{rw} : X \to Y \quad g_1^{rw} == g_2^{rw} : Y \to Z}{g_1 \circ f == g_2 \circ f : X \to Z}$$

$$(s\text{-repl})\ \frac{f_1^{rw} == f_2^{rw} : X \to Y \quad g^{rw} : Y \to Z}{g \circ f_1 == g \circ f_2 : X \to Z}$$

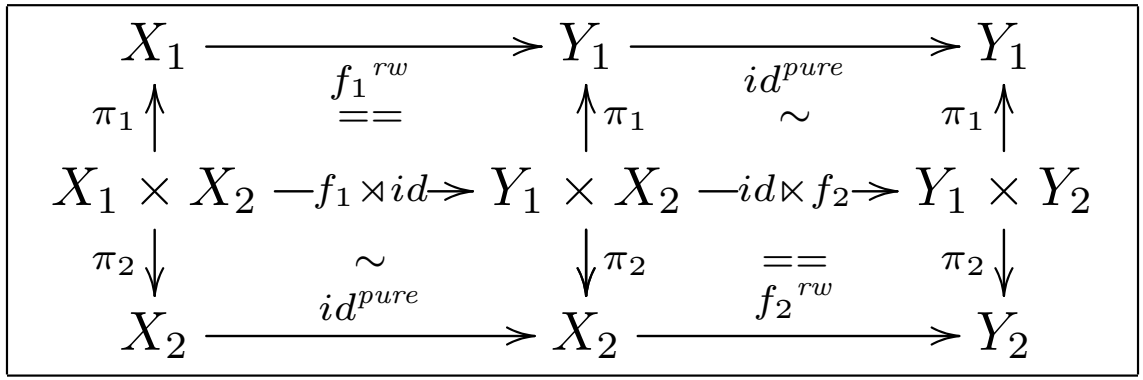
$$(\text{pure-to-ro})\ \frac{f^{pure}}{f^{ro}} \qquad (\text{ro-to-rw})\ \frac{f^{ro}}{f^{rw}}$$

$$(\text{ro-}w\text{-to-}s)\ \frac{f^{ro} \sim g^{ro}}{f == g} \qquad (s\text{-to-}w)\ \frac{f^{rw} == g^{rw}}{f \sim g}$$

$$(w\text{-subs})\ \frac{f^{rw} : X \to Y \quad g_1^{rw} \sim g_2^{rw} : Y \to Z}{g_1 \circ f \sim g_2 \circ f : X \to Z}$$

$$(\text{pure-}w\text{-repl})\ \frac{f_1^{rw} \sim f_2^{rw} : X \to Y \quad g^{pure} : Y \to Z}{g \circ f_1 \sim g \circ f_2 : X \to Z}$$

Decorated Rules: Sequential Products



$$\text{(dec-prod-proj-1)}\ \frac{f_1^{pure} : X_1 \to Y_1 \quad f_2^{rw} : X_2 \to Y_2}{\pi_{Y_1,Y_2,1} \circ (f_1 \times f_2) \ \sim f_1 \circ \pi_{X_1,X_2,1}}$$

$$\text{(dec-p-prod-proj-1)}\ \frac{f_1^{rw} : X_1 \to Y_1 \quad f_2^{pure} : X_2 \to Y_2}{\pi_{Y_1,Y_2,1} \circ (f_1 \times f_2)_p \ == f_1 \circ \pi_{X_1,X_2,1}}$$

$$\text{(dec-prod-proj-2)}\ \frac{f_1^{pure} : X_1 \to Y_1 \quad f_2^{rw} : X_2 \to Y_2}{\pi_{Y_1,Y_2,2} \circ (f_1 \times f_2) \ == f_2 \circ \pi_{X_1,X_2,2}}$$

$$\text{(dec-p-prod-proj-2)}\ \frac{f_1^{rw} : X_1 \to Y_1 \quad f_2^{pure} : X_2 \to Y_2}{\pi_{Y_1,Y_2,2} \circ (f_1 \times f_2)_p \ \sim f_2 \circ \pi_{X_1,X_2,2}}$$

Properties of the State Structure by Plotkin et al.

1. `Annihilation lookup-update`
   $\forall\, i \in Loc,\ u_i \circ l_i == id_1 : 1 \to 1$
2. `Interaction lookup-lookup`
   $\forall\, i \in Loc,\ l_i \circ \langle\,\rangle_i \circ l_i == l_i : 1 \to V_i$
3. `Interaction update-update`
   $\forall\, i \in Loc,\ u_i \circ \pi_2 \circ (u_i \rtimes id_i) == u_i \circ \pi_2 : V_i \times V_i \to 1$
4. `Interaction update-lookup`
   $\forall\, i \in Loc,\ l_i \circ u_i \sim\ id_i : V_i \to V_i$
5. `Commutation lookup-lookup` $\forall\, i \neq j \in$
   $Loc,\ (id_i \ltimes\ l_j) \circ l_i == perm_{j,i} \circ\ (id_j \ltimes\ l_i) \circ l_j : 1 \to V_i \times V_j$
6. `Commutation update-update` $\forall\, i \neq j \in$
   $Loc,\ u_j \circ \pi_2 \circ (u_i \rtimes id_j) == u_i \circ \pi_1 \circ (id_i \ltimes u_j) : V_i \times V_j \to 1$
7. `Commutation update-lookup` $\forall\, i \neq j \in Loc,\ l_j \circ u_i \circ \pi_1 ==$
   $\pi_2 \circ (u_i \rtimes id_j) \circ (id_i \ltimes l_j) : V_i \times 1 \to V_j\ .$

A Simple Example [Plotkin-Power'02]

u_i^{rw} is $\texttt{update}_\texttt{i}^{rw}$

Interaction update-update: $\forall\, i \in Loc$,
$u_i^{rw} \circ \pi_2^{pure} \circ (u_i \rtimes id_i)^{rw} == u_i^{rw} \circ \pi_2^{pure} : \texttt{Val}_\texttt{i} \times \texttt{Val}_\texttt{i} \to \texttt{void}$

Example	Inference System	Proof Assistant: Coq
$i := 2;\ i := 3;$	$\begin{array}{ccc} 2 & \xrightarrow{u_i} & * \\ {\scriptstyle\pi_1}\uparrow & == & \uparrow{\scriptstyle\pi_1} \\ (2,3) & \xrightarrow{u_i \rtimes id_i} & (*,3) \\ {\scriptstyle\pi_2}\downarrow & \sim & \downarrow{\scriptstyle\pi_2} \\ 3 & \xrightarrow[id_i]{} & 3 \xrightarrow{u_i} * \end{array}$	$u_i \circ \pi_2 \circ (\mathsf{pprod}(u_i\ id))$
$==$	$==$	$==$
$i := 3;$	$(2,3) \xrightarrow{\pi_2} 3 \xrightarrow{u_i} *$	$u_i \circ \pi_2$

Coq Implementation of a Proof: Interaction update-update

`obs-local-global`

2 subgoals i : Loc
s : forall (X : Type) (f g : term () X),
(forall i : Loc, lookup i ∘ f ∼ lookup i ∘ g) → f == g
Heqs : s = obs_local_global
k : Loc
e : i = k
==========(1/2)
lookup i ∘ ((update i ∘ pi2) ∘ perm_prod (update i) id) ∼
lookup i ∘ (update i ∘ pi2)
==========(2/2)
lookup k ∘ ((update i ∘ pi2) ∘ perm_prod (update i) id) ∼
lookup k ∘ (update i ∘ pi2)

Coq Implementation of a Proof: Interaction update-update

axiom-1

2 subgoals i : Loc
s : forall (X : Type) (f g : term () X),
(forall i : Loc, lookup i ∘ f ∼ lookup i ∘ g) → f == g
Heqs : s = obs_local_global
k : Loc
e : i = k
==========(1/2)
pi2 ∘ perm_prod (update i) id ∼ lookup i ∘ (update i ∘ pi2)
==========(2/2)
lookup k ∘ ((update i ∘ pi2) ∘ perm_prod (update i) id) ∼
lookup k ∘ (update i ∘ pi2)

Coq Implementation of a Proof: Interaction update-update

dec-p-prod-proj-2

2 subgoals i : Loc
s : forall (X : Type) (f g : term () X),
(forall i : Loc, lookup i ∘ f ∼ lookup i ∘ g) → f == g
Heqs : s = obs_local_global
k : Loc
e : i = k
==========(1/2)
pi2 ∼ lookup i ∘ (update i ∘ pi2)
==========(2/2)
lookup k ∘ ((update i ∘ pi2) ∘ perm_prod (update i) id) ∼
lookup k ∘ (update i ∘ pi2)

Coq Implementation of a Proof: Interaction update-update

axiom-1

2 subgoals i : Loc
s : forall (X : Type) (f g : term () X),
(forall i : Loc, lookup i ∘ f ∼ lookup i ∘ g) → f == g
Heqs : s = obs_local_global
k : Loc
e : i = k
==========(1/2)
pi2 ∼ pi2
==========(2/2)
lookup k ∘ ((update i ∘ pi2) ∘ perm_prod (update i) id) ∼
lookup k ∘ (update i ∘ pi2)

Coq Implementation of a Proof: Interaction update-update

. . .

1 subgoal
i : Loc
s : forall (X : Type) (f g : term () X),
(forall i : Loc, lookup i ∘ f ∼ lookup i o g) → f == g
Heqs : s = obs_local_global
k : Loc
n0 : i ≠ k
==========(1/1)
lookup k ∘ ((update i ∘ pi2) ∘ perm_prod (update i) id) ∼
lookup k ∘ (update i ∘ pi2)

So far & Next

1. Preliminary results : A Coq library for the global states effect ($\approx$ 1700 LoC) and a related submission to JFLA14
   - Library is available: `http://coqeffects.forge.imag.fr/`
2. Future work :
   - developing the framework for local state (allocation)
   - developing the library for exceptions
   - developing the concepts/Coq for combining effects (monad transformers [Haskell])
   - generalization to the other effects

So far & Next

1. Preliminary results : A Coq library for the global states effect ($\approx$ 1700 LoC) and a related submission to JFLA14
   - Library is available: `http://coqeffects.forge.imag.fr/`
2. Future work :
   - developing the framework for local state (allocation)
   - developing the library for exceptions
   - developing the concepts/Coq for combining effects (monad transformers [Haskell])
   - generalization to the other effects

The End!

MERCI POUR VOTRE ATTENTION!

QUESTIONS?

The End!

MERCI POUR VOTRE ATTENTION!

QUESTIONS?