# An introduction to monadic semantics for computational effects

Fabio Zanasi
ILLC - Universiteit van Amsterdam

April 12, 2012

## Background

Denotational semantics of programming languages.

- ▶ Programs ≈ functions in the mathematical sense: value → value
- $\qquad \qquad \textbf{Program computation can have side effects: value} \overset{\textit{effects}}{\mapsto} \textbf{value}$
- Computational effects:
  - Modify the state.
  - I/O
  - Exception throwing.

#### Outline

- Warmup: semantics for typed λ-calculus.
- Monads and Kleisli triples.
- Monadic metalanguage.
- Sketch: monadic-style translation and direct-style interpretation.

# λ-calculus

$$\lambda_{\rightarrow}^{\times}$$
 - syntax

Types

$$a ::= \mathbf{a} \mid a'$$
 $A ::= a \mid A \rightarrow B \mid A \times B$ 

Terms

$$x ::= \mathbf{x} \mid x'$$
 
$$M ::= x \mid \lambda x : A.M \mid MN \mid < M, N > \mid \pi_1 M \mid \pi_2 N$$

$$\lambda_{\rightarrow}^{\times}$$
 - typing

Environment

$$\Gamma = \{x_1 : A_1, x_2 : A_2, \dots, x_n : A_n\}$$

Type judgement

$$\Gamma \vdash M : A$$

Typing rules

$$\lambda_{\rightarrow}^{\times}$$
 - typing rules

$$\overline{\Gamma x : A \vdash x : A} \ Var$$

$$\frac{\Gamma, x: A \vdash M: B}{\Gamma \vdash \lambda x: A.M: A \rightarrow B} \ Int \rightarrow \qquad \frac{\Gamma \vdash M: A \rightarrow B \quad \Gamma \vdash N: A}{\Gamma \vdash MN: B} \ El \rightarrow$$

$$\frac{\Gamma \vdash M \colon A \quad \Gamma \vdash N \colon B}{\Gamma \vdash \langle N, M \rangle \colon A \times B} \quad Int \times \quad \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_1 M \colon A} \quad El_I \times \quad \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \times B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \to B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \to B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \to B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \to B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \to B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \to B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \to B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \to B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \to B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \to B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \to B}{\Gamma \vdash \pi_2 M \colon B} \quad El_r \times \frac{\Gamma \vdash M \colon A \to B}{\Gamma \vdash \Phi} \quad El_r \times \frac{\Gamma \vdash M \colon A \to B}{\Gamma \vdash \Phi} \quad El_r \times \frac{\Gamma \vdash M \colon A \to B}{\Gamma \vdash \Phi} \quad El_r \times$$

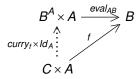
#### Definition

A category is called  $\underline{\text{cartesian closed}}$  (CC) if it has all finite products and exponentials.

#### Definition

An exponential of objects A and B in a category C is given by an object  $B^A$  and a morphism  $eval_{AB}: B^A \times A \rightarrow B$  with the following UMP.

▶ For any object  $C \in C_{Ob}$  and morphism  $f : C \times A \rightarrow B$  there is a unique morphism  $curry_f : C \rightarrow B^A$  such that the following diagram commutes.



#### Fix a CC category C.

- Types are interpreted as objects.
- Environments are interpreted as products of objects.
- Type judgements are interpreted as morphisms.

Types are interpreted as objects.

$$||a|| = A_a$$

$$||A \rightarrow B|| = ||B||^{|A||}$$

$$||A \times B|| = ||A|| \times ||B||$$

Environments are interpreted as products of objects.

$$\|\varnothing\| = 1_{\mathcal{C}}$$
$$\|\Gamma, x : A\| = \|\Gamma\| \times \|A\|$$

Type judgements are interpreted as morphisms.

## Monads

## Eugenio Moggi's insight

Pure program = 
$$A \rightarrow B$$

Impure program =  $A \rightarrow TB$ 

Semantics of *T* given by a monad.

## Computational effects

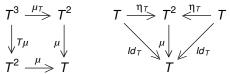
- ► TA = A + E programs with exceptions
- $TA = A + \{\bot\}$  possibly non-terminating programs
- $TA = \mathcal{P}_{fin}A$  non-deterministic programs
- $TA = (A \times S)^S$  imperative programs
- $TA = A \times N$  programs with timers
- $TA = R^{R^A}$  programs with a continuation

#### Monads

#### Definition

A monad on  $\mathcal{C}$  is a triple  $\langle T, \eta, \mu \rangle$  where  $T : \mathcal{C} \to \mathcal{C}$  is an endofunctor,  $\eta: Id \to T$ ,  $\mu: T^2 \to T$  are natural transformations such that the following diagrams commute.





## Kleisli triples

#### Definition

A Kleisli triple in a category C is a triple  $< T, \eta, \cdot^* >$  where

- ▶  $T: C_{Ob} \rightarrow C_{Ob}$  expresses the type of computations,
- $\eta = \{\eta_A : A \to TA\}_{A \in \mathcal{C}_O b}$  expresses the inclusion of values into computations,
- ·\* :  $(f: A \to TB) \mapsto (f^*: TA \to TB)$  expresses the extension of f to act on computations,

and the following equations hold.

$$\begin{array}{rcl} \eta_A^{\star} & = & Id_{TA} \\ f^{\star} \circ \eta_A & = & f \\ g^{\star} \circ f^{\star} & = & (g^{\star} \circ f)^{\star} \end{array}$$

## Kleisli Category

#### Definition

Given a category  $\mathcal{C}$ , a Kleisli triple  $< \mathcal{T}, \eta, \cdot^* > \text{in } \mathcal{C}$ , the Kleisli Category  $\mathcal{C}_{\mathcal{T}}$  is defined as follows.

- $\mathcal{C}_{\mathcal{T}Ob} := \mathcal{C}_{Ob}$
- $\vdash Hom_{\mathcal{C}_{\mathcal{T}}}(A,B) := Hom_{\mathcal{C}}(A,TB)$
- $g \circ f$  in  $C_T := g^* \circ f$  in C

# Monadic metalanguage

#### ML

#### The idea

- The syntactic counterpart of a category equipped with a monad T
- Implementation: a λ-calculus parameterized to an effect T
  - Syntactically: a new type constructor T and associated term constructors val and let.
  - Semantically: CCC equipped with a monad to interpret T.

## ML - syntax

New type constructor T

$$A ::= a \mid TA \mid A \rightarrow B \mid A \times B$$

New term constructors val and let

$$M ::= x \mid \lambda x : A.M \mid MN \mid < M, N > \mid \pi_1 M \mid \pi_2 N \mid valM \mid let x = M in N$$

## ML - typing rules

Additional typing rules

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash valM : TA} Int val \qquad \frac{\Gamma \vdash M : TA \quad \Gamma, x : A \vdash N : TB}{\Gamma \vdash let \ x = M \ in \ N : TB} Int \ let$$

#### ML - axioms

➤ The axioms for *val* and *let* capture the intended interpretation of the new constructors by matching the equations of a Kleisli triple.

$$let x = M in val x = M$$
 (1)

$$let x = valM in valN = N[x := M]$$
 (2)

let 
$$y = L$$
 in (let  $x = M$  in  $N$ ) = let  $x = (let y = L$  in  $M$ ) in  $N(3)$ 

With y not free in N in (3).

Remark: A complete definition of ML includes equation judgements and inference rules. The three axioms can be formulated as equations that are derivable in ML. See Moggi 1991 for more details.

Interpretation of types.

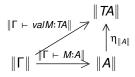
$$||a|| = A_a$$
  
 $||TA|| = T||A||$   
 $||A \rightarrow B|| = ||B||^{||A||}$   
 $||A \times B|| = ||A|| \times ||B||$ 

#### Interpretation of val

Intuitively *valM*: *TA* expresses the view of a value *M* of type *A* as a 'special case' of computation of type *TA*.

By type derivation of  $\Gamma \vdash valM : TA$  we can assume a morphism  $\|\Gamma \vdash M : A\|$ .

$$\|\Gamma \vdash valM : TA\| := \eta_{\|A\|} \circ \|\Gamma \vdash M : A\|$$



#### Interpretation of let

Intuitively let x = M in N stands for the application of  $\lambda x.N$  to M when N and M are not just values, but computations.

$$\frac{\Gamma \vdash M : TA \quad \Gamma, x : A \vdash N : TB}{\Gamma \vdash let \ x = M \ in \ N : TB} \ Int \ let$$

- ▶ Suppose that  $\|\Gamma \vdash M : TA\|$  is  $f : \|\Gamma\| \to \|TA\|$  and  $\|\Gamma, x : A \vdash N : TB\|$  is  $g : \|\Gamma\| \times \|A\| \to \|TB\|$ . Then  $\|\Gamma \vdash Iet \ x = M \ in \ N\| : TB$  should be some morphism  $h : \|\Gamma\| \to \|TB\|$ .
- ▶ Take the Kleisli composition of g and  $\langle Id_{\|\Gamma\|} \times f \rangle$ .
- So intuitively we want  $\| \text{let } x = M \text{ in } N \| = g^* \circ f$ .

#### A problem

 Complication given by the presence of a non-empty environment Γ.

$$g^* = \|\Gamma, x : A \vdash N : TB\|^* : T(\|\Gamma\| \times \|A\|) \to T\|B\|$$
$$< Id_{\|\Gamma\|} \times f > : \|\Gamma\| \to \|\Gamma\| \times \|TA\|$$

▶ Domain-range mismatch in  $g^* \circ < Id_{\parallel} \Gamma \parallel \times f >$ .

$$\|\Gamma\| \xrightarrow{\|\Gamma \vdash M:TA\|} > \|TA\|$$

$$\downarrow < Id_{\|\Gamma\|}, \|\Gamma \vdash M:TA\| >$$

$$\|\Gamma\| \times \|A\|$$

$$\downarrow \|\Gamma, x:A \vdash N:TB\|$$

$$\|\Gamma\| \times \|TA\| \xrightarrow{?} T(\|\Gamma\| \times \|A\|) \xrightarrow{\|\Gamma, x:A \vdash N:TB\|^*} \|TB\|$$

## Strong monad

#### A solution

We enforce the existence of a natural transformation t from pairs (value, computation) to computations of pairs.

$$\|\Gamma\| \times \|\mathit{TA}\| \xrightarrow[t_{\|\Gamma\|, \|\mathit{TA}\|}]{} \mathit{T}(\|\Gamma\| \times \|A\|)$$

This is given under the assumption that the corresponding monad is strong.

## Strong monad

#### Definition

A strong monad on a category  $\mathcal C$  is a tuple  $< T, \eta, \mu, t>$  where  $< T, \eta, \mu>$  is a monad and t is a natural transformation  $t_{A,B}A\times TB \to T(A\times B)$  such that the following equations hold in  $\mathcal C$ .

$$T(r_{A}) \circ t_{1} = r_{TA}$$

$$T(\alpha_{A}, B, C) \circ t_{A \times B, C} = t_{A, B \times C} \circ (Id_{A} \times t_{B, C} \circ \alpha_{A, B, TC})$$

$$t_{A, B} \circ (A \times \eta_{B}) = \eta_{A \times B}$$

$$t_{A, B} \circ (Id_{A} \times \mu_{B}) = \mu_{A \times B} \circ T(t_{A, B}) \circ t_{A, TB}$$

Where  $r_A: (1 \times A) \to A$  and  $\alpha_{A,B,C}: (A \times B) \times C \to A \times (B \times C)$  are natural isomorphisms.

See Moggi 1989 for more details and motivation.

#### Interpretation of let

By type derivation of  $\Gamma \vdash let \ x = M \ in \ N$  we can assume morphisms  $\|\Gamma \vdash M : TA\|$  and  $\|\Gamma, x : A \vdash N : TB\|$ .

$$\|\Gamma \vdash let \ x = M \ in \ N\| := \|\Gamma, x : A \vdash N : TB\|^* \circ$$

$$t_{\|\Gamma\|, \|A\|} \circ < ld_{\|\Gamma\|}, \|\Gamma \vdash M : TA\| >$$

$$\|\Gamma\| \xrightarrow{\|\Gamma \vdash M:TA\|} \|TA\|$$

$$\downarrow < Id_{\|\Gamma\|}, \|\Gamma \vdash M:TA\| >$$

$$\|\Gamma\| \times \|A\|$$

$$\downarrow \|TA\| \xrightarrow{t_{\|\Gamma\|, \|A\|}} T(\|\Gamma\| \times \|A\|)$$

$$\|\Gamma\| \times \|TA\| \xrightarrow{t_{\|\Gamma\|, \|A\|}} T(\|\Gamma\| \times \|A\|)$$

$$\|TA\| \times \|TA\| \xrightarrow{t_{\|\Gamma\|, \|A\|}} T(\|\Gamma\| \times \|A\|)$$

# Computational effects

#### Overview

- ► TA = A + E programs with exceptions
- $TA = A + \{\bot\}$  possibly non-terminating programs
- $TA = \mathcal{P}_{fin}A$  non-deterministic programs
- $TA = (A \times S)^S$  imperative programs
- $TA = A \times N$  programs with timers
- $TA = R^{R^A}$  programs with a continuation

## Identity monad

#### Identity monad

 $\text{A Kleisli triple} < \textit{Id}, \eta, \cdot^{\star} > \text{where } \eta \; \coloneqq \; \{\textit{Id}_{\textit{A}}\}_{\textit{A} \in \mathcal{C}_{\textit{Ob}}} \; \text{and} \; \textit{f}^{\star} \; \coloneqq \; \textit{f}.$ 

$$\|\Gamma \vdash val_{ld}M\| = \|\Gamma \vdash M\|$$
  
$$\|\Gamma \vdash let_{ld} x = M \text{ in } N\| = \|\Gamma \vdash N[x := M]\|$$

## **Exception monad**

#### **Exception monad**

A Kleisli triple  $< T, \eta, \cdot^* >$  where we fix an 'exception' object E and let

$$T:A\mapsto A+E$$

$$\eta := \{ \iota_A : A \to A + E \}_{A \in \mathcal{C}_{OD}}$$

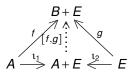
$$(f^* : (A + E) \to (B + E))(x \in A + E) := \begin{cases} f(x) & \text{if } x \in A \\ x & \text{Otherwise } x \in E \end{cases}$$

$$A \stackrel{\iota_1}{\longrightarrow} A + E \stackrel{\iota_2}{\longleftarrow} E$$

## Exception monad

#### **Exception monad**

A Kleisli triple  $\langle T, \eta, \cdot^* \rangle$  where we fix an 'exception' object *E* 



$$\begin{split} \|\Gamma \vdash val_{exc}M\| &= \iota_1 \circ \|\Gamma \vdash M\| \\ \|\Gamma \vdash let_{exc} x = M \ in \ N\| &= \left[\|\Gamma \vdash N[x \coloneqq M]\|, \iota_2\right] \end{split}$$

#### State monad

#### State monad

A Kleisli triple  $\langle T, \eta, \cdot^* \rangle$  where we fix a 'state' object S and let

$$T: A \mapsto (A \times S)^{S}$$

$$\eta := \{ \eta_{A} : a \mapsto (g_{a} : s \mapsto (a, s)) \}_{A \in \mathcal{C}_{Ob}}$$

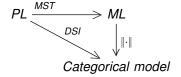
$$(f^{*} : (A \times S)^{S} \to (B \times S)^{S}) (g_{a} : S \to (A \times S)) (s : S) :=$$

$$f(\pi_{1}(g_{a}(s)) : A) (\pi_{2}(g_{a}(s)) : S) : (B \times S)$$

# Monadic semantics of programming languages

## The monadic metalanguage as a 'compiled language'

(Simplified picture)



#### Example:

• CPS transformation can be seen as a monadic style transformation for the call-by-value  $\lambda$ -calculus where T is the continuation monad.

## A toy programming language

#### PL - syntax

PL is a call-by-value programming language consisting of a signature  $\Sigma$  of base types  $\tau_1, \ldots, \tau_k$  and commands of the form  $p: \tau_i \to \tau_j$ . Programs e over  $\Sigma$  are defined by the following BNF

$$e := x | p(e) | \mu(e) | vale | let x = e_1 in e_2$$

and the following typing rules.

$$\frac{\Gamma \times \tau \vdash x : \tau}{\Gamma \times \tau \vdash x : \tau} \qquad \frac{\Gamma \vdash e : \tau_1}{\Gamma \vdash \rho(e) : \tau_2} \ \rho : \tau_1 \to \tau_2$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash vale : T\tau} \quad \frac{\Gamma \vdash \mu e : T\tau}{\Gamma \vdash e : \tau} \quad \frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash let \ x = e_1 \ in \ e_2 : \tau_2}$$

**Remark** Just as for the definition of *ML*, for the sake of simplicity we disregard the equational part of *PL*.

## The monadic-style translation

#### The monadic-style translation

Consider the metalanguage ML with the signature  $\Sigma^{\circ}$  including the same base types of  $\Sigma$  and a function  $p:\tau_1\to T\tau_2$  for every command  $p:\tau_1\to\tau_2$  in  $\Sigma$ . The translation from programs over  $\Sigma$  to terms over  $\Sigma^{\circ}$  is defined as follows.

$$x^{\circ}$$
 :=  $valx$   
 $(p(e))^{\circ}$  :=  $let \ x = e^{\circ} \ in \ p(x)$   
 $(vale)^{\circ}$  :=  $vale^{\circ}$   
 $(\mu e)^{\circ}$  :=  $let \ x = e^{\circ} \ in \ x$   
 $(let \ x = e_1 \ in \ e_2)^{\circ}$  :=  $let \ x = e_1^{\circ} \ in \ e_2^{\circ}$ 

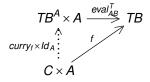
## Direct-style interpretation

#### The idea

- 'Interpretation without the compilation step'
- ▶ Model: not C but the Kleisli Category  $C_T$ .

#### Models of PL

A model of PL is a category C with finite products, a strong monad T and T-exponentials, that is, for any A,  $B \in C_{Ob}$ , an exponential  $(TB)^A$  of A and TB.



## Direct-style interpretation (sketch)

- Interpretation of types.
  - Objects in  $C_{TOb} = C_{Ob}$
  - A functional type  $\tau_1 \to \tau_2$  is interpreted as a T-exponential  $(T\|\tau_2\|)^{\|\tau_1\|}$
- Interpretations of terms.
  - Morphisms in C<sub>T</sub>
- Relation with the monadic-style translation

Given 
$$\Gamma \vdash_{pl} M : A$$
 in PL and  $\Gamma \vdash_{ML} M^{\circ} : TA$  in ML we have that  $\|\Gamma \vdash_{pl} M : A\| = \|\Gamma \vdash_{ML} M^{\circ} : TA\|$ .

## The monadic-style translation

#### Advantages

- The monadic metalanguage is an interface hiding the categorical model.
- The same metalanguage can describe any computational effect given as a (strong) monad.
- The monadic-style translation can be flexibly (and monotonically) adapted for source programming languages with different features.