



# Quantitative Global Memory



Sandra Alves <sup>1</sup> Delia Kesner <sup>2</sup>

Miguel Ramos

WoLLIC 23



<sup>1</sup>CRACS/INESC-TEC, DCC, Faculdade de Ciências, Universidade do Porto <sup>2</sup>IRIF, CNRS, Université Paris Cité & Institut Universitaire de France <sup>3</sup>LIACC, DCC, Faculdade de Ciências, Universidade do Porto





#### **Programming Languages**



#### λ-calculus (Pure)

- Simple structure
- No side-effects
- Easy to reason about
- Useless for programmers(?)

#### Real (Impure)

- Complicated structure
- Side-effects
- Hard to reason about
- Interact with the real world







#### **Programming Languages**



Is the  $\lambda$ -calculus *useless* for programmers?

[The correspondence] reduces the problem of specifying ALGOL 60 semantics

to that of specifying the semantics of a structurally simpler language.

Peter Landin

in "Correspondence between ALGOL 60 and Church's Lambda-notation: part I"

How can we add effects to pure languages?

[W]e distinguish the object A of values (of type A) from the object TA of computations (of type A).

Eugenio Moggi

in "Notions of Computation and Monads"





#### **Global State**



#### Moggi's CBV Encoding

Let *S* be the type of states.

Then  $TA = S \gg (A \times S)$ :

$$v \rightsquigarrow \lambda s.(v,s)$$

$$t \ u \ \ \, \rightsquigarrow \ \ \, \lambda s. let (u', s') = u \ s$$

$$in (t \ u') \ s'$$

#### **Effect Operations**

Let  $\ell$  be a state location:

Retrieving a value:

$$\mathsf{get}_\ell(\lambda x.t)$$

Setting a value:

 $\mathsf{set}_\ell(\mathit{v},\mathit{t})$ 





## Intersection Types



Extension of simple types with type constructor ∩

if  $\tau, \sigma$  are types, then  $\tau \cap \sigma$  is a type

Originally enjoy associativity, commutativity and idempotency

$$(\tau \cap \sigma) \cap \theta = \tau \cap (\sigma \cap \theta)$$
  $(\tau \cap \sigma) = (\sigma \cap \tau)$ 

• Express models capturing qualitative computational properties

"t is terminating iff t is typable"





## Non-Idempotent Intersection Types



- Intersection types that do not enjoy idempotency  $(\tau \cap \tau) \neq \tau$
- Express models capturing upper bound quantitative computational properties

"t is terminating in at most X steps iff t is typable with a derivation of size X"  $\downarrow$ 

evaluation length + size of normal form

Size explosion

$$t_0 := y$$
 $t_n := (\lambda x.xx)t_{n-1}$ 
 $t_n \rightarrow_{\beta}^n y^{2^n}$ 

linear in  $n$ 





### Split and Exact Measures



• To obtain split measures

counters in judgments + tight constants + persistent typing rules

(evaluation length, size of normal form)

To obtain exact measures.

tight derivations = minimal derivations

• Obtain models capturing exact quantitative computational properties

"t is terminating in exactly X steps with normal form of size Y iff t is typable with counter (X, Y)"





#### Quantitative Global Memory



Goal

To build a quantitative model (expressed as a tight type system) that captures exact quantitative properties of a  $\lambda$ -calculus with operations that interact with a global state.





# Syntax



```
Values v, w ::= x \mid \lambda x.t

Terms t, u ::= v \mid xt \mid \text{get}_{\ell}(\lambda x.t) \mid \text{set}_{\ell}(v, t)

States s, q ::= \epsilon \mid \text{upd}_{\ell}(v, s)

Configurations c ::= (t, s)
```





# **Operational Semantics**



$$\frac{(t,s) \to_{\mathbf{r}} (u,q) \quad \mathbf{r} \in \{\beta_{v}, \mathbf{g}, \mathbf{s}\}}{(vt,s) \to_{\mathbf{r}} (vu,q)}$$

$$\frac{s \equiv \mathrm{upd}_{\ell}(v,q)}{(\mathrm{get}_{\ell}(\lambda x.t),s) \to_{\mathbf{g}} (t\{x \setminus v\},s)}$$

$$\frac{(t,s) \to_{\mathbf{r}} (u,q) \quad \mathbf{r} \in \{\beta_{v}, \mathbf{g}, \mathbf{s}\}}{(vt,s) \to_{\mathbf{r}} (vu,q)}$$

$$\frac{s \equiv \mathrm{upd}_{\ell}(v,q)}{(\mathrm{get}_{\ell}(\lambda x.t),s) \to_{\mathbf{g}} (t\{x \setminus v\},s)}$$

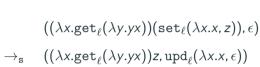
$$\frac{(t,s) \to_{\mathbf{r}} (u,q) \quad \mathbf{r} \in \{\beta_{v}, \mathbf{g}, \mathbf{s}\}}{(vt,s) \to_{\mathbf{r}} (vu,q)}$$



Weak reduction: we do not reduce inside abstractions We allow open normal forms Size of normal forms is "number of applications"







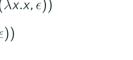
$$ightarrow_{\mathrm{g}} \quad ((\lambda x.x)z, \mathrm{upd}_{\ell}(\lambda x.x, \epsilon))$$
 $ightarrow_{\beta_{\nu}} \quad (z, \mathrm{upd}_{\ell}(\lambda x.x, \epsilon))$ 



Quantitative Global Memory

$$ightarrow_{\mathsf{g}} \quad ((\lambda x.x)z, \operatorname{upd}_{\ell}(\lambda x.x, \epsilon)) \\ 
ightarrow_{\beta_{\mathsf{v}}} \quad (z, \operatorname{upd}_{\ell}(\lambda x.x, \epsilon))$$

Operational Semantics & Example













# **Encoding Arrow Types**



$$\underbrace{A \Rightarrow B}_{\text{IL}} \overset{\text{Girard's CBV}}{\leadsto} \underbrace{!A \multimap !B}_{\text{ILL}} \overset{\text{Moggi's CBV}}{\leadsto} !A \multimap T(!B)$$

- IA is an intersection of value types
  - $A = [A_1, \dots, A_n]$
- T is the global state monad

$$TA = S \gg (A \times S)$$

!A) is a computation wrapping an intersection of value types

$$T[A_1,\ldots,A_n]=S\gg ([A_1,\ldots,A_n]\times S)$$





## **Types**



• Values and Neutral Forms

Tight Constants tt ::= 
$$\mathbf{v} \mid \mathbf{a} \mid \mathbf{n}$$

Value Types  $\sigma$  ::=  $\mathbf{v} \mid \mathbf{a} \mid \mathcal{M} \mid \mathcal{M} \Rightarrow \delta$ 

Multi-types  $\mathcal{M}$  ::=  $[\sigma_i]_{i \in I}$  where  $I$  is a finite set

• States, Configurations, and Computations

State Types 
$$\mathcal{S}$$
 ::=  $\{\ell_i : \mathcal{M}_i\}_{i \in I}$  where all  $\ell_i$  are distinct Configuration Types  $\kappa$  ::=  $\tau \times \mathcal{S}$ 

Monadic Types  $\delta$  ::=  $\mathcal{S} \gg \kappa$ 





# **Typing**



Judgments are decorated with counters

# 
$$\beta$$
-steps | normal form |   
( b , m , d )   
# memory accesses

• We have three different kinds of typing judgments

$$\begin{array}{c} \text{computations} \\ \hline \Gamma \vdash^{(b,m,d)} t : \delta \end{array} \qquad \begin{array}{c} \text{states} \\ \hline \Delta \vdash^{(b,m,d)} s : \mathcal{S} \end{array} \qquad \begin{array}{c} \hline \Gamma \vdash^{(b,m,d)} (t,s) : \kappa \end{array}$$

- Some typing rules have two (or more) different versions
  - Consuming: increase only b and m counters
  - *Persistent*: increase the *d* counter



Quantitative Global Memory



# (Some) Typing Rules



$$\frac{\Gamma \vdash^{(b,m,d)} v : \mathcal{M}}{\Gamma \vdash^{(b,m,d)} v : \mathcal{S} \gg (\mathcal{M} \times \mathcal{S})} \stackrel{(\uparrow)}{\longrightarrow} \frac{(\Gamma_i \vdash^{(b_i,m_i,d_i)} v : \sigma_i)_{i \in I}}{+_{i \in I} \Gamma_i \vdash^{(+_{i \in I}b_i,+_{i \in I}m_i,+_{i \in I}d_i)} v : [\sigma_i]_{i \in I}}$$
(m)

$$\frac{\Gamma \vdash (b,m,d) v : \mathcal{M} \Rightarrow (\widetilde{\mathcal{S}_m} \gg (\tau \times \mathcal{S}_f)) \qquad \Delta \vdash (b',m',d') t : \widetilde{\mathcal{S}_i} \gg (\mathcal{M} \times \widetilde{\mathcal{S}_m})}{\Gamma + \Delta \vdash (1+b+b',m+m',d+d') vt : \mathcal{S}_i \gg (\tau \times \mathcal{S}_f)}$$
(@)

$$\frac{\Gamma \vdash^{(b,m,d)} v : \mathcal{M} \quad \Delta \vdash^{(b',m',d')} t : \{(\ell : \mathcal{M})\}; \mathcal{S} \gg \kappa}{\Gamma + \Delta \vdash^{(b+b',1+m+m',d+d')} \mathtt{set}_{\ell}(v,t) : \mathcal{S} \gg \kappa} \ (\mathtt{set})$$



Ouantitative Global Memory

of 20



# Exact Measures (Wrong)



Why do we need tightness and persistent typing rules?

Let 
$$\sigma = [v] \Rightarrow (S \gg (\tau \times S'))$$
.

	$y: [v] \vdash^{(0,0,0)} y: v $ (m)
$\frac{1}{(2\pi)^{1/2}} (ax)$	$y: [v] \vdash^{(0,0,0)} y: [v] $ (†)
$x: [\sigma] \vdash^{(0,0,0)} x: [v] \Rightarrow (\mathcal{S} \gg (\tau \times \mathcal{S}'))$ $x: [\sigma], y: [v] \vdash^{(\blacksquare,0,0)} xy: \sigma$	$y: [v] \vdash^{(0,0,0)} y: \mathcal{S} \gg ([v] \times \mathcal{S}) \tag{0}$



$$(\underbrace{xy}, s) \not\rightarrow \text{ for any}$$





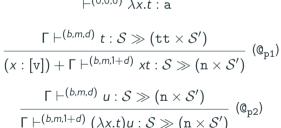
# Typing Rules A Persistent



$$\frac{\Gamma \vdash^{(b,m,d)} v : v/a}{\Gamma \vdash^{(b,m,d)} v : S \gg (v/a \times S)}$$

$$\frac{}{\vdash^{(0,0,0)} \lambda x.t : a}$$

$$(\uparrow)$$







# Exact Measures (Correct)



```
\frac{y:[a]\vdash^{(0,0,0)}y:a}{y:[a]\vdash^{(0,0,0)}y:\emptyset\gg(a\times\emptyset)}\stackrel{(\uparrow)}{(\uparrow)}{x:[v],y:[a]\vdash^{(\mathbf{Q},0,\mathbf{Q})}xy:\emptyset\gg(n\times\emptyset)}
```



 $(xy,s) \not\to \text{ for any } s$ 





#### Validity of the Model



# Soundness

If 
$$\Phi \triangleright \Gamma \vdash \stackrel{\text{(b.m.d)}}{}(t,s) : \kappa \text{ tight,}$$

$$\exists (u,q) \text{ s.t. } u \in \text{no, } (t,s) \rightarrow \stackrel{\text{(b.m)}}{}(u,q), \text{ and } |(u,q)| = d$$

#### Completeness

If  $(t,s) \rightarrow (b,m)$  (u,q) s.t.  $u \in no$ ,  $\exists \Phi \triangleright \Gamma \vdash (b,m,|(u,q)|) (t,s) : \kappa \text{ tight.}$ 





#### Typing Example



Consider the term exemplifying the operational semantics:

$$((\lambda x. \mathtt{get}_{\ell}(\lambda y. yx))(\mathtt{set}_{\ell}(\lambda x. x, z)), \epsilon) \rightarrow (2.2) \underbrace{|z| = 0}_{\mathsf{z}}, \mathtt{upd}_{\ell}(\lambda x. x, \epsilon))$$

We can build the following tight derivation:

$$\frac{\Phi \quad \Psi}{z: [v] \vdash^{(2,2,0)} (\lambda x. get_{I}(\lambda y. yx))(set_{I}(I, z)): \emptyset \gg (v \times \emptyset)} \stackrel{\text{(@)}}{} \frac{-(0,0,0) \cdot \epsilon: \emptyset}{(conf)}$$

$$z: [v] \vdash^{(2,2,0)} ((\lambda x. get_{I}(\lambda y. yx))(set_{I}(I, z)), \epsilon): v \times \emptyset$$





#### Conclusion



#### Summary

- Simple language with global memory
- Following a weak (open) CBV strategy
- Provided a quantitative model capturing exact measures

#### Future Work

- Different effects: exceptions, I/O, non-determinism, ...
- Different Strategies: CBV (unrestricted), CBN, CBNeed, ...
- Unifying frameworks: λ!-calculus, CBPV, EE-calculus, ...



