Quantitative Global Memory

Sandra Alves [1] Delia Kesner [2] Miguel Ramos [3]

WoLLIC 23

[1]CRACS/INESC-TEC, DCC, Faculdade de Ciências, Universidade do Porto
[2]IRIF, CNRS, Université Paris Cité & Institut Universitaire de France
[3]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 λ-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 $TA = S \gg (A \times S)$:

$$
\begin{aligned}
v &\leadsto \lambda s.(v, s) \\
t\ u &\leadsto \lambda s.\text{let } (u', s') = u\ s \\
&\qquad \text{in } (t\ u')\ s'
\end{aligned}
$$

Effect Operations

Let ℓ be a state location:

- Retrieving a value:

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

- Setting a value:

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

Intersection Types

- Extension of simple types with type constructor $\cap$

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

- Originally enjoy associativity, commutativity and idempotency

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

- 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

$$t_0 := y \qquad t_n := (\lambda x.xx)t_{n-1} \quad \rightsquigarrow \quad \underbrace{t_n}_{\text{linear in } n} \to_\beta^n \overbrace{y^{2^n}}^{\text{exponential in } n}$$

Split and Exact Measures

- To obtain split measures

$$\underbrace{\text{counters in judgments + tight constants + persistent typing rules}}_{\text{(evaluation length, size of normal form)}}$$

- 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
λ-calculus with operations that interact with a global state.

Syntax

$$
\begin{array}{rrcl}
\text{Values} & v, w & ::= & x \mid \lambda x.t \\
\text{Terms} & t, u & ::= & v \mid v\,t \mid \mathtt{get}_{\ell}(\lambda x.t) \mid \mathtt{set}_{\ell}(v, t) \\
\text{States} & s, q & ::= & \epsilon \mid \mathtt{upd}_{\ell}(v, s) \\
\text{Configurations} & c & ::= & (t, s)
\end{array}
$$

Operational Semantics

(Configurations)

$$\frac{}{((\lambda x.t)v, s) \rightarrow_{\beta_v} (t\{x \backslash v\}, s)} \qquad \frac{(t, s) \rightarrow_{\mathtt{r}} (u, q) \quad \mathtt{r} \in \{\beta_v, \mathtt{g}, \mathtt{s}\}}{(vt, s) \rightarrow_{\mathtt{r}} (vu, q)}$$

$$\frac{s \equiv \mathtt{upd}_\ell(v, q)}{(\mathtt{get}_\ell(\lambda x.t), s) \rightarrow_{\mathtt{g}} (t\{x \backslash v\}, s)} \qquad \frac{}{(\mathtt{set}_\ell(v, t), s) \rightarrow_{\mathtt{s}} (t, \mathtt{upd}_\ell(v, s))}$$

Weak reduction: we do not reduce inside abstractions

We allow *open* normal forms

Size of normal forms is “number of applications”

Operational Semantics ⚜ Example

$$
\begin{array}{ll}
 & ((\lambda x.\texttt{get}_\ell(\lambda y.yx))(\texttt{set}_\ell(\lambda x.x, z)), \epsilon) \\
\to_{\texttt{s}} & ((\lambda x.\texttt{get}_\ell(\lambda y.yx))z, \texttt{upd}_\ell(\lambda x.x, \epsilon)) \\
\to_{\beta_v} & (\texttt{get}_\ell(\lambda y.yz), \texttt{upd}_\ell(\lambda x.x, \epsilon)) \\
\to_{\texttt{g}} & ((\lambda x.x)z, \texttt{upd}_\ell(\lambda x.x, \epsilon)) \\
\to_{\beta_v} & (z, \texttt{upd}_\ell(\lambda x.x, \epsilon))
\end{array}
$$

$(0 \ \# \ \beta_v$-steps$, 0 \ \#$ memory accesses$)$

Encoding Arrow Types

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

- $!A$ is an intersection of value types

$$!A = [A_1, \dots, A_n]$$

- T is the global state monad

$$TA = \mathcal{S} \gg (A \times \mathcal{S})$$

- $T(!A)$ is a computation wrapping an intersection of value types

$$T[A_1, \dots, A_n] = \mathcal{S} \gg ([A_1, \dots, A_n] \times \mathcal{S})$$

Types

- Values and Neutral Forms

Tight Constants	$\mathtt{tt}$	$::=$	$\mathtt{v} \mid \mathtt{a} \mid \mathtt{n}$
Value Types	σ	$::=$	$\mathtt{v} \mid \mathtt{a} \mid \mathcal{M} \mid \mathcal{M} \Rightarrow \delta$
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 ℓ_i are distinct
Configuration Types	κ	$::=$	$\tau \times \mathcal{S}$
Monadic Types	δ	$::=$	$\mathcal{S} \gg \kappa$

Typing

- Judgments are decorated with counters

$$\overbrace{(\ b\ ,}^{\#\ \beta\text{-steps}}\ \underbrace{m\ ,\ \overbrace{d\ )}^{|\text{normal form}|}}_{\#\ \text{memory accesses}}$$

- We have three different kinds of typing judgments

$$\overbrace{\Gamma \vdash^{(b,m,d)} t : \delta}^{\textit{computations}} \qquad \overbrace{\Delta \vdash^{(b,m,d)} s : \mathcal{S}}^{\textit{states}} \qquad \overbrace{\Gamma \vdash^{(b,m,d)} (t, s) : \kappa}^{\textit{configurations}}$$

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

(Some) Typing Rules

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

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

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

Exact Measures (Wrong)

Why do we need *tightness* and *persistent* typing rules?

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

$$\dfrac{\dfrac{}{x : [\sigma] \vdash^{(0,0,0)} x : [\mathtt{v}] \Rightarrow (\mathcal{S} \gg (\tau \times \mathcal{S}'))}(\mathtt{ax}) \quad \dfrac{\dfrac{\dfrac{}{y : [\mathtt{v}] \vdash^{(0,0,0)} y : \mathtt{v}}(\mathtt{ax})}{y : [\mathtt{v}] \vdash^{(0,0,0)} y : [\mathtt{v}]}(\mathtt{m})}{y : [\mathtt{v}] \vdash^{(0,0,0)} y : \mathcal{S} \gg ([\mathtt{v}] \times \mathcal{S})}(\uparrow)}{x : [\sigma], y : [\mathtt{v}] \vdash^{(1,0,0)} xy : \mathcal{S} \gg (\tau \times \mathcal{S}')}(@)$$

$$(\underbrace{xy}_{|\mathrm{xy}| = 1}, s) \not\to \text{ for any } s$$

Typing Rules ⚜ Persistent

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

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

$$\frac{\Gamma \vdash^{(b,m,d)} t : \mathcal{S} \gg (\mathtt{tt} \times \mathcal{S}')}{(x : [\mathtt{v}]) + \Gamma \vdash^{(b,m,1+d)} xt : \mathcal{S} \gg (\mathtt{n} \times \mathcal{S}')}\ (@_{\mathtt{p1}})$$

$$\frac{\Gamma \vdash^{(b,m,d)} u : \mathcal{S} \gg (\mathtt{n} \times \mathcal{S}')}{\Gamma \vdash^{(b,m,1+d)} (\lambda x.t)u : \mathcal{S} \gg (\mathtt{n} \times \mathcal{S}')}\ (@_{\mathtt{p2}})$$

Exact Measures (Correct)

$$\dfrac{\dfrac{\dfrac{}{y : [\mathsf{a}] \vdash^{(0,0,0)} y : \mathtt{a}}\ (\mathtt{ax})}{y : [\mathsf{a}] \vdash^{(0,0,0)} y : \emptyset \gg (\mathtt{a} \times \emptyset)}\ (\uparrow)}{x : [\mathtt{v}], y : [\mathsf{a}] \vdash^{(\boxed{0},0,\boxed{1})} xy : \emptyset \gg (\mathtt{n} \times \emptyset)}\ (@_{\mathtt{p1}})$$

$$(\underbrace{xy}_{|\mathsf{xy}| = \boxed{1}}, s) \not\to \text{ for any } s$$

Validity of the Model

Soundness

$$\text{If } \Phi \triangleright \Gamma \vdash^{(b,m,d)} (t,s) : \kappa \text{ tight},$$
$$\exists (u,q) \text{ s.t. } u \in \texttt{no},\ (t,s) \twoheadrightarrow^{(b,m)} (u,q), \text{ and } |(u,q)| = d$$

Completeness

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

Typing Example

Consider the term exemplifying the operational semantics:

$$((\lambda x.\mathtt{get}_\ell(\lambda y.yx))(\mathtt{set}_\ell(\lambda x.x, z)), \epsilon) \twoheadrightarrow^{(2,2)} (\underbrace{z}_{|z| = 0}, \mathtt{upd}_\ell(\lambda x.x, \epsilon))$$

We can build the following tight derivation:

$$\dfrac{\dfrac{\Phi \quad \Psi}{z : [\mathtt{v}] \vdash^{(2,2,0)} (\lambda x.\mathtt{get}_l(\lambda y.yx))(\mathtt{set}_l(\mathtt{I}, z)) : \emptyset \gg (\mathtt{v} \times \emptyset)}(@) \quad \dfrac{}{\vdash^{(0,0,0)} \epsilon : \emptyset}(\mathtt{emp})}{z : [\mathtt{v}] \vdash^{(2,2,0)} ((\lambda x.\mathtt{get}_l(\lambda y.yx))(\mathtt{set}_l(\mathtt{I}, z)), \epsilon) : \mathtt{v} \times \emptyset}(\mathtt{conf})$$

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, ...

The End