Parsimonious Logic and Computational Complexity

Damiano Mazza
CNRS, UMR 7030, Laboratoire d'Informatique de Paris Nord
Université Paris 13, Sorbonne Paris Cité

Quantitative Semantics for Logic and Computation
Marseille, 2–3 September 2016

The Parsimonious Trinity

multiplicative affine logic +

$$\frac{\Gamma \vdash A}{!\Gamma \vdash\, !A} \qquad \frac{\Gamma, A, !A \vdash C}{\Gamma, !A \vdash C} \qquad \frac{\Gamma \vdash A \quad \Delta \vdash\, !A}{\Gamma, \Delta \vdash\, !A}$$

SMCC with terminal unit
+ monoidal endofunctor $!(-)$
+ natural iso $!A \cong A \otimes !A$

Affine calculus with streams

$$\begin{aligned} t, u ::= {} & a \mid \lambda a.t \mid tu \mid t \otimes u \mid t[a \otimes b := u] \\ \mid {} & x_i \mid !t \mid t :: u \mid t[!x := u] \end{aligned}$$

The infinitary affine λ-calculus

- Linear logic (or its affine variant) has two layers:

  multiplicative: SMCC + terminal unit; affine λ-terms;
  exponential: allows to recover intuitionistic logic/λ-calculus.

- Idea: the exponential modality is a limit [Méllìès et al.], [EhrhardRegnier].

- Infinitary affine terms: $t, u ::= x_i \mid \lambda x.t \mid t\langle u_0, u_1, u_2, \ldots\rangle$

  1. finite terms are dense: $t = \sup \lfloor t \rfloor_n$, with $\lfloor t \rfloor_n$ the "truncation" of t;
  2. reduction is continuous.

- The usual λ-calculus embeds via Girard's translation

$$[\![MN]\!] := [\![M]\!]_0 \langle [\![N]\!]_1, [\![N]\!]_2, [\![N]\!]_3, \ldots\rangle$$

Church Meets Cook and Levin

- Cook-Levin theorem: "computation is local".
- Idea: local = continuous:

$$M\underline{x} \rightarrow^{l(|x|)} \underline{b} \quad \Longrightarrow \quad \exists m(l).\ \lfloor [\![M]\!] \rfloor_{m(l)}\, \underline{x} \rightarrow^{*} \underline{b}$$

- In Turing machines, $m = O(l^2)$. In the λ-calculus, $m = O(2^l)$!
- This is related to size explosion: $M \rightarrow^{\Theta(n)} N_n$ with $|N_n| = \Theta(2^n|M|)$.

Enter parsimony

- Parsimonious logic has an alternative exponential layer:

$$A, B ::= X \mid 1 \mid A \otimes B \mid A \multimap B \mid !A \qquad \textit{Milner's law: } !A \cong A \otimes !A$$

$$\frac{\Gamma \vdash A}{!\Gamma \vdash !A}\,! \qquad \frac{\Gamma, A, !A \vdash C}{\Gamma, !A \vdash C}\,\text{abs} \qquad \frac{\Gamma \vdash A \quad \Delta \vdash !A}{\Gamma, \Delta \vdash !A}\,\text{coabs}$$

- The parsimonious λ-calculus:

$$M, N ::= a \mid \lambda a.M \mid MN \mid M \otimes N \mid M[a \otimes b := N] \qquad \text{(multiplicative)}$$

$$\mid x_i \mid !M \mid M :: N \mid M[!x := N] \;+\; \text{constraints} \qquad \text{(exponential)}$$

- Affine access to finite initial segments of ultimately constant streams.

What parsimony can and cannot do

- Some examples/non examples:
  - $\lambda a.(x_0 \otimes !x_1)[!x := a] : !A \multimap A \otimes !A$ (Milner's law)
  - $\lambda p.(a :: b)[a \otimes b := p] : A \otimes !A \multimap !A$ (Milner's law^{-1})
  - $\lambda a.\langle x_0, x_2, \ldots\rangle \otimes \langle x_1, x_3, \ldots\rangle[!x := a] : !A \multimap !A \otimes !A$ (contraction)
  - $\lambda a.\langle x_1, x_3, x_5, \ldots\rangle[!x := a] : !A \multimap !A$
  - $\Delta := \lambda a.(x_0 !x_1)[!x := a] \qquad \Delta!\Delta \to^* \Delta!\Delta$
  - No general fixpoint combinator. Only linear fixpoints.

- The untyped parsimonious λ-calculus is Turing-complete.

- It fixes the Church/Cook-Levin disagreement; no size explosion.

- For complexity reasons (*cf.* a few slides below), a translation like Girard's $\mathbf{LJ} \to \mathbf{LL}$ cannot exist for parsimonious logic.

Parsimonious programming

- Parsimonious PCF allows only linear fixpoints:

$$\mathsf{Y} : !(!A \multimap A) \multimap A \qquad \text{vs.} \qquad \mathsf{Y}_\ell : !(A \multimap A) \multimap A$$

- In parsimonious Ocaml: `let rec f x =` Φ (`f` appears once in Φ).

- Quite annoying if you want to recurse on trees. . .

- Question: is there a *compilation* PCF $\rightarrow$ parsimonious PCF?

- Possible answer: $\mathsf{Y}f : A \rightarrow B \mapsto \mathsf{Y}_\ell f' : A \otimes B \otimes \mathsf{Bool} \otimes S_{A,B} \multimap B$ such that $(\mathsf{Y}f')(a, -, \uparrow, \epsilon) = (\mathsf{Y}f)(a)$.

Example: the Fibonacci numbers

```
fib_aux(n,m,d,s) :=
if (d = - and isEmpty(s)) then m
else let (n',m',d',s') =
        if (d = +) then
                if (n=0 or n=1) then (?,1,-,s)
                else (n-1,?,+,(n,*)::s)
        else
                let (a,b)::r = s in
                if (b=*) then (a-2,?,+,(a,m)::r)
                else (?,m+b,-,r)
in fib_aux(n',m',d',s')

fib(n) := fib_aux(n,?,+,.)
```

- Is parsimonious logic the “logic of `while` loops”?

Implicit computational complexity

- What languages may be decided in parsimonious Str $\multimap$ Bool?

types	**predicates**	**in the λ-calculus**
untyped	RE	RE
polymorphic	Conjecture: PR?	PA_2
linear polymorphic	P	??
simple	L	$\subsetneq$ LINTIME

- Quite different from usual linear logic ICC ("light logics"): global (light logics) vs. local (parsimony) complexity of cut-elimination.

- Novel perspective: non-uniform computation (P/poly, L/poly).

Intersection types and approximations

intersection type derivation = simply-typed approximation

$$A, B ::= \alpha \mid A \otimes B \mid A \multimap B \mid \langle A_0, \ldots, A_{n-1} \rangle$$

$$t, u ::= \text{(mult)} \mid x_i \mid !\bot \mid t :: u \mid t[!x := u]$$

$$M, N ::= \text{(mult)} \mid x_i \mid !M \mid M :: N \mid M[!x := N]$$

$$\frac{}{\mathbf{\Gamma}, x : \mathbf{A}; \Delta \vdash x_i \sqsubset x_i : \mathbf{A}(i)} \qquad \frac{\mathrm{fv}(M) \subseteq \overline{x}}{\overline{x} : \mathbf{\Gamma}; \vdash !\bot \sqsubset !M : \langle\rangle}$$

$$\frac{\mathbf{\Gamma}; \Delta \vdash t \sqsubset M : A \quad \mathbf{\Gamma}; \Delta' \vdash u \sqsubset !M^{++} : \mathbf{B}}{\mathbf{\Gamma}; \Delta, \Delta' \vdash t :: u \sqsubset !M : A :: \mathbf{B}}$$

$$\frac{\mathbf{\Gamma}; \Delta' \vdash u \sqsubset N : \mathbf{A} \quad \mathbf{\Gamma}, x : \mathbf{A}; \Delta \vdash t \sqsubset M : C}{\mathbf{\Gamma}; \Delta, \Delta' \vdash t[!x := u] \sqsubset M[!x := N] : C}$$

Excursus: approximations and resource λ-terms

- Consider the subcalculus of approximations

$$t, u ::= x_i \mid \lambda x.t \mid t\langle u_1, \ldots, u_n \rangle$$

- Erase indices and ordering in sequences:

$$t, u ::= x \mid \lambda x.t \mid t[u_1, \ldots, u_n]$$

- Looks familiar?

affine approximations	$\Leftrightarrow$	resource λ-terms
rigid intersection types	$\Leftrightarrow$	non-idempotent intersection types
$t \sqsubset M$	$\Leftrightarrow$	$t^- \in \mathrm{Taylor}(M)$

Time is Size, Space is Depth

Theorem. *Let*

$$\vdash t_n \sqsubset M : \mathsf{Str}_n \multimap \mathsf{Bool}$$

where Str_n *are "*n*-linearizations" of* $(!(\alpha \multimap \alpha))^{\otimes 2} \multimap \alpha \multimap \alpha$*, of depth* $d(n)$*. Then,*

$$\mathrm{lang}(M) \in \mathsf{TIME}(O(|t_n|)) \cap \mathsf{SPACE}(O(d(n) \log |t_n|)).$$

PROOF. For time, use rewriting; for space, use the GoI. □

- HO circuit $= t : \mathsf{Str}[] \multimap \mathsf{Bool}$, size $= |t|$, depth $= \mathrm{depth}(\mathsf{Str}[])$.

- Consistent with Terui, 2004; similar to Borodin 1977:

$$\mathsf{DEPTH/SIZE}(d, s) \subseteq \mathsf{TIME}(O(s)) \cap \mathsf{SPACE}(O(d + \log s))$$

Questions and perspectives

- Is parsimonious system F exactly primitive recursive?
- Is the compilation PCF $\rightarrow$ parsimonious PCF canonical?
- Denotational semantics: a concrete *strict* model?
- Small circuit classes? (Or: dropping higher order)
- A more abstract view of the Cook-Levin theorem?