A Complete Polynomial λ-Calculus

joint work with Simona Ronchi Della Rocca

Erika De Benedetti

Università degli Studi di Torino

ICTCS (Varese), 20 September 2012

Introduction

- **ICC**: Implicit Computational Complexity
- The problem: the design of programming languages with bounded computational complexity, in particular `PTIME`.
- The line: a **ML-like** approach
  - λ-calculus as paradigmatic programming language.
  - types as semantic properties of terms.
  - Type assignment for λ-calculus such that:
    - types guarantee the correctness of terms, in particular their complexity bound.

Our ancestors

Two similar proposals, based on Light Logics (derived from the Linear Logic of Girard) where the cut-elimination procedure is bounded in time by the size of the proof, exploiting the isomorphism:

FORMULAE as **TYPES**

- `DLAL` (Baillot, Terui): Types are formulae of `LAL` (Light Affine Logic) (Girard, Asperti, Roversi)
- `STA` (Gaboardi, RDR): Types are formulae of `SOFT` (Soft Linear Logic) (Lafont)

Both:

- are sound and complete w.r.t. `FPTIME` i.e.:
  - all and only polynomial functions can be coded by typed terms (using a standard uniform coding)
- give type to a proper subset of strongly normalizing terms.

A system with stratification types: ISTA

- We explore a different (non logical) approach, where types are **stratified**, so equivalent to intersection types with idempotence and commutativity, but without associativity.
- Our system ISTA is:
  - sound and complete w.r.t. FPTIME
  - sound and complete w.r.t. strong normalization

Definitions

The set of types of ISTA is defined by the following syntax:

- $\mathtt{A} ::= \mathtt{a} \mid \sigma \to \mathtt{A} \mid \forall \mathtt{a}.\mathtt{A}$ (linear types)
- $\sigma ::= \mathtt{A} \mid \{\sigma_1, ..., \sigma_n\} \quad n > 0$ (stratified types)
  where
  - $\{\sigma_1, ..., \sigma_n\}$ is treated as a set
  - $\{^j \mathtt{A}\}^j$ stands for $\underbrace{\{...\{}_{j} \ \mathtt{A} \ \underbrace{\}...\}}_{j}$
  - brackets { } are not associative
  - $\bar{\sigma} = \{\mathtt{A}_1, ..., \mathtt{A}_n\}$
- ISTA proves judgments of the shape $\pi : \Gamma \vdash \mathtt{M} : \sigma$

Typing rules for ISTA

$$\frac{}{\mathtt{x} : \mathtt{A} \vdash \mathtt{x} : \mathtt{A}}\ (Ax)$$

$$\frac{\Gamma \vdash \mathtt{M} : \sigma \quad \mathtt{x} \notin \mathrm{dom}(\Gamma)}{\Gamma, \mathtt{x} : \mathtt{A} \vdash \mathtt{M} : \sigma}\ (w) \qquad \frac{\Gamma, \mathtt{x} : \sigma \vdash \mathtt{M} : \mathtt{B}}{\Gamma \vdash \lambda \mathtt{x}.\mathtt{M} : \sigma \to \mathtt{B}}\ (\to I)$$

$$\frac{\Gamma_1 \vdash \mathtt{M} : \sigma \to \mathtt{A} \quad \Gamma_2 \vdash \mathtt{N} : \sigma \quad \Gamma_1 \# \Gamma_2}{\Gamma_1, \Gamma_2 \vdash \mathtt{MN} : \mathtt{A}}\ (\to E)$$

$$\frac{\Gamma \vdash \mathtt{M} : \mathtt{A} \quad \mathtt{a} \notin \mathtt{FTV}(\Gamma)}{\Gamma \vdash \mathtt{M} : \forall \mathtt{a}.\mathtt{A}}\ (\forall I) \qquad \frac{\Gamma \vdash \mathtt{M} : \forall \mathtt{a}.\mathtt{B}}{\Gamma \vdash \mathtt{M} : \mathtt{B}[\mathtt{A}/\mathtt{a}]}\ (\forall E)$$

$$\frac{\Gamma_i \vdash \mathtt{M} : \sigma_i \quad 1 \leq i \leq n}{\bigcup_{i=1}^{n} \{\Gamma_i\} \vdash \mathtt{M} : \{\sigma_1, ..., \sigma_n\}}\ (st) \qquad \frac{\Gamma, \mathtt{x}_1 : \mathtt{A}, ..., \mathtt{x}_n : \mathtt{A} \vdash \mathtt{M} : \tau}{\Gamma, \mathtt{x} : \{\mathtt{A}\} \vdash \mathtt{M}[\mathtt{x}/\mathtt{x}_1, ..., \mathtt{x}/\mathtt{x}_n] : \tau}\ (m_l)$$

$$\frac{\Gamma, \mathtt{x}_1 : \sigma_1, ..., \mathtt{x}_n : \sigma_n \vdash \mathtt{M} : \tau \quad \sigma_i \text{ not linear}}{\Gamma, \mathtt{x} : \{\sigma_1, ..., \sigma_n\} \vdash \mathtt{M}[\mathtt{x}/\mathtt{x}_1, ..., \mathtt{x}/\mathtt{x}_n] : \tau}\ (m_s)$$

Normalization bound

- The depth of a derivation π, denoted by $d(\pi)$, is the maximum nesting of applications of rule (st)

Normalization bound

$\pi : \Gamma \vdash \mathtt{M} : \sigma$, and $\mathtt{M} \xrightarrow[\beta]{*} \mathtt{nf(M)}$ in m steps, then $m \leq |\mathtt{M}|^{\mathrm{d}(\pi)+1}$, where $|\mathtt{M}|$ is the size of $\mathtt{M}$ and $d(\pi)$ is the depth of the derivation π.

Note that

- A typable term has infinite types, so infinite derivations. Every derivation gives a bound of its normalization time
- A typable term has a minimal derivation. So the minimal bound depends only on the term itself.

Strong normalization

ISTA is sound and complete w.r.t. strong normalization:

- if a term is typable in ISTA then it is strongly normalizing (soundness)

  Proof. Termination of reduction does not depend on the strategy

- all strongly normalizing terms can be typed in ISTA (completeness)

  Proof. Following the guidelines of Neergaard[1]:

  1. Longest reduction strategy
  2. Subject expansion under perpetual strategy
  3. Completeness follows as a corollary

[1] Neergaard 2005

Typability w.r.t. System F

- With respect to system **F**:
  - More terms can be typed:

    e.g. $\vdash (\lambda \mathtt{xy.y(x}I)(\mathtt{x}K))\Delta : ((\mathtt{A} \to \mathtt{A}) \to \mathtt{B} \to \mathtt{C}) \to \mathtt{C}$

    $I = \lambda x.x, \quad K = \lambda xy.x, \quad \Delta = \lambda x.xx$

  - More types can be assigned:

    e.g. $\vdash \lambda \mathtt{x.xx} : \forall \vec{a}.\{\{\sigma\}, \{\sigma \to \mathtt{B}\}\} \to \mathtt{B}$, $\vec{a} \subseteq FV(\sigma) \cup FV(B)$

- Stratification does not limit the number of terms that can be typed, but is necessary in order to limit the complexity.

Soft Type Assignment system (STA)

- To prove soundness and completeness w.r.t FPTIME, we use the corresponding property of STA[2].
- We use the same technique of Bucciarelli and Piperno, when proving that adding intersection to simple types does not increase the definability of functions.
- The set of types of STA is a proper subset of the SOFT formulae and is defined by the following syntax:
  - $U ::= a \mid \mu \to U \mid \forall a.U$ (linear types)
  - $\mu ::= U \mid !\mu$ (modal types)
- restrictions with respect to SOFT: modalities are not allowed neither on the r.h.s. of an arrow nor under the universal quantifier.

[2] Gaboardi-Ronchi 2007

Reminder: STA

$$\frac{}{\mathtt{x} : \mathtt{U} \vdash_{\mathtt{STA}} \mathtt{x} : \mathtt{U}}\ (Ax) \qquad \frac{\Theta \vdash_{\mathtt{STA}} \mathtt{M} : \mu \quad \mathtt{x} \notin dom\Theta}{\Theta, \mathtt{x} : \mathtt{U} \vdash_{\mathtt{STA}} \mathtt{M} : \mu}\ (w)$$

$$\frac{\Theta, \mathtt{x} : \mu \vdash_{\mathtt{STA}} \mathtt{M} : \mathtt{U}}{\Theta \vdash_{\mathtt{STA}} \lambda \mathtt{x}.\mathtt{M} : \mu \to \mathtt{U}}\ (\to I) \qquad \frac{\Theta \vdash_{\mathtt{STA}} \mathtt{M} : \mu \to \mathtt{U} \quad \Xi \vdash_{\mathtt{STA}} \mathtt{N} : \mu \quad \Theta \# \Xi}{\Theta, \Xi \vdash_{\mathtt{STA}} \mathtt{MN} : \mathtt{U}}\ (\to E)$$

$$\frac{\Theta \vdash_{\mathtt{STA}} \mathtt{M} : \mathtt{U} \quad \mathtt{a} \notin FV(\Theta)}{\Theta \vdash_{\mathtt{STA}} \mathtt{M} : \forall \mathtt{a}.\mathtt{U}}\ (\forall I) \qquad \frac{\Theta \vdash_{\mathtt{STA}} \mathtt{M} : \forall \mathtt{a}.\mathtt{B}}{\Theta \vdash_{\mathtt{STA}} \mathtt{M} : \mathtt{B}[\mathtt{U}/\mathtt{a}]}\ (\forall E)$$

$$\frac{\Theta, \mathtt{x}_1 : \mu, \ldots, \mathtt{x}_n : \mu \vdash_{\mathtt{STA}} \mathtt{M} : \nu}{\Theta, \mathtt{x} : !\mu \vdash_{\mathtt{STA}} \mathtt{M}[\mathtt{x}/\mathtt{x}_1, ..., \mathtt{x}_n] : \nu}\ (m) \qquad \frac{\Theta \vdash \mathtt{M} : \mu}{!\Theta \vdash_{\mathtt{STA}} \mathtt{M} : !\mu}\ (sp)$$

Representation of numerals

- We represent numerals in Church style, so m is represented by $\underline{m} = \lambda \mathbf{xy}.\, \underbrace{\mathbf{x}(\mathbf{x}...(\mathbf{x}}_{m}\, \mathbf{y})...))$
- In STA, $\underline{m}$ can be typed either

  uniformely by $\mathbf{N} = \forall \mathrm{a}.!(\mathrm{a} \to \mathrm{a}) \to \mathrm{a} \to \mathrm{a}$, or

  by $\mathbf{N}_n = \forall \mathrm{a}.!^n(\mathrm{a} \to \mathrm{a}) \to \mathrm{a} \to \mathrm{a}$, for all $n \geq 1$
- In ISTA, $\underline{m}$ can be typed either

  uniformely by $\mathbf{NI} = \forall \mathrm{a}.\{\mathrm{a} \to \mathrm{a}\} \to \mathrm{a} \to \mathrm{a}$, or

  by $\mathbf{NI}_n = \forall \mathrm{a}.\{^n\mathrm{a} \to \mathrm{a}\}^n \to \mathrm{a} \to \mathrm{a}$, for all $n \geq 1$
- Both in ISTA and in STA derivations for numerals have depth 0.

Representation of functions in STA

Let $\phi : \mathcal{N}^p \longrightarrow \mathcal{N}$ be a function of arity p.

- $\mathtt{M}$ represents ϕ in STA iff
  1. $\mathtt{M}\underline{n_1}...\underline{n_p} = \underline{\phi(n_1, ..., n_p)}$
  2. there are integers $i_1, ..., i_p$ such that $\mathbf{x}_1 :!^{i_1} \mathbf{N}_{j_1}, ..., \mathbf{x}_p :!^{i_p} \mathbf{N}_{j_p} \vdash_{\mathtt{STA}} \mathtt{M}\mathbf{x}_1...\mathbf{x}_p : \mathbf{N}_j$, for some j_h, j, $1 \leq h \leq p$

FPTIME characterization

All and only the polynomial functions can be coded in STA.

Representation of functions in STA

Example:

- succ: $\mathbf{N}_i \to \mathbf{N}_{i+1}$
- add: $\mathbf{N}_i \to \mathbf{N}_j \to \mathbf{N}_{max(i,j)+1}$
- mult: $\mathbf{N}_i \to !^i\mathbf{N}_j \to \mathbf{N}_{i+j}$ for all $i, j \geq 1$

So they cannot be iterated, but only composed in order to obtain polynomials.

Representation of functions in ISTA

Let $\phi : \mathcal{N}^p \longrightarrow \mathcal{N}$ be a function of arity p.

- M represents ϕ in ISTA iff
  1. $\mathtt{M}\underline{n_1}...\underline{n_p} = \underline{\phi(n_1, ..., n_p)}$
  2. $\mathbf{x}_1 : \sigma_1, ..., \mathbf{x}_p : \sigma_p \vdash \mathtt{M}\mathbf{x}_1...\mathbf{x}_p : \mathbf{NI}_j$, and for all $i \in \{1, ..., p\}$, either $\sigma_i = \mathbf{NI}_k$ or $\bar{\sigma}_i = \{\mathbf{NI}_{i_1}, ...\mathbf{NI}_{i_m}\}$ for some k, i_h, j, $1 \leq h \leq p$, $m > 0$

FPTIME completeness

- Translation $(.)^\circ$ from $\mathcal{TS}$ to $\mathcal{T}$ is defined as follows:

$$(\mathtt{a})^\circ = \mathtt{a}; \quad (\mu \to \mathtt{U})^\circ = (\mu)^\circ \to (\mathtt{U})^\circ; \quad (!\mu)^\circ = \{(\mu)^\circ\}$$

- Translation extended to derivations:

$$(\pi : \Theta \vdash_{\mathtt{STA}} \mathtt{M} : \mu)^\circ = (\pi)^\circ : (\Theta)^\circ \vdash \mathtt{M} : (\mu)^\circ$$

such that:

  - the subject is preserved
  - the depth is preserved

FPTIME completeness

In ISTA all the polynomial functions are definable.

From ISTA to STA

$(.)^*$ is a translation from ISTA-derivations to STA-derivations such that:

- $$(\pi : \Gamma \vdash M : A)^* = (\pi)^* : \Theta \vdash_{STA} \lceil M \rceil : (A)^*$$

  such that

  - $(A)^*$ is a linear type;
  - $\lceil M \rceil$ is a partial linearization of M.
  - the depth is preserved

From ISTA to STA(2)

Example 1

Let $\mathtt{M} = \lambda \mathtt{x}.\mathtt{xx}$:

- $\pi :\vdash \mathtt{M} : \{\{\mathtt{b} \to \mathtt{a}\}, \{\mathtt{b}\}\} \to \mathtt{a}$
- $(\pi)^* \vdash_{\mathtt{STA}} \lambda \mathtt{x}_1.\lambda \mathtt{x}_2.\mathtt{x}_1\mathtt{x}_2 :!!(\mathtt{b} \to \mathtt{a}) \to !!\mathtt{b} \to \mathtt{a}$

Example 2

Let $\mathtt{N} = \lambda \mathtt{x}.\lambda \mathtt{y}.\mathtt{yxx}$:

- $\pi :\vdash \mathtt{N} : \{\mathtt{a}\} \to (\mathtt{a} \to \mathtt{a} \to \mathtt{a}) \to \mathtt{a}$
- $(\pi)^* :\vdash_{\mathtt{STA}} \mathtt{N} :!\mathtt{a} \to (\mathtt{a} \to \mathtt{a} \to \mathtt{a}) \to \mathtt{a}$

Functions representation

For simplicity, let us consider unary functions.

- Let $\mathtt{M}$ represent in ISTA a program computing a function $\phi : \mathcal{N} \longrightarrow \mathcal{N}$, i.e.,

$$\pi :\vdash \mathtt{M} : \sigma \to \mathbf{NI}_j$$

  where the components of σ are $\mathbf{NI}_{r_1}, ..., \mathbf{NI}_{r_m}$ $(1 \leq i \leq m)$

- Let the corresponding derivation in STA be

$$(\pi)^* :\vdash_{\mathtt{STA}} \lceil \mathtt{M} \rceil :!^{i_1}\mathbf{N}_{j_1} \to ... \to !^{i_m}\mathbf{N}_{j_m} \to \mathbf{N}_j$$

- $\lceil \mathtt{M} \rceil$ represents the function ψ, such that

$$\phi(n) = \psi(\underbrace{n, ..., n}_{m})$$

Conservativity: proof

We prove that there is a term in STA computing ϕ: its normal form is

$$\vdash_{\mathtt{STA}} \lambda \mathtt{x.x}II(\lambda \mathtt{z.z}\underbrace{(\lambda \mathtt{zw.xzw})...(\lambda \mathtt{zw.xzw})}_{m})\lceil \mathtt{M} \rceil : !^{k}\mathbf{N} \to \mathbf{N}_{j}$$

for any $\mathbf{k} \geq \max\{\mathbf{i_1}, ..., \mathbf{i_m}\} + \mathbf{2}$

In the proof we exploit two properties of STA:

1. $\Gamma, x :!^{i}\nu \vdash_{\mathtt{STA}} \mathtt{M} : \mu$ implies $\Gamma, x :!^{j}\nu \vdash_{\mathtt{STA}} \mathtt{M} : \mu$, for every $j \geq i$
   (by a sequence of applications of rule (m));
2. there is a coercion $C^{i}_{j} = \lambda \mathtt{xzw.xzw}$ from any $\mathbf{N}_i$ to $\mathbf{N}_j$, $(j \leq i)$:

$$\vdash_{\mathtt{STA}} C^{i}_{j} : \mathbf{N}_i \to \mathbf{N}_j$$

- In fact, $(\lambda \mathtt{xzw.xzw})\underline{n} \xrightarrow[\beta]{*} \underline{n}$

PTIME soundness

FPTIME soundness

All the functions definable in ISTA are polynomial

proof

Let $\pi :\vdash \mathtt{M} : \sigma \to \mathbf{NI}_j$, where the components of σ are $\mathbf{NI}_{r_1}, ..., \mathbf{NI}_{r_m}$ and let $(\pi)^* :\vdash_{\mathtt{STA}} \lceil \mathtt{M} \rceil : \mu = !^{i_1}\mathbf{N}_{j_1} \to ... \to !^{i_m}\mathbf{N}_{j_m} \to \mathbf{N}_j$. Then

$$\vdash_{\mathtt{STA}} \lambda \mathtt{x}.\mathtt{x}II(\lambda \mathtt{z}.\mathtt{z}\underbrace{(\lambda \mathtt{zw}.\mathtt{xzw})...(\lambda \mathtt{zw}.\mathtt{xzw})}_{m})\lceil \mathtt{M} \rceil : !^k\mathbf{N} \to \mathbf{N}_j$$

and

$$(\lambda \mathtt{x}.\mathtt{x}II(\lambda \mathtt{z}.\mathtt{z}\underbrace{(\lambda \mathtt{zw}.\mathtt{xzw})...(\lambda \mathtt{zw}.\mathtt{xzw})}_{m})\lceil \mathtt{M} \rceil)\underline{n} \underset{\beta}{\longrightarrow} \lceil \mathtt{M} \rceil \underbrace{\underline{n}...\underline{n}}_{m}$$

in a number of steps linear in m. m depends only on M, so is independent from the input.

Espressivity

- In ISTA it is possible to type higher order functionals such as

$$\mathtt{ITER}^3\ \underline{\mathtt{succ}}\ \underline{\mathtt{n}}$$

which is a more natural way of writing a program rather than

$$\underline{\mathtt{succ}}\ (\underline{\mathtt{succ}}\ (\underline{\mathtt{succ}}\ \underline{\mathtt{n}}))$$

- $\vdash \mathtt{ITER}^3 : \{\{\mathbf{NI}_i \to \mathbf{NI}_{i+1}\}, \{\mathbf{NI}_{i+1} \to \mathbf{NI}_{i+2}\}, \{\mathbf{NI}_{i+2} \to \mathbf{NI}_{i+3}\}\} \to \mathbf{NI}_i \to \mathbf{NI}_{i+3}$, so the term is not typed in STA

Iteration of $\underline{\texttt{mult}}$

- On the other hand, in ISTA terms such as

$$\texttt{ITER}^3\ \underline{\texttt{mult}}\ \underline{n}\ \underline{m}$$

  cannot be typed

- In fact

$$\texttt{ITER}^3 : \forall a_1.\forall a_2.\forall a_3.\forall a_4.\{\{a_1 \to a_2\},\{a_2 \to a_3\},\{a_3 \to a_4\}\} \to a_1 \to a_4$$

- Since $\underline{\texttt{mult}} : \mathbf{NI}_i \to \{{}^i\mathbf{NI}_j\}^i \to \mathbf{NI}_{i+j}$, the presence of the stratified type prevents $\texttt{ITER}^3$ from being typed

Conclusions

- ISTA is sound and complete w.r.t. FPTIME
- ISTA is sound and complete w.r.t. strong normalization
- In ISTA all numerical polynomial algorithms that can be expressed by a strongly normalizing term are typed
- Adding stratification types to STA types does not increase the definability of polynomial functions, but adds expressivity to the language