Characterizing polynomial and exponential complexity classes in elementary λ-calculus

Erika De Benedetti

Università di Torino - ENS de Lyon

joint work with

Patrick Baillot Simona Ronchi Della Rocca

TCS 2014 (Rome)
2 September 2014

Introduction

- ICC: capturing complexity classes by programming languages or calculi, without relying on a given computational model
- done through *linear logic* (subsystems) using the proofs-as-programs approach
- two different lines:
  - *monovalent*: one language $\leftrightarrow$ one complexity class
  - *polyvalent*: one language $\leftrightarrow$ **several** complexity classes
- stratification: a key concept in implicit complexity

Elementary linear logic

- elementary linear logic (ELL): born as a monovalent characterization of elementary complexity (2_k^n for any $k \in \mathbb{N}$) [Girard98]
- more recently shown to give a *polyvalent* characterization of PTIME and the family k-EXP [B.2011], **but**...

  relies on proof-nets and does not give characterization of functions

My talk:

- get similar (better?) results for a cheaper price
- recast !-modality stratification in λ-calculus
- introduce two alternative characterizations for functions

Roadmap

Untyped $\lambda^!$-calculus

- terms:

$$\mathtt{M}, \mathtt{N} ::= \mathtt{x} \mid \lambda \mathtt{x}.\mathtt{M} \mid \lambda^! \mathtt{x}.\mathtt{M} \mid \mathtt{MN} \mid !\mathtt{M}$$

  write $!^k\mathtt{M} = \underbrace{!\ldots!}_{k}\mathtt{M}$

- *depth* $\delta(\mathtt{N}, \mathtt{M})$ *of a subterm* $\mathtt{N}$ *in* $\mathtt{M}$:
  number of nested ! enclosing $\mathtt{N}$.

- *depth* $\delta(\mathtt{M})$ *of a term* $\mathtt{M}$:
  maximal depth of all its subterms.

  example: $\mathtt{M} = !((\lambda \mathtt{x}.\mathtt{x})\ !!\mathtt{y})$
  so $\delta(\lambda \mathtt{x}.\mathtt{x}, \mathtt{M}) = 1$, $\delta(\mathtt{y}, \mathtt{M}) = 3$ and $\delta(\mathtt{M}) = 3$

Reduction

- reduction rules:

$$\begin{aligned} (\lambda \mathtt{x}.\mathtt{M})\mathtt{N} &\to \mathtt{M}[\mathtt{N}/\mathtt{x}] && (\beta\text{-redex}) \\ (\lambda^{!}\mathtt{x}.\mathtt{M})!\mathtt{N} &\to \mathtt{M}[\mathtt{N}/\mathtt{x}] && (!\text{-redex}) \end{aligned}$$

- the reduction is confluent
- intuition behind the abstractions:
  - $\lambda \mathtt{x}.\mathtt{M}$ expects input from depth 0
  - $\lambda^{!}\mathtt{x}.\mathtt{M}$ expects input from depth 1

Well-formedness

- *well-formed terms*:
  $\mathtt{M}$ such that, for any sub-term which is an abstraction,
  1. $\lambda\mathtt{x}.\mathtt{N}$ implies $\mathtt{x}$ occurs in $\mathtt{N}$ at most *once* at depth 0
  2. $\lambda^!\mathtt{x}.\mathtt{N}$ implies $\mathtt{x}$ occurs in $\mathtt{N}$ (possibly) many times at depth 1

example:

$$\lambda f.\lambda x.(f\ (f\ x)) \qquad \text{not w.f.}$$

$$\lambda^! f.!(\lambda x.(f\ (f\ x)) \qquad \text{w.f.}$$

Well-formedness

- *well-formed terms*:
  $\mathtt{M}$ such that, for any sub-term which is an abstraction,
  1. $\lambda\mathtt{x}.\mathtt{N}$ implies $\mathtt{x}$ occurs in $\mathtt{N}$ at most *once* at depth 0
  2. $\lambda^{!}\mathtt{x}.\mathtt{N}$ implies $\mathtt{x}$ occurs in $\mathtt{N}$ (possibly) many times at depth 1
- class of well-formed terms is preserved by reduction
- intuitions behind well-formed terms:
  1. no duplication at depth 0 (linearity)
  2. argument of program remains at same depth

see also [MadetPhDThesis2012]

Depth and reduction

let us consider the class of well-formed terms

Property

If $\mathtt{M} \to \mathtt{M}'$, *then* $\delta(\mathtt{M}') \leq \delta(\mathtt{M})$.

i.e. depth of w.f. terms does not increase ...

let $\mathtt{M} \in \mathbf{nf}_i$, for $i \in \mathbb{N}$, if $\mathtt{M}$ has no redexes at depth $\leq i$

Reduction strategy

Lemma

If $\mathtt{M} \in \mathbf{nf}_i$ *and* $\mathtt{M} \to \mathtt{M}'$ *by reducing a redex at depth* $i+1$*, then* $\mathtt{M}' \in \mathbf{nf}_i$*.*

let $\delta = \delta(\mathtt{M})$:

Level-by-level reduction strategy: if $\mathtt{M} \notin \mathbf{nf}$, then reduce (non deterministically) a redex at depth i, where i is the least depth such that $\mathtt{M} \notin \mathbf{nf}_i$

... then the resulting term is in normal form iff it is in $\mathbf{nf}_\delta$

Data

- Booleans: $\mathtt{TT} = \lambda \mathtt{x}.\lambda \mathtt{y}.\mathtt{x} \qquad \mathtt{FF} = \lambda \mathtt{x}.\lambda \mathtt{y}.\mathtt{y}$
- Church tally integers: $\underline{n} = \lambda^{!} f.!(\lambda x.f^n x)$

  in ordinary λ-calculus: $\lambda f.\lambda x.f^n x$

- Church binary words: $\underline{w} = \lambda^{!} f_0.\lambda^{!} f_1 !.(\lambda x.f_{i_1}\ (f_{i_2}\ ...(f_{i_n}\ x)...))$ if w is the word $\langle i_1 \cdots i_n \rangle, n \in \mathbb{N}$

  in ordinary λ-calculus: $\lambda f_0.\lambda f_1.\lambda x.f_{i_1}\ (f_{i_2}\ ...(f_{i_n}\ x)...)$

- Scott words:

$$\begin{array}{rcl} \widehat{i \cdot w} & = & \lambda f_0.\lambda f_1.\lambda x.f_i\ \widehat{w} \quad (i = 0, 1) \\ \widehat{\epsilon} & = & \lambda f_0.\lambda f_1.\lambda x.x \end{array}$$

Representing functions

- let $\mathtt{P}$ be a closed well-formed term in normal form
  let $\underline{w}$ be a Church binary word and $n = |\underline{w}|$
- we consider $\mathtt{P}!\underline{w}$ because we want to be able to duplicate the argument
- $\mathtt{P}$ *represents* $f : \{0,1\}^\star \to \{0,1\}^\star$ if there is $k \in \mathbb{N}$ such that

$$\mathtt{P}!\underline{w} \to !^k\overline{f(w)},$$

for any word w and encoding of binary word $\overline{w}$

Complexity bounds

We examine the reduction of $\mathtt{P}!\underline{w}$ up to a given depth i:

Property

The term $\mathtt{P}!\underline{w}$ reduces by the level-by-level strategy to a 2-normal form in $poly(n)$ number of steps.

Denote $2_0^x = x$ and $2_{i+1}^x = 2^{2_i^x}$.

Property

For any $i \geq 2$ the term $\mathtt{P}!\underline{w}$ reduces by the level-by-level strategy to a i-normal form in $2_{i-2}^{poly(n)}$ number of steps.

but the i-normal form is not necessarily a data...

A typed $\lambda^!$-calculus

- we use *types* to ensure that the program will eventually yield a data
- type language:

$$\begin{array}{lll} \mathtt{A}, \mathtt{B} & ::= \mathtt{a} \mid \sigma \multimap \sigma \mid \forall \mathtt{a}.\mathtt{A} \mid \mu \mathtt{a}.\mathtt{A} & \text{(linear types)} \\ \sigma, \tau & ::= \mathtt{A} \mid !\sigma & \text{(types)} \end{array}$$

where

$\forall \mathtt{a}.\mathtt{A}$	polymorphic quantification
$\mu \mathtt{a}.\mathtt{A}$	type fixpoint

we write $!^k\sigma$ for $!...!\sigma$ (k repetitions)

Type system

- typing judgement:

$$\Gamma \mid \Delta \mid \Theta \vdash \mathtt{M} : \sigma$$

where

Γ: only linear types A	$\rightsquigarrow$ occurrences at depth 0
Δ: only !-types $!\sigma$	$\rightsquigarrow$ occurrences at depth 1
Θ: *parking*	$\rightsquigarrow$ occurrences at depth 0
	... but *eventually* will be at depth 1

type system adapted from [CDLR05]

- a term $\mathtt{M}$ is well-typed iff there is $\Gamma \mid \Delta \mid \emptyset \vdash \mathtt{M} : \sigma$

Typing rules

$$\frac{}{\Gamma, \mathtt{x} : \mathtt{A} \mid \Delta \mid \Theta \vdash \mathtt{x} : \mathtt{A}}\ (Ax^L) \qquad \frac{}{\Gamma \mid \Delta \mid \mathtt{x} : \sigma, \Theta \vdash \mathtt{x} : \sigma}\ (Ax^P)$$

$$\frac{\Gamma, \mathtt{x} : \mathtt{A} \mid \Delta \mid \Theta \vdash \mathtt{M} : \tau}{\Gamma \mid \Delta \mid \Theta \vdash \lambda \mathtt{x}.\mathtt{M} : \mathtt{A} \multimap \tau}\ (\multimap I^L) \qquad \frac{\Gamma \mid \Delta, \mathtt{x} :!\sigma \mid \Theta \vdash \mathtt{M} : \tau}{\Gamma \mid \Delta \mid \Theta \vdash \lambda^! \mathtt{x}.\mathtt{M} :!\sigma \multimap \tau}\ (\multimap I^I)$$

$$\frac{\Gamma_1 \mid \Delta \mid \Theta \vdash \mathtt{M} : \sigma \multimap \tau \quad \Gamma_2 \mid \Delta \mid \Theta \vdash \mathtt{N} : \sigma \quad \Gamma_1 \# \Gamma_2}{\Gamma_1, \Gamma_2 \mid \Delta \mid \Theta \vdash \mathtt{MN} : \tau}\ (\multimap E)$$

$$\frac{\emptyset \mid \emptyset \mid \Theta' \vdash \mathtt{M} : \sigma}{\Gamma \mid\ !\Theta', \Delta \mid \Theta \vdash\ !\mathtt{M} :!\sigma}\ (!)$$

$+\forall$ and μ rules

Properties

- the system enjoys subject-reduction
- if `M` is well-typed then `M` is well-formed

Datatypes

$$
\begin{array}{lcl}
\mathrm{B} & = & \forall \mathtt{a}.\mathtt{a} \multimap \mathtt{a} \multimap \mathtt{a} \\
\mathrm{N} & = & \forall \mathtt{a}.!(\mathtt{a} \multimap \mathtt{a}) \multimap !(\mathtt{a} \multimap \mathtt{a}) \\
\mathrm{W} & = & \forall \mathtt{a}.!(\mathtt{a} \multimap \mathtt{a}) \multimap !(\mathtt{a} \multimap \mathtt{a}) \multimap !(\mathtt{a} \multimap \mathtt{a}) \\
\mathrm{W}_S & = & \mu \mathtt{b}.\forall \mathtt{a}.(\mathtt{b} \multimap \mathtt{a}) \multimap (\mathtt{b} \multimap \mathtt{a}) \multimap (\mathtt{a} \multimap \mathtt{a})
\end{array}
$$

$$
\begin{array}{lcl}
\mathtt{TT} & = & \lambda x.\lambda y.x, \qquad \mathtt{FF} = \lambda \mathtt{x}.\lambda \mathtt{y}.\mathtt{y} \\
\underline{n} & = & \lambda^{!} f.\lambda x.(f\ (f\ \ldots (f\ x)\ldots) \\
\underline{w} & = & \lambda^{!} f_0.\lambda^{!} f_1 !(\lambda x.(f_{i_1}\ (f_{i_2}\ \ldots (f_{i_n}\ x)\ldots) \\
\widehat{i \cdot w} & = & \lambda f_0.\lambda f_1.\lambda x.f_i\ \widehat{w}_{(i=0,1)} \qquad \widehat{\epsilon} = \lambda f_0.\lambda f_1.\lambda x.x
\end{array}
$$

observe that booleans and Scott words have depth 0
while Church integers and words have depth 1

Properties of datatypes

Property (Reading prop. of Booleans)

If $\vdash \mathtt{M} : !^k\mathrm{B}$ *for* $k \geq 0$ *and* $\mathtt{M} \in \mathbf{nf}_k$, *then* $\mathtt{M} =_! {}^k\mathtt{TT}$ *or* $\mathtt{M} =_! {}^k\mathtt{FF}$.

reading property of datatype: we can extract the result by reducing up to depth k

Theorem (Complexity soundness)

Let $\vdash \mathtt{P} : !\mathrm{W} \multimap !^{k+2}\mathrm{B}$, *where* $\mathtt{P}$ *is a program, and let* $\vdash \underline{w} : \mathrm{W}$ *of length* n.
Then:
$\mathtt{P}!\underline{w}$ *reduces to either* $!^{k+2}\mathtt{TT}$ *or* $!^{k+2}\mathtt{FF}$ *in time* $2_k^{poly(n)}$.

Expressiveness

Complexity classes:

- $\mathtt{DTIME}(F(n))$: *predicates* on binary words computable in time $O(F(n))$ on a deterministic TM.
- $\mathtt{FDTIME}(F(n))$: *functions* on binary words defined in the same way.

$$\begin{aligned} \mathtt{k\text{-}EXP} &= \mathtt{DTIME}(2_k^{poly(n)}) \quad \text{, for } k \geq 0 \\ \mathtt{k\text{-}FEXP} &= \mathtt{FDTIME}(2_k^{poly(n)}) \quad \text{, for } k \geq 0 \end{aligned}$$

so for $k = 0$, $\mathtt{0\text{-}EXP} = \mathtt{PTIME}$ and $\mathtt{0\text{-}FEXP} = \mathtt{FPTIME}$

Lemma (Extensional completeness)

Let $k \geq 0$ and f be a predicate in $\mathtt{k\text{-}EXP}$; then there is a term $\mathtt{M}$ representing f such that $\vdash \mathtt{M} : !\mathrm{W} \multimap !^{k+2}\mathrm{B}$.

Characterization of predicates

simulation of a TM running for $2_k^{p(n)}$ steps:
quite standard, but to obtain the right type in the end we need an encoding of words **of depth 0** $\Rightarrow$ *Scott words*

Theorem

Let $k \in \mathbb{N}$. The functions represented by closed terms of type $!\mathrm{W} \multimap !^{k+2}\mathrm{B}$ are exactly the predicates of $\mathtt{k\text{-}EXP}$.

Which output type to characterize FPTIME functions?

- we showed that $!\mathrm{W} \multimap !^2\mathrm{B} = \mathtt{PTIME}$

  what about functions?

- with $!\mathrm{W} \multimap !^2\mathrm{W}$ we can also type exponential functions
- with $!\mathrm{W} \multimap !\mathrm{W}$ we do not achieve completeness

problem: Church word is of depth 1, but we need a datatype of depth 0...

$\rightsquigarrow$ use Scott binary word, i.e. $!\mathrm{W} \multimap !^2\mathrm{W}_S$

Capturing functions classes

the type W_S allows us to extend the previous theorem from predicates to functions:

Property (Reading prop. of Scott words)

Assume $\vdash \mathtt{M} : !^k \mathrm{W}_S$ *for* $k \geq 0$ *and* $\mathtt{M} \in \mathbf{nf}_k$*, then* $\mathtt{M} =_! !^k \widehat{w}$ *for some* $w \in \{0,1\}^\star$*.*

Theorem

Let $k \in \mathbb{N}$*. The functions represented by closed terms of type* $!\mathrm{W} \multimap !^{k+2}\mathrm{W}_S$ *are exactly the functions of* $\mathtt{k\text{-}FEXP}$*.*

Compositionality of FPTIME

- FPTIME is closed by composition, **but**...
  in $!\mathrm{W} \multimap !^2\mathrm{W}_S$ input/output types do not match!
- as we saw before, $!\mathrm{W} \multimap !^i\mathrm{W}_{i=1,2}$ does not work...
- we wish to represent a word $w' \in \{0,1\}^\star$ by a pair $\langle n, w\rangle$, where $n \in \mathbb{N}, w \in \{0,1\}^\star$ and

$$w' = \begin{cases} w, & \text{if } |w| \leq n \\ \text{prefix of } w \text{ of length } n, & \text{otherwise} \end{cases}$$

A new datatype to characterize k-FEXP

- consider the $\otimes$ operator on types: how to type $\langle n, w \rangle$?

  - $\mathrm{N} \otimes \mathrm{W}_S$: not expressive enough (duplication of integer)
  - $!\mathrm{N} \otimes !\mathrm{W}_S$: iteration of the word?

  $\Rightarrow$ we use $!\mathrm{N} \otimes !^2\mathrm{W}_S$

- output: combined type $!^{k+1}\mathrm{N} \otimes !^{k+2}\mathrm{W}_S$, for $k \geq 0$

- syntactic sugar on terms:

$$\mathtt{M} \otimes \mathtt{N} = \lambda \mathtt{x}.\mathtt{x}\mathtt{M}\mathtt{N} \qquad \lambda^!(\mathtt{y} \otimes \mathtt{z}).\mathtt{M} = \lambda \mathtt{x}.\mathtt{x}\lambda^!\mathtt{y}.\lambda^!\mathtt{z}.\mathtt{M}$$

- we take a pair $!^{k+1}\underline{n} \otimes !^{k+2}\widehat{w}$ to represent $\langle n, w \rangle$

Properties of the datatype

Property (Reading prop. of pairs)

Assume $\vdash \mathtt{M} : !^{k+1}\mathrm{N} \otimes !^{k+2}\mathrm{W}_S$ *for* $k \geq 0$ *and* $\mathtt{M} \in \mathbf{nf}_{k+2}$, *then* $\mathtt{M} =_{!^{k+1}} \underline{n} \otimes !^{k+2}\widehat{w}$ *for some* $n \in \mathbb{N}, w \in \{0,1\}^\star$

i.e. typing ensures it is sufficient to reduce until depth $k+2$ to obtain the result

Theorem (Complexity soundness)

Let $\vdash \mathtt{P} : !\mathrm{N} \otimes !^2\mathrm{W}_S \multimap !^{k+1}\mathrm{N} \otimes !^{k+2}\mathrm{W}_S$, *where* $\mathtt{P}$ *is a program, and let* $\vdash !\underline{n} \otimes !^2\widehat{w} : !\mathrm{N} \otimes !^2\mathrm{W}_S$ *s.t.* $|w| \leq n$.
Then:
$\vdash \mathtt{P}(!\underline{n} \otimes !^2\widehat{w}) : !^{k+1}\mathrm{N} \otimes !^{k+2}\mathrm{W}_S$ *can be computed in time* $2_k^{poly(n)}$.

Expressiveness and characterization

Lemma (Extensional completeness)

Let $k \geq 0$ and f be a function in k-$\mathtt{FEXP}$; then there is a term $\mathtt{M}$ representing f such that $\vdash \mathtt{M} : !\mathrm{N} \otimes !^2\mathrm{W}_S \multimap !^{k+1}\mathrm{N} \otimes !^{k+2}\mathrm{W}_S$.

which leads to the following

Theorem

Let $k \in \mathbb{N}$. The functions represented by closed terms of type $!\mathrm{N} \otimes !^2\mathrm{W}_S \multimap !^{k+1}\mathrm{N} \otimes !^{k+2}\mathrm{W}_S$ are exactly the functions of k-$\mathtt{FEXP}$.

Some observations

- we can compose terms of type $(!\mathrm{N} \otimes !^2\mathrm{W}_S) \multimap (!\mathrm{N} \otimes !^2\mathrm{W}_S)$, i.e. $\mathtt{FPTIME}$ is closed by composition
- we can compose terms representing $f \in \mathtt{FPTIME}$ and $g \in k\text{-}\mathtt{FEXP}$, i.e. $g \circ f \in k\text{-}\mathtt{FEXP}$
- if $k \geq 1$ the class $k\text{-}\mathtt{FEXP}$ is not closed by composition, **but**... the typing shows that $f_1 \in k_1\text{-}\mathtt{FEXP}$ and $f_2 \in k_2\text{-}\mathtt{FEXP}$ imply $f_2 \circ f_1 \in (k_1 + k_2)\text{-}\mathtt{FEXP}$

Conclusions

- a simple framework for ! stratification in λ-calculus
- degrees of the stratification exactly match degrees of the exponential time hierarchy
- distinction between 2 main ingredients at work: (syntactic) ! stratification and typing
- accounts for both predicate and function complexity classes
- composite datatype seems promising for other characterizations (ex. P-bounded-EXP, NP using polynomial witness, ...)