

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

DICE 2014 (Grenoble)

5 april 2014

Introduction

- aim: capture complexity classes by calculi
- two different approaches:
 - *monovalent*: one language \leftrightarrow one complexity class
 - *polyvalent*: one language \leftrightarrow **several** complexity classes
e.g. [Jones01], [Leivant02] for family k -EXP, $k \in \mathbb{N}$
- stratification: a key concept in implicit complexity

Elementary linear logic and its come-backs

- 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

- 1 Introduction
- 2 An elementary lambda-calculus
- 3 Types and complexity soundness
- 4 Completeness
- 5 An alternative characterization of functions

Untyped $\lambda^!$ -calculus

- terms:

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

write $!^k M = \underbrace{! \dots !}_k M$

- *depth* $\delta(N, M)$ of a subterm N in M :
number of nested $!$ enclosing N .
- *depth* $\delta(M)$ of a term M :
maximal depth of all its subterms.

example: $M = !((\lambda x.x) !!y)$

so $\delta(\lambda x.x, M) = 1$, $\delta(y, M) = 3$ and $\delta(M) = 3$

Reduction

- reduction rules:

$$(\lambda x.M)N \rightarrow M[N/x] \quad (\beta\text{-redex})$$

$$(\lambda^! x.M)!N \rightarrow M[N/x] \quad (!\text{-redex})$$

- the reduction is confluent
- intuition behind the abstractions:

$\lambda x.M$ expects input from depth 0

$\lambda^! x.M$ expects input from depth 1

Well-formedness

■ *well-formed terms:*

M such that, for any sub-term which is an abstraction,

- 1 $\lambda x.N$ implies x occurs in N at most *once* at depth 0
- 2 $\lambda^!x.N$ implies x occurs in N (possibly) many times at depth 1

example:

$\lambda f.\lambda x.(f (f x))$ non w.f.

$\lambda^!f.!(\lambda x.(f (f x)))$ w.f.

Well-formedness

- *well-formed terms*:

M such that, for any sub-term which is an abstraction,

- 1 $\lambda x.N$ implies x occurs in N at most *once* at depth 0

- 2 $\lambda^!x.N$ implies x occurs in 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 $M \rightarrow M'$, then $\delta(M') \leq \delta(M)$.

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

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

Reduction strategy

Lemma

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

let $\delta = \delta(M)$:

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

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

Data

■ Booleans: $\text{TT} = \lambda x.\lambda y.x$ $\text{FF} = \lambda x.\lambda y.y$

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

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

■ Scott words:

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

Representing functions

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

$$P!_{\underline{w}} \rightarrow !^k \overline{f(w)},$$

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

Complexity bounds

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

Property

The term $P!_{\underline{w}}$ reduces by the level-by-level strategy to a 2-normal form in $\text{poly}(n)$ 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 $P!_{\underline{w}}$ reduces by the level-by-level strategy to a i -normal form in $2_{i-2}^{\text{poly}(n)}$ 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:

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

where

$\forall a. A$ polymorphic quantification

$\mu a. A$ type fixpoint

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

Type system

- typing judgement:

$$\Gamma \mid \Delta \mid \Theta \vdash M : \sigma$$

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 M is well-typed iff there is $\Gamma \mid \Delta \mid \emptyset \vdash M : \sigma$

Typing rules

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

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

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

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

+ \forall and μ rules

Properties

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

Datatypes

$$B = \forall a. a \multimap a \multimap a$$

$$N = \forall a. !(a \multimap a) \multimap !(a \multimap a)$$

$$W = \forall a. !(a \multimap a) \multimap !(a \multimap a) \multimap !(a \multimap a)$$

$$W_S = \mu b. \forall a. (b \multimap a) \multimap (b \multimap a) \multimap (a \multimap a)$$

$$TT = \lambda x. \lambda y. x, \quad FF = \lambda x. \lambda y. y$$

$$\underline{n} = \lambda^! f. \lambda x. (f (f \dots (f x) \dots))$$

$$\underline{w} = \lambda^! f_0. \lambda^! f_1. !(\lambda x. (f_{i_1} (f_{i_2} \dots (f_{i_n} x) \dots))$$

$$\widehat{i \cdot w} = \lambda f_0. \lambda f_1. \lambda x. f_i \widehat{w}_{(i=0,1)} \quad \widehat{\epsilon} = \lambda f_0. \lambda f_1. \lambda x. x$$

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

Properties of datatypes

Property

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

i.e. typing ensures that by reducing up to depth k we can extract the result

Theorem (Complexity soundness)

Let $\vdash P : !W \multimap !^{k+2} B$, where P is a program, and let $\vdash \underline{w} : W$ of length n .

Then:

$P \underline{w}$ reduces to either $!^{k+2} \mathbf{TT}$ or $!^{k+2} \mathbf{FF}$ in time $2_k^{\text{poly}(n)}$.

Expressiveness

Complexity classes:

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

$$k\text{-EXP} = \text{DTIME}(2_k^{\text{poly}(n)}) \quad , \text{ for } k \geq 0$$

$$k\text{-FEXP} = \text{FDTIME}(2_k^{\text{poly}(n)}) \quad , \text{ for } k \geq 0$$

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

Lemma (Extensional completeness)

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

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 $!W \multimap !^{k+2}B$ are exactly the predicates of k -EXP.

Which output type to characterize FPTIME functions?

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

what about functions?

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

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

\leadsto use Scott binary word, i.e. $!W \multimap !^2W_S$

Capturing functions classes

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

Property

Assume $\vdash M : !^k W_S$ for $k \geq 0$ and $M \in \mathbf{nf}_k$, then $M = !^k \hat{w}$ for some $w \in \{0, 1\}^*$.

Theorem

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

Compositionality of FPTIME

- FPTIME is closed by composition, **but...**
in $!W \multimap !^2W_S$ input/output types do not match!
- as we saw before, $!W \multimap !^iW_{i=1,2}$ does not work...
- we wish to represent a word $w' \in \{0, 1\}^*$ by a pair $\langle n, w \rangle$,
where $n \in \mathbb{N}$, $w \in \{0, 1\}^*$ 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 $\langle n, w \rangle$?
 - $N \otimes W_S$: not expressive enough (duplication of integer)
 - $!N \otimes !W_S$: iteration of the word?
- ⇒ we use $!N \otimes !^2 W_S$
- output: combined type $!^{k+1} N \otimes !^{k+2} W_S$, for $k \geq 0$
- syntactic sugar on terms:

$$M \otimes N = \lambda x. xMN \quad \lambda^!(y \otimes z).M = \lambda x. x \lambda^! y. \lambda^! z.M$$

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

Properties of the datatype

Property

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

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

Theorem (Complexity soundness)

Let $\vdash P : !N \otimes !^2W_S \multimap !^{k+1}N \otimes !^{k+2}W_S$, where P is a program, and let $\vdash !\underline{n} \otimes !^2\widehat{w} : !N \otimes !^2W_S$ s.t. $|w| \leq n$.

Then:

$\vdash P(!\underline{n} \otimes !^2\widehat{w}) : !^{k+1}N \otimes !^{k+2}W_S$ can be computed in time $2_k^{\text{poly}(n)}$.

Expressiveness and characterization

Lemma (Extensional completeness)

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

which leads to the following

Theorem

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

Key points of the completeness Lemma proof

- similar to the analogous proof of $!W \multimap !^2W_S$, **but...**
to obtain the right type in the end, we need to build a pair of integer and Scott word
 - 1 argument of the function: pair $\langle n, w \rangle$
 - 2 second component of the output: the word extracted from the tape (depends on n, w)
 - 3 first component of the output: term representing $2_k^{poly(n)}$ (depends on n)
 - 4 the TM is of k -FEXP time, so the integer suffices to extract the word on the tape

Some observations

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

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, ...)