Implicit Complexity: to infinity... and beyond!

Complexité Implicite : vers l’infini et au-delà!

Romain Péchoux,
IPT Mocqua - CNRS, Inria, Université de Lorraine - LORIA

Séminaire PPS

February 4th, 2021
Today’s talk

On my work in Inria team MOCQUA (Classical and QUAntum MOdels of Computation):

- Study of computational models
- and their properties (including complexity)

Today, we will focus on:

1. a brief overview of Implicit Computational Complexity
2. a characterization of type-2 polynomial time
Computational Complexity (CC) studies problems/functions wrt resource usage.

The Universe of mathematical functions

(Images: NASA)
Computational Complexity (CC)

Computational Complexity (CC) studies problems/functions wrt resource usage.

The Universe of mathematical functions

The Galaxy of computable functions

(Images: NASA)
Computational Complexity (CC) studies problems/functions wrt resource usage.

Assume Cobham-Edmonds thesis: \textit{tractable/feasible} = \textit{polynomial time}.

(Images: NASA)
Implicit computational complexity (ICC)

**ICC**: Subfield of CC aiming at providing characterizations of complexity classes:
- **machine-independent**
- with **no prior knowledge** on the complexity analyzed codes

If the characterization is **tractable** then ICC provides **automatic** static complexity analysis methods for **high level** PL.

**State of the art**: 
- 30 years of intensive research,
- hundreds of publications,
- some academic tools
  - (Costa, SPEED, TcT, ...).
The ICC approach

ICC criterion

Take your favourite PL $\mathcal{L}$ and your favourite complexity class $\mathcal{C}$:

$\mathcal{R} \subseteq \mathcal{L}$ is an ICC criterion if $\{[[p]] \mid p \in \mathcal{R}\} = \mathcal{C}$.

Examples of complexity class $\mathcal{C}$

- $\mathcal{P}$, $\mathcal{FP}$,
- $\mathcal{PSPACE}$, $\mathcal{FPSPACE}$,
- $\mathcal{EXP}$, $2\cdot\mathcal{EXP}$, $\ldots$, ELEMENTARY,
- $\mathcal{NP}$,
- $\mathcal{NC}^0$, $\mathcal{NC}^1$, $\ldots$, $\mathcal{NC}$
- $\mathcal{PP}$, $\mathcal{BPP}$, $\mathcal{EQP}$, $\mathcal{BQP}$, $\ldots$
A bunch of techniques (1/2)

Some ICC criteria

- **function algebra**: [Cobham65], [Bellantoni-Cook92], [Clote99] for a survey

- **linear logic** based approaches
  - light logics: LLL [Girard87], LAL [Asperti98], DLAL [Baillot-Terui04],
  - soft logics: SLL [Lafont04], STA [Gaboardi-Ronchi Della Rocca07],
  - non size-increasing [Hofmann99].

- **“potential”** based methods
  - interpretations: “quasi” [Bonfante-Marion-Moyen11], “sup” [Marion-Péchoux09], higher-order [Baillot-Dal Lago16],
  - amortized resource analysis: [Jost et al.10], [Hoffmann-Hofmann10],
  - sized-types: [Vasconcelos08], [Avanzini-Dal Lago17],
  - cost semantics: [Danner et al.15].
Some ICC criteria

- **control flow (tiering-based) techniques:**
  - safe recursion [Bellantoni-Cook92],
  - ramified recurrence [Leivant-Marion94],
  - tiering [Marion11],
  - read-only/write-only: [Jones01], [De Carvalho-Simonsen14].

- **matrix-based** type systems:
  - $\mu$-measure [Niggl-Wunderlich06],
  - mwp bounds [Kristiansen-Jones09], resource control graphs [Moyen09].

- **empirical approaches** (some of them using abstract interpretations): COSTA [Albert et al.07], SPEED [Gulwani09], TcT[Avanzini-Moser-Schaper16].
Main techniques (1/2): typing

Tractable functions

Characterized by all techniques by preventing exponentiation, i.e. by preventing the iteration of methods duplicating the size of their inputs.

- Prevent iteration with a type discipline:
  - \( !A \rightarrow \& A \) in LAL,
  - \( 1 \rightarrow 0 \) in tier-based approaches,
  - Read-Only \( \rightarrow \) Write-Only in Jones/Simonsen
  - \( \begin{pmatrix} P \\ \vdots \end{pmatrix} \) in mwp (whereas \( \begin{pmatrix} M \\ \vdots \end{pmatrix} \) is required for iterability).
Main techniques (2/2): potentials

Tractable functions

Characterized by all techniques by preventing exponentiation, i.e. by preventing the iteration of methods duplicating the size of their inputs.

> By using a potential-based constraints implying a decrease along reduction:

\[
\begin{align*}
  t_1 & \rightarrow t_2 & \rightarrow \cdots & \rightarrow t_n \\
  P & \geq [t_1] & \geq [t_2] & \geq \cdots & \geq [t_n]
\end{align*}
\]

> (polynomial) interpretations-based methods,

> amortized resource analysis,

> ert-transformers method [Kaminski et al.06],

> sized-types.
### Intensional limits

**Definition [Intensional completeness]**

A characterization is intensionally complete if any tractable algorithm computing this function is accepted.

**Theorem [Hajek79]**

Providing an intensionally-complete characterization of tractable functions is a $\Sigma_0^2$-complete problem.

However, for automation purpose, the studied characterizations are decidable (even better tractable).

**Observation**

Hence there are false negative.
Beyond ICC: extensions

Intensional improvements

- Soft Type Assignment [Gaboardi-Ronchi Della Rocca07]
- Dual Light Affine Logic [Baillot-Terui04]
- Sup-interpretations [Marion-Péchoux09]

Adaptations of existing tools

- Tiering on imperative programs [Marion11], [Marion-Leivant13]
- Tiering on OO programs [Hainry-Péchoux18]
- Interpretations of HO-TRS (STTRS) [Baillot-Dal Lago12]

Extensions to new paradigms

- Concurrent systems
  - Light logics and multi-threads [Amadio-Madet11]
  - Soft logics and processes [Martini-Dal Lago-Sangiorgi16]
- Probabilistic programs: [Avanzini-Dal Lago-Ghyselen19]
- Quantum programs [Dal Lago-Masini-Zorzi10]
- Real functions [Bournez-Gomaa-Hainry11]
- Coinductive data [Gaboardi-Péchoux15]
Summary on ICC

Strong links with other research domains:

- Termination techniques (often coming from and/or combined with)
- Computability theory (Primrec, undecidable classes, . . .)
- Finite model theory (common goals)
- Static analysis (type systems, abstract interpretations, empirical approaches)

A survey on ICC in my HDR, available at https://members.loria.fr/RPechoux/
ICC: the case of “infinite” data (1/2)

Several research directions:

▶ streams, infinite structures,
▶ coinductive data types,
▶ higher order functions,
▶ real numbers and their representations,
▶ higher order complexity.
ICC: the case of “infinite” data (2/2)

Some corresponding results:

- Extension of interpretations to stream data [Gaboardi-Péchoux09]
- Extension of LALC to coductive data [Gaboardi-Péchoux15]
- Characterization of P/poly in the infinitary lambda calculus [Mazza14]
- The parsimonious lambda-calculus [Mazza-Terui15]:
  - Characterization of P/poly [Mazza-Terui15]
  - Characterization of P(\mathbb{R}) [Hainry-Mazza-Péchoux20]
- Characterization of BFF [Féree-Hainry-Hoyrup-Péchoux15]
  - first PL based characterization of type-2 polytime
  - but not tractable (type-2 polynomial inequalities).
A reminder on type-2 polynomial time

BFF was introduced by Melhorn in 1976.

**Theorem [Cook and Urquhart [1989]]**

\[
\text{BFF} = \lambda (F \cup \{R\})_2
\]

\(R\) is a type-2 bounded iterator:

\[
R(\epsilon, a) = a \\
R(ix, a) = \min(\phi(ix, R(x, a)), \psi(ix))
\]

**Theorem [Cook and Kapron [1990]]**

The set of functionals computable by an OTM in time \(P(|\phi|, |a|)\) is exactly BFF.

2nd order polynomials and size function are defined by:

- \(P(X_1, X_0) \ ::= c \in \mathbb{N} \mid X_0 \mid X_1(P) \mid P + P \mid P \times P\)
- \(|\phi|(n) = \max_{|x| \leq n} |\phi(x)|\)
How to get rid of type-2 polynomials?

Definition [Oracle Polynomial Time (OPT) [Cook92]]

Let $n^{\phi, a}$ be the biggest size of $a$ and of an oracle’s answer in the run of $M(\phi, a)$. An OTM is in OPT if its runtime is bounded by $P(n^{\phi, a})$, for some type-1 polynomial $P$.

$\text{BFF} \subset \text{OPT}$ as it contains exponential functions.

Theorem [Kapron and Steinberg [2018]]

$\text{BFF} = \lambda(\text{OPT} \cap \text{FLR})_2 = \lambda(\text{OPT} \cap \text{FLAR})_2$

- FLR = Finite Length Revision
- FLAR = Finite LookAhead Revision
Finite Length Revision

**Definition [Finite Length Revision - Kawamura and Steinberg [2017]]**

An OTM is in FLR, if, for any input, the number of times the oracle answer is bigger than all of the previous oracle answers is bounded by a constant.

**Example**

```plaintext
while (x > 0) {
  y = φ(x);
  x = x - 1;
}
```

not (FLR) if \( φ \downarrow \)

**Example**

```plaintext
while (x < n \&\& y < 8) {
  y = φ(x);
  x = x + 1;
}
```

(FLR) with constant 8
Finite LookAhead Revision

Definition [Finite LookAhead Revision - Kapron and Steinberg [2018]]

An OTM is in FLAR, if, for any input, the number of times a query is posed whose size exceeds the size of all previous queries is bounded by a constant.

Example

```plaintext
while (x > 0) {
    y = φ(x);
    x = x - 1;
}
(FLAR) with constant 0
```

Example

```plaintext
while (x < n && y < 8) {
    y = φ(x);
    x = x + 1;
}
not (FLAR) for φ = λz.4
```
A tier-based characterization of BFF

Imperative PL with oracles and tier-based type system

(Expressions) \( e ::= x | \text{true} | \text{false} | \text{op}(\bar{e}) | \phi(e \upharpoonright e) \)

(Statements) \( st ::= [x:=e]; | st \; st \; | \text{while}(e)\{st\} | \text{if}(e)\{st\} \text{else}\{st\} \)

Let \([ST]\) be the set of functions computed by programs in ST (Safe and Terminating).

Theorem [Hainry-Kapron-Marion-Péchoux20]

\[ \text{BFF} = \lambda([ST])_2 \]

Drawbacks

- lambda closure (for completeness)
- termination assumption (for soundness)
A new tier-based characterization of BFF

New tier-based PL

e\text{\textsuperscript{i}}: e is a type-i object.

(Expressions) \quad e ::= x^0 \mid \text{op}(\overline{e}) \mid x^1(e \mapsto e)

(Statements) \quad st ::= [x^0 := e]; \mid st \mid if(e)\{st\}\{st\} \mid while(e)\{st\}

(Procedures) \quad P ::= P(x^1, x^0)\{st \text{ return } x^0\}

(Terms) \quad t ::= x^i \mid \lambda x^i.t \mid t_1 @ t_2 \mid \text{call } P(\{x^0 \rightarrow t^0\}, t^0)

(Programs) \quad prog ::= t^0 \mid \text{declare } P \text{ in } prog

- SAFE is the set of typable programs.
- SAFE_0 is the set of typable programs without lambda-abstraction.
- SN is the set of strongly normalizing programs.
Example

\[
\text{prog}(X, x) = \text{declare } KS(Y, Z, v) \{
    w := Y(\varepsilon \uparrow \varepsilon) ;
    z := \varepsilon ;
    \text{while } (v \neq \varepsilon) \{
        v := \text{pred}(v) ;
        z := Z(z \uparrow w)
    \}
    \text{return } z
\}
\text{in call } KS(\{y \rightarrow X \odot y\}, \{y \rightarrow X \odot (X \odot y)\}, x)
\]

\begin{itemize}
    \item \([\text{prog}] \in (W \rightarrow W) \rightarrow W \rightarrow W\)
    \item \([\text{prog}] (f^{W \rightarrow W}, w^W) = F_{|w|}(f) \text{ with } \begin{cases} F_0(f) = \varepsilon \\ F_{n+1}(f) = (f \circ f)(F_n(f) \leq |f(1)|) \end{cases} \)
    \item \text{prog} \in \text{SAFE}_0 \cap \text{SN}
\end{itemize}
A complete type-1 polynomial time characterization of BFF

Soundness and completeness without lambda closure:

Theorem [New! [Hainry, Kapron, Marion, Péchoux]]

\[ \text{BFF} = \llbracket \text{SAFE}_0 \cap \text{SN} \rrbracket = \llbracket \text{SAFE} \cap \text{SN} \rrbracket \]

Preserved for a Ptime instance of Size Change Termination [Lee-BenAmram-Jones01]:

Theorem [New! [Hainry, Kapron, Marion, Péchoux21]]

\[ \text{BFF} = \llbracket \text{SAFE}_0 \cap \text{SCP}_S \rrbracket = \llbracket \text{SAFE} \cap \text{SCP}_S \rrbracket \]

\( \text{SCP}_S \) can be decided in time quadratic in \(|\text{prog}|\) (using [BenAmram-Lee07]).

Theorem [New! [Hainry, Kapron, Marion, Péchoux21]]

- \( \text{prog} \in \text{SAFE} \cap \text{SCP}_S \) is Ptime-complete (using [Mairson04])
- \( \text{prog} \in \text{SAFE}_0 \cap \text{SCP}_S \) is in time cubic in \(|\text{prog}|\) (using [Hainry et al.20])
Conclusion

There are plenty of ICC-related open issues:

▶ Continue to extend techniques using
   ▶ Simulation techniques/Improvement theory [Sands95],
   ▶ Non idempotent intersection types [Kesner-Vial20],
   ▶ Shape-analysis [Sagiv et al.02]

▶ Continue the development of practical tools
   ▶ ComplexityParser [Hainry-Jeandel-Péchoux-Zeyen20]

▶ Develop extensions to Quantum programs
   ▶ What kind of programs? [Péchoux-Perdrix-Rennela-Zamdzhiev20]
   ▶ Existing ICC Probabilistic approaches
     [Avanzini-Dal Lago-Ghyselen19], [Avanzini-Moser-Schaper20]
   ▶ Existing ICC Quantum approaches [Dal Lago-Masini-Zorzi10], [Yamakami18]
   ▶ What is the role of entanglement?
Thank you for your attention!
Type system: simply-typed part

\[ \Gamma, \Omega, \Delta \vdash \overline{x} : \overline{W} \rightarrow \overline{W} \quad \Gamma, \Omega, \Delta \vdash \overline{x}, \overline{y}, x : \overline{W} \]  

(PR)

\[ \Gamma, \Omega, \Delta \vdash P(\overline{x}, \overline{x}) \{ \text{st return } \overline{x} \} : (\overline{W} \rightarrow \overline{W}) \rightarrow \overline{W} \rightarrow \overline{W} \]  

(CC)

\[ \Gamma, \Omega, \Delta \vdash \text{call } P(\overline{c}, \overline{t}) : \overline{W} \]  

\[ \Gamma, x : \overline{T} \]  

(VAR)

\[ \Gamma, \Omega, \Delta \vdash t_1 : T \rightarrow T' \quad \Gamma, \Omega, \Delta \vdash t_2 : T \]  

(APP)

\[ \Gamma, \Omega, \Delta \vdash t_1 \@ t_2 : T' \]  

\[ \Gamma, \Omega, \Delta \vdash \lambda x. t : T \rightarrow T' \]  

(ABS)

\[ \Gamma, \Omega, \Delta \vdash \text{prog} : T \]  

\[ \Gamma, \Delta \vdash \text{body}(p) : (k, \text{k}_{\text{in}}, \text{k}_{\text{out}}) \quad \Omega(p) = \langle \Gamma, (k, \text{k}_{\text{in}}, \text{k}_{\text{out}}) \rangle \]  

(DEC)

\[ \Gamma, \Omega, \Delta \vdash \text{declare } p \text{ in prog} : T \]
Type system: tier-based part $1/2$

\[
\begin{align*}
\Gamma(x) &= k \\
\Gamma, \Delta \vdash x : (k, k_{in}, k_{out}) & \quad \text{(VAR)} \\
k_1 \to \cdots \to k_{|e|} \to k \in \Delta(\text{op})(k_{in}) & \quad \forall i \leq |e|, \quad \Gamma, \Delta \vdash e_i : (k_i, k_{in}, k_{out}) \\
\Gamma, \Delta \vdash \text{op}(e) : (k, k_{in}, k_{out}) & \quad \text{(OP)} \\
\Gamma, \Delta \vdash e_1 : (k, k_{in}, k_{out}) & \quad \Gamma, \Delta \vdash e_2 : (k_{out}, k_{in}, k_{out}) & \quad k \prec k_{in} \land k \preceq k_{out} \\
\Gamma, \Delta \vdash x(e_1 \upharpoonright e_2) : (k, k_{in}, k_{out}) & \quad \text{(OR)} \\
\Gamma, \Delta \vdash \text{st} : (k, k_{in}, k_{out}) & \quad \text{(SUB)} \\
\Gamma, \Delta \vdash \text{st} : (k+1, k_{in}, k_{out}) & \\
\Gamma, \Delta \vdash \text{st}_1 : (k, k_{in}, k_{out}) & \quad \Gamma, \Delta \vdash \text{st}_2 : (k, k_{in}, k_{out}) & \quad \text{(SEQ)} \\
\Gamma, \Delta \vdash \text{st}_1 ; \text{st}_2 : (k, k_{in}, k_{out})
\end{align*}
\]
Type system: tier-based part 2/2

\[
\Gamma, \Delta \vdash x : (k_1, k_{in}, k_{out}) \quad \Gamma, \Delta \vdash e : (k_2, k_{in}, k_{out}) \quad k_1 \preceq k_2 \quad \text{(ASG)}
\]

\[
\Gamma, \Delta \vdash x := e : (k_1, k_{in}, k_{out})
\]

\[
\Gamma, \Delta \vdash e : (k, k_{in}, k_{out}) \quad \forall w \in \{true, false\}, \quad \Gamma, \Delta \vdash st_w : (k, k_{in}, k_{out}) \quad \text{(CND)}
\]

\[
\Gamma, \Delta \vdash if(e)\{st_{true}\} \text{ else } \{st_{false}\} : (k, k_{in}, k_{out})
\]

\[
\Gamma, \Delta \vdash e : (k, k_{in}, k) \quad \Gamma, \Delta \vdash st : (k, k, k) \quad 1 \preceq k \quad \text{(WINIT)}
\]

\[
\Gamma, \Delta \vdash while(e)\{st\} : (k, k_{in}, 0)
\]

\[
\Gamma, \Delta \vdash e : (k, k_{in}, k_{out}) \quad \Gamma, \Delta \vdash st : (k, k_{out}) \quad 1 \preceq k \preceq k_{out} \quad \text{(WH)}
\]