Tiered complexity at higher order

Emmanuel Hainry Bruce Kapron* Jean-Yves Marion
Romain Péchoux

LORIA, Université de Lorraine and University of Victoria*

湘南国際村センター 2019

Introduction

Study of polynomial time complexity:

- **Type-1** $(\mathbb{N} \to \mathbb{N})$:
  - Several tools for program analysis:
    - type systems (light logics),
    - interpretations (abstract, polynomial, ...),
    - ...
- **Type-2** $((\mathbb{N} \to \mathbb{N}) \to \mathbb{N})$ and above:
  - No tools.
  - Programming languages with restrictions:
    - BTLP, ITLP (Irwin-Kapron-Royer [2001])

Goal: a static analysis tool for certifying **Type-2** polynomial time complexity

Introduction to type-2 complexity

Type-2 polynomial time $\mathtt{FP}_2$ has been defined by Mehlhorn [1976].

Theorem [Cook and Urquhart [1993]]

$$\mathtt{FP}_2 = \lambda(\mathtt{FP}_1 \cup \{\mathcal{R}\})_2$$

- $\mathtt{FP}_1$ is the class of type-1 polynomial time functions,
- $\mathcal{R} : \Sigma^* \times \Sigma^* \times (\Sigma^* \to \Sigma^*) \times (\Sigma^* \to \Sigma^*) \to \Sigma^*$ is defined by:

$$\begin{aligned} \mathcal{R}(\epsilon, a, \phi, \psi) &= a \\ \mathcal{R}(ix, a, \phi, \psi) &= \min(\phi(ix, \mathcal{R}(x, a, \phi, \psi)), \psi(ix)), \end{aligned}$$

- min returns the operand of minimal size.

Basic Feasible Functionals

Theorem [OTM based characterization by Cook-Kapron[1990]]

The set of type-2 functionals computable by an Oracle Turing Machine (OTM) M in time $P(|\phi|, |\mathbf{a}|)$ is exactly $\mathtt{FP}_2$.

- OTM are Turing Machines with an oracle ϕ,
- P is a type-2 polynomial defined by:

$$P(X_1, X_0) ::= c \in \mathbb{N} \mid X_0 \mid X_1(P) \mid P + P \mid P \times P,$$

- $|\phi|(n) = \max_{|x| \leq n}(|\phi(x)|)$.

The class $\mathtt{FP}_2$ is called BFF for Basic Feasible Functionals.

How to get rid of type-2 polynomials?

One option: Oracle Polynomial Time (OPT) by Cook[1992]:

Definition

$m_{\phi,\mathbf{a}}^{M}$ is the maximum of the size of the input **a** and of the biggest oracle's answer in the run of $M(\phi, \mathbf{a})$.

Definition

An OTM is in OPT if it runs in time bounded by $P(m_{\phi,\mathbf{a}}^{M})$ on any input, for some type-1 polynomial P.

However $BFF \subsetneq OPT$ as it contains exponential functions.

How to recover $\mathtt{FP}_2$: finite length revision

Definition [Finite Length Revision]

An OTM has Finite Length Revision (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

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

not (FLR) if $\phi \searrow$

Example

```
x = 0;
while (x<n && y<8){
        y = φ(x);
        x = x+1;
}
```

(FLR) with constant 8

How to recover $\mathtt{FP}_2$: finite lookahead revision

Definition [Finite LookAhead Revision]

An OTM has Finite LookAhead Revision (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

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

(FLAR) with constant 0

Example

```
x = 0;
while (x<n && y<8){
      y = ϕ(x);
      x = x+1;
}
```

not (FLAR) for $\phi = \lambda n.4$

How to recover $\mathtt{FP}_2$?

Definition

- $SPT = OPT \cap FLR$
- $MPT = OPT \cap FLAR$

Both $SPT \subsetneq \mathtt{FP}_2$ and $MPT \subsetneq \mathtt{FP}_2$.

Theorem [Kapron and Steinberg[2018]]

$$\mathtt{FP}_2 = \lambda(SPT)_2 = \lambda(MPT)_2$$

Motivations

- Find a criterion for complexity certificates.
- Provide a characterization of $\mathtt{FP}_2$ on imperative languages.
- Develop a static analysis technique with polynomial bounds:
  - of type-1 (Hilbert's 10th pb, Tarski's Quantifier Elimination)
  - implicit (not explicitly provided)

Objective: **Adapt Implicit Computational Complexity** techniques to an imperative setting with oracles.

Tool: Safe recursion and **Tiering**

Safe recursion

Theorem [Bellantoni-Cook[1992]]

The class of functions:

- constants, projections, successor, predecessor, conditional,
- defined by safe composition:

$$f(\overline{x}^{\mathbf{1}};\overline{a}^{\mathbf{0}}) = s(r(\overline{x}^{\mathbf{1}};);t(\overline{x}^{\mathbf{1}};\overline{a})^{\mathbf{0}})$$

- and defined by safe recursion:

$$\begin{aligned} f(\epsilon,\overline{y}^{\mathbf{1}};\overline{a}^{\mathbf{0}}) &= g(\overline{y}^{\mathbf{1}};\overline{a}^{\mathbf{0}}) \\ f(i(x)^{\mathbf{1}},\overline{y}^{\mathbf{1}};\overline{a}) &= h_i(x^{\mathbf{1}},\overline{y}^{\mathbf{1}};f(x^{\mathbf{1}},\overline{y}^{\mathbf{1}};\overline{a})^{\mathbf{0}}) \qquad i \in \{0,1\}, \end{aligned}$$

provided s, r, t, g, h_i are already defined in the class,

is exactly FP_1.

Tiering

Imperative language over binary words Σ^*

$$E ::= \mathtt{x} \mid \mathtt{true} \mid \mathtt{false} \mid op(E, \dots, E)$$
$$I ::= [\mathrm{x}{:=}E]; \mid I\ I \mid \mathtt{while}(E)\{I\} \mid \mathtt{if}(E)\{I\}\mathtt{else}\{I\}$$

Tier $\tau \in \{\mathbf{0}, \mathbf{1}\}$ with $\mathbf{0} < \mathbf{1}$.

Intuition:

- $\mathbf{0}$: data may grow and cannot control the program flow.
- $\mathbf{1}$: data cannot grow and may control the program flow.

Typing rules

$$\frac{\Gamma(\mathtt{x}) = \tau}{\Gamma \vdash \mathtt{x} : \tau} \qquad \frac{\Gamma \vdash e : \tau}{\Gamma \vdash op(e) : \tau}\ (\text{Des}) \qquad \frac{\Gamma \vdash e : \tau}{\Gamma \vdash op(e) : \mathbf{0}}\ (\text{Cons})$$

$$\frac{}{\Gamma \vdash c : \tau}\ \text{Cst} \qquad \frac{\Gamma \vdash I : \tau \quad \tau \leq \tau'}{\Gamma \vdash I : \tau'}\ (\text{Sub})$$

$$\frac{\Gamma \vdash I_1 : \tau \quad \Gamma \vdash I_2 : \tau}{\Gamma \vdash I_1\ I_2 : \tau}\ (\text{Seq}) \qquad \frac{\Gamma \vdash e : \tau \quad \Gamma \vdash I_i : \tau}{\Gamma \vdash \mathtt{if}(E)\{I_1\}\mathtt{else}\{I_2\} : \tau}\ (\text{If})$$

$$\frac{\Gamma \vdash \mathtt{x} : \tau \quad \Gamma \vdash E : \tau' \quad \tau \leq \tau'}{\Gamma \vdash \mathtt{x} := E : \tau}\ (\text{A}) \qquad \frac{\Gamma \vdash E : \mathbf{1} \qquad \Gamma \vdash I : \tau}{\Gamma \vdash \mathtt{while}(E)\{I\} : \mathbf{1}}\ (\text{Wh})$$

Safe operators

Extension to polynomial time computable operators:

$$op :: \tau_1 \times \ldots \times \tau_n \to \tau$$

- Neutral operators computing a predicate :

$$\tau \leq \min_{i \in [1,n]} \tau_i$$

- Positive operators satisfying:

$$\forall \overline{w},\ |[\![op]\!](w_1, \ldots, w_n)| \leq \max_{i \in [1,n]} |w_i| + c,\ \text{for } c \geq 0$$

$$\tau = \mathbf{0}$$

Example: addition

Example ($add :: int \times int \rightarrow int$)

```
add(x,y){
  while (x>0){
    x = x-1;
    y = y+1;
  }
  return y;
}
```

- `y` is necessarily of tier $\mathbf{0}$.
- `x` is necessarily of tier $\mathbf{1}$.
- consequently, `add` $:: \mathbf{1} \times \mathbf{0} \rightarrow \mathbf{0}$.

Example: multiplication

Example ($mult :: int \times int \rightarrow int$)

```
mult(x,y){
  int z = 0;
  while (x>0){
      x = x-1;
      z = add(y,z);    //add:1 × 0 → 0
  }
  return z;
}
```

- the output of add is $\mathbf{0}$. Consequently, z is of tier $\mathbf{0}$.
- both x and y are of tier $\mathbf{1}$.
- consequently, `mult` $:: \mathbf{1} \times \mathbf{1} \rightarrow \mathbf{0}$.

Counter-example: exponential

Example ($exp :: int \to int$)

```
exp(x){
  int y=1;
  while (x>0){
     x = x-1;
     z = y;
     y^0 = add(y^1,z);      //add:1 × 0 → 0
  }
  return y;
}
```

- The tier of y cannot be defined!

Results

Theorem [Marion [2011]]

The set of functions computable by a typable and terminating program with safe operators is exactly $\mathtt{FP}_1$.

- Soundness:
  - No flow from **0** to **1** (guards of tier **1**)
  - At most n^k configurations under termination assumption
- Completeness:
  - Simulation of a polynomial time TM

Theorem [Hainry, Marion and Péchoux [2013]]

Type inference can be done in polynomial time.

- Reduction to 2-SAT

Imperative language with oracles

Design a type system ensuring that programs are in $MPT = OPT \cap FLAR$.

$$E ::= \mathtt{x} \mid \mathtt{true} \mid \mathtt{false} \mid op(E, \ldots, E) \mid \boldsymbol{\phi(\mathbf{E} \restriction \mathbf{E})}$$
$$I ::= [\mathtt{x}{:=}E]; \mid I\ I \mid \mathtt{while}(E)\{I\} \mid \mathtt{if}(E)\{I\}\mathtt{else}\{I\}$$

In $\phi(w \restriction v)$:

- w is the oracle input
- v is the oracle input bound
- $w \restriction v = w_1 \ldots w_{|v|}$, if $|v| \geq k$

Towards a type system for MPT

Observations:

1. The number of lookahead revisions can be controlled by tiers.
2. A restriction on the oracle input bound is needed.
3. Operators are in need of a more flexible treatment.

Solutions:

1. Use more than two tiers: $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \ldots, \mathbf{k}, \ldots\}$.
2. Keep track of the tier of the outermost while $\mathbf{k}_{out}$.
3. Keep track of the tier of the innermost while $\mathbf{k}_{in}$.

Judgments: $\Gamma, \Delta \vdash I : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})$

Type system (easy)

$$\frac{\Gamma(\mathrm{x}) = \mathbf{k}}{\Gamma, \Delta \vdash \mathrm{x} : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})} \qquad \frac{\forall i \in \{1,2\},\ \vdash I_i : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})}{\vdash I_1\ \ I_2\ \ : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})}\ \text{(SEQ)}$$

$$\frac{}{\vdash\ ;\ \ : (\mathbf{0}, \mathbf{k}_{in}, \mathbf{k}_{out})}\ \text{(SK)} \qquad \frac{\vdash I\ \ : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})}{\vdash I\ \ : (\mathbf{k}{+}\mathbf{1}, \mathbf{k}_{in}, \mathbf{k}_{out})}\ \text{(SUB)}$$

$$\frac{\vdash E : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out}) \qquad \forall i \in \{1,2\},\ \vdash I_i : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})}{\vdash \mathtt{if}(E)\{I_1\}\ \mathtt{else}\ \{I_2\}\ \ : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})}\ \text{(IF)}$$

$$\frac{\vdash \mathrm{x} : (\mathbf{k}_1, \mathbf{k}_{in}, \mathbf{k}_{out}) \qquad \vdash E : (\mathbf{k}_2, \mathbf{k}_{in}, \mathbf{k}_{out}) \qquad \mathbf{k}_1 \preceq \mathbf{k}_2}{\vdash \mathrm{x} := E\ \ : (\mathbf{k}_1, \mathbf{k}_{in}, \mathbf{k}_{out})}\ \text{(ASG)}$$

Type system (hard)

$$\frac{\mathbf{k}_1 \to \cdots \to \mathbf{k}_n \to \mathbf{k} \in \Delta(op)(\mathbf{k}_{in}) \qquad \forall i,\ \vdash E_i : (\mathbf{k}_i, \mathbf{k}_{in}, \mathbf{k}_{out})}{\Gamma, \Delta \vdash op(E_1, \ldots, E_n) : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})} \text{ (OP)}$$

with $\mathbf{k}_1 \to \cdots \to \mathbf{k}_n \to \mathbf{k} \in \Delta(op)(\mathbf{k}_{in})$ if:

- $\mathbf{k} \leq \min_{i \in [1,n]} \mathbf{k}_i$ and $\max_{i \in [1,n]} \mathbf{k}_i \leq \mathbf{k}_{in}$
- $\mathbf{k} < \mathbf{k}_{in}$ for positive operators.

$$\frac{\vdash E : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out}) \quad \vdash E' : (\mathbf{k}_{out}, \mathbf{k}_{in}, \mathbf{k}_{out}) \quad \mathbf{k} < \mathbf{k}_{in} \quad \mathbf{k} \leq \mathbf{k}_{out}}{\vdash \phi(E \upharpoonright E') : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})} \text{ (OR)}$$

$$\frac{\vdash E : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out}) \qquad \vdash I : (\mathbf{k}, \mathbf{k}, \mathbf{k}_{out}) \qquad \mathbf{1} \preceq \mathbf{k} \preceq \mathbf{k}_{out}}{\vdash \mathtt{while}(E)\{I\}\ : (\mathbf{k}, \mathbf{k}_{in}, \mathbf{k}_{out})} \text{ (W)}$$

Example

Example

The program computes the decision problem $\exists n \leq x,\ \phi(n) = 0$.

$$
\begin{array}{l}
\mathtt{y} = \mathtt{x}\ ; \\
\mathtt{z} = \textit{false}\ ; \\
\mathtt{while}(\mathtt{x}^{\mathbf{1}} >= 0)\{ \\
\quad \mathtt{if}(\phi(\mathtt{y}^{\mathbf{0}} \restriction \mathtt{x}^{\mathbf{1}}) == 0)\{ \\
\quad\quad \mathtt{z}^{\mathbf{0}} = \textit{true}\ ; \\
\quad \}\ \mathtt{else}\ \{;\} \\
\quad \mathtt{x}^{\mathbf{1}} = \mathtt{x}^{\mathbf{1}} - 1; \\
\} \\
\ \mathtt{return}\ \ \mathtt{z};
\end{array}
$$

The program is in MPT.

The program is typable and the inner command has tier $(\mathbf{1}, \mathbf{1}, \mathbf{1})$.

A more complex example

Example

$\sum_{i=0}^{\max_{x=0}^{n} \phi(x)} \phi(i)$ can be computed by:

$$
\begin{array}{l}
\mathtt{x} := n\ ;\\
\mathtt{y}^{2} := \mathtt{x}^{3}\ ;\\
\mathtt{z}^{2} := 0\ ;\\
\mathtt{while}(\mathtt{x}^{3} >= 0)\{\\
\quad \mathtt{z}^{2} := \max(\phi(\mathtt{y}^{2} \upharpoonright \mathtt{x}^{3})^{2}, \mathtt{z}^{2})\ ;\\
\quad \mathtt{x}^{3} := \mathtt{x} - 1^{3}\ ;\\
\};\\
\mathtt{v}^{1} := \mathtt{z}^{2}\ ;\\
\mathtt{u}^{0} := 0\ ;\\
\mathtt{while}(\mathtt{z}^{2} >= 0)\{\\
\quad \mathtt{w}^{1} := \phi(\mathtt{v}^{1} \upharpoonright \mathtt{z}^{2})^{1}\ ;\\
\quad \mathtt{while}(\mathtt{w}^{1} >= 0)\{\\
\quad\quad \mathtt{u}^{0} := \mathtt{u} + 1^{0}\ ;\\
\quad\quad \mathtt{w}^{1} := \mathtt{w} - 1^{1}\ ;\\
\quad \}\ ;\\
\quad \mathtt{z}^{2} := \mathtt{z}^{2} - 1\ ;\\
\}\\
\mathtt{return}\ \ \mathtt{u}\ ;
\end{array}
$$

This program can be typed by $(\mathbf{3}, \mathbf{0}, \mathbf{0})$.

False negative

Example

The program computes the decision problem $\exists n \leq x,\ \phi(n) = 0$.

$$
\begin{array}{l}
\mathtt{x} := \epsilon\ ; \\
\mathtt{z} := 0\ ; \\
\mathtt{while}(\mathtt{y} >= \mathtt{x})^{\mathbf{k}}\{ \\
\quad \mathtt{if}(\phi(\mathtt{y} \upharpoonright \mathtt{x}) == 0)\{\mathtt{z} := 1\}\ \mathtt{else}\ \{;\} \\
\quad \mathtt{x} := \mathtt{x} + 1\ ;\ \ : (\mathbf{k}, \mathbf{k}, \mathbf{k}') \\
\} \\
\ \mathtt{return}\ \ \mathtt{z}\ ;
\end{array}
$$

$\mathtt{x}$ and $\mathtt{y}$ have tier at least $\mathbf{k}$ in the guard.

$\mathtt{x}$ is of tier strictly less than the inner tier $\mathbf{k}$ as $+1$ is positive.

But it is not in *FLAR*.

Results

Let ST be the class of typable and terminating programs.

Theorem [Soundness]

$ST \subseteq \lambda(MPT)_2$.

Theorem [Completeness]

$ST_1 = \mathtt{FP}_1$
$\lambda(ST)_2 = \mathtt{FP}_2$.

By simulating a variant of $\mathcal{R}$.

Conclusion

Conclusion

We have presented:

- a completeness result at type-1,
- a completeness result at type-2 for a natural extension,
- a decidable type inference (in polynomial time).

Drawbacks and Open questions

- Termination is assumed.
- Completeness is obtained under lambda-closure.