Higher-order Interpretations for Higher-order Complexity

Emmanuel Hainry & Romain Péchoux

DICE-FOPARA, 22-23 april 2017

CNRS, INRIA, Université de Lorraine & LORIA, Nancy, France

Introduction

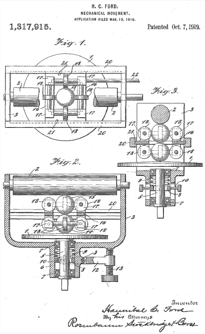
First-order computability and complexity

- First-order **computability** is well understood:
  - Agreed upon definitions
  - Hierarchies layering the Turing degree of problems
  - Church-Turing's thesis

- First-order **computational Complexity** is well understood:
  - Agreed upon definitions
  - Classes and hierarchies layering the difficulty of problems
  - Various characterizations:
    - machine based characterizations
    - machine independent characterizations
      $\rightarrow$ Implicit Computational Complexity

Higher-order computability and complexity

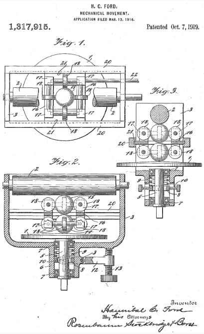
- **Higher-order computability** is (well) understood:
  - Order 2 = computations over reals.
  - **No** Church-Turing's thesis!
    - General Purpose Analog Computer by Shannon,
    - Blum-Shub-Smale model,
    - Computable Analysis (CA) by Weihrauch,
    - Oracle TM,
    - ...



Higher-order computability and complexity

- **Higher-order computability** is (well) understood:
    - Order 2 = computations over reals.
    - **No** Church-Turing's thesis!
        - General Purpose Analog Computer by Shannon,
        - Blum-Shub-Smale model,
        - Computable Analysis (CA) by Weihrauch,
        - Oracle TM,
        - ...
- **Higher-order complexity** is **not** well understood.
    - Polytime complexity on OTM = Basic Feasible Functions (BFF) by Constable, Melhorn
    - Polytime complexity in CA = $P(\mathbb{R})$ by Ko
    - No homogeneous theory for higher-order:

$$P(\mathbb{R}) \neq BFF$$



Objectives of this talk:

- **not developping** a new complexity theory for higher-order,
- **adapting first-order tools** for program complexity analysis,
- **validating** the theory by capturing existing higher-order complexity classes

Objectives of this talk:

- **not developping** a new complexity theory for higher-order,
- **adapting first-order tools** for program complexity analysis,
- **validating** the theory by capturing existing higher-order complexity classes

Framework:

- tool = (polynomial) **interpretations**
- target = $\mathbf{BFF}_i$, the Basic Feasible Functionals at any order.

First-order interpretations

First-order interpretations of TRS

- Defined in the 70s for showing TRS **termination**:
  - $\forall b$ of arity n, $(\!|b|\!) : \mathbb{N}^n \to^{\uparrow} \mathbb{N}$
  - $\forall l \to r \in R,\ (\!|l|\!) > (\!|r|\!)$
- Quasi-interpretation (QI) for **complexity** analysis:
  - $\forall b$ of arity n, $(\!|b|\!) : \mathbb{N}^n \to^{\nearrow} \mathbb{N}$
  - $\forall l \to r \in R,\ (\!|l|\!) \geq (\!|r|\!)$

Theorem (Bonfante et al. 2011)

Let QI^{poly}_{add} be the set of functions computed by TRS admitting an additive and polynomial QI.

- $QI^{poly}_{add} \cap RPO \equiv FSPACE$
- $QI^{poly}_{add} \cap RPO^{prod} \equiv FPTIME$

First-order interpretations of TRS

Example

$$
\begin{aligned}
double(\epsilon) &\rightarrow \epsilon \\
double(s(x)) &\rightarrow s(s(double(x)))
\end{aligned}
$$

First-order interpretations of TRS

Example

$$double(\epsilon) \to \epsilon$$
$$double(s(x)) \to s(s(double(x)))$$

$$(\!|\epsilon|\!) = 0,\ (\!|s|\!)(X) = X + 1,\ (\!|double|\!)(X) = 3X + 1$$

First-order interpretations of TRS

Example

$$\mathit{double}(\epsilon) \to \epsilon$$
$$\mathit{double}(s(x)) \to s(s(\mathit{double}(x)))$$

$$(\!|\epsilon|\!) = 0,\ (\!|s|\!)(X) = X + 1,\ (\!|\mathit{double}|\!)(X) = 3X + 1$$

$$(\!|\mathit{double}\ 0|\!) = 1 > 0 = (\!|\epsilon|\!)$$
$$(\!|\mathit{double}\ s(x)|\!) = 3X + 4 > 3X + 3 = (\!|s(s(\mathit{double}(x)))|\!)$$

First-order interpretations of TRS

Example

$$double(\epsilon) \to \epsilon$$
$$double(s(x)) \to s(s(double(x))$$

$$(\!|\epsilon|\!) = 0,\ (\!|s|\!)(X) = X + 1,\ (\!|double|\!)(X) = 3X + 1$$

$$(\!|double\ 0|\!) = 1 > 0 = (\!|\epsilon|\!)$$
$$(\!|double\ s(x)|\!) = 3X + 4 > 3X + 3 = (\!|s(s(double(x))|\!)$$

Termination by $RPO^{prod} \Rightarrow [\![double]\!] : x \mapsto 2x \in FPTIME$

Higher-order interpretations of TRS: State of the art

- **Termination:**
    - Van De Pol (1993) adapted interpretations for showing termination of **higher-order TRS.**

Higher-order interpretations of TRS: State of the art

- **Termination:**
  - Van De Pol (1993) adapted interpretations for showing termination of **higher-order TRS.**
- **Complexity:**
  - Férée et al. (2010) adapted interpretations to **first-order stream programs** for characterizing BFF (BFF_2) and $P(\mathbb{R})$.

Higher-order interpretations of TRS: State of the art

- **Termination:**
  - Van De Pol (1993) adapted interpretations for showing termination of **higher-order TRS.**

- **Complexity:**
  - Férée et al. (2010) adapted interpretations to **first-order stream programs** for characterizing BFF (BFF_2) and $P(\mathbb{R})$.

  - Baillot & Dal Lago (2016) adapted interpretations to **higher-order Simply Typed TRS** for characterizing FPtime.
    $\rightarrow$ a first step towards a better expressivity

Higher-order language

Higher Order Programming Language

Definition (Functional Language)

$$
\begin{array}{lcl}
M & := & x \\
 & | & c \\
 & | & op \\
 & | & M_1\ M_2 \\
 & | & \lambda x.M \\
 & | & \text{case } M \text{ of } c_1 \rightarrow M_1 | c_2 \rightarrow M_2 | ... | c_n \rightarrow M_n \\
 & | & \text{letRec } f = M
\end{array}
$$

+ Inductive Typing

Example

Example

$$\text{letRec map} = \lambda\text{g}.\lambda\text{x.case x of } \mathbf{c} \text{ y z} \rightarrow \mathbf{c} \text{ (g y) (map g z)}$$
$$\mid \mathbf{nil} \rightarrow \mathbf{nil}$$

$$\text{List(A)} ::= \mathbf{nil} \mid \mathbf{c} \text{ A List(A)}$$

map: $(A \rightarrow B) \rightarrow List(A) \rightarrow List(B)$

Semantics

Four kinds of reductions:

- β reduction:

$$\lambda x.M\ N \longrightarrow_{\beta} M\{N/x\}$$

- case reduction:

$$\text{case } c_p j N_j \quad \text{of } c_1 \rightarrow M_1 | ... | c_n \rightarrow M_n \longrightarrow_{\text{case}} M_j\ N_j$$

- letRec reduction:

$$\text{letRec } f = M \longrightarrow_{\text{letRec}} M\{\text{letRec } f = M/f\}$$

- Operator reduction (total functions over terms):

$$opM \rightarrow_{op} [\![op]\!](M)$$

+Left-most outermost reduction strategy

Higher-order interpretations

Interpretations of types

Definition

- $(\!|b|\!) = \bar{\mathbb{N}} = \mathbb{N} \cup \{\top\}$
- $(\!|T \to T'|\!) = (\!|T|\!) \to^{\uparrow} (\!|T'|\!)$

Definition

- $f : A \to^{\uparrow} B$ a strictly monotonic function from A to B:

$$\forall x, y \in A,\ x <_A y \implies f(x) <_B f(y).$$

- $x <_{\bar{\mathbb{N}}} y$ iff $x < y$ or $y = \top$
- $f <_{A \to^{\uparrow} B} g$ iff $\forall x \in A,\ f(x) <_B g(x)$

Example (map: $(A \to B) \to List(A) \to List(B)$)

$(\!|map|\!)$ is in $(\bar{\mathbb{N}} \to^{\uparrow} \bar{\mathbb{N}}) \to^{\uparrow} \bar{\mathbb{N}} \to^{\uparrow} \bar{\mathbb{N}}$.

Lattices

$$\bot_{\bar{\mathbb{N}}} = 0 \qquad \top_{\bar{\mathbb{N}}} = \top$$

$$\bot_{(\!|T \to T'|\!)} = \Lambda X^{(\!|T|\!)}.\bot_{(\!|T'|\!)} \qquad \top_{(\!|T \to T'|\!)} = \Lambda X^{(\!|T|\!)}.\top_{(\!|T'|\!)}$$

$$\sqcup^{(\!|T \to T'|\!)}(F, G) = \Lambda X^{(\!|T|\!)}. \sqcup^{(\!|T'|\!)} (F(X), G(X))$$
$$\sqcap^{(\!|T \to T'|\!)}(F, G) = \Lambda X^{(\!|T|\!)}. \sqcap^{(\!|T'|\!)} (F(X), G(X))$$

Lemma

For any type T, $((\!|T|\!), \leq, \sqcup, \sqcap, \top, \bot)$ is a complete lattice.

Interpretations of terms

$$n \oplus_{\bar{\mathbb{N}}} = \Lambda X.(n + X)$$

$$n \oplus_{(\!|T \to T'|\!)} : \Lambda F.\Lambda X.(n \oplus_{(\!|T'|\!)} F(X))$$

Definition (Interpretations)

$$\begin{array}{rcl} (\!|x|\!) & = & X \\ (\!|c|\!) & = & 1 \oplus (\Lambda X_1. \ldots .\Lambda X_n. \sum_{i=1}^{n} X_i) \\ (\!|M\ N|\!) & = & (\!|M|\!)(\!|N|\!) \end{array}$$

Interpretations of terms

$$n\oplus_{\bar{\mathbb{N}}} = \Lambda X.(n + X)$$

$$n\oplus_{(\!|T \to T'|\!)} : \Lambda F.\Lambda X.(n \oplus_{(\!|T'|\!)} F(X))$$

Definition (Interpretations)

$$\begin{array}{rcl} (\!|x|\!) & = & X \\ (\!|c|\!) & = & 1 \oplus (\Lambda X_1. \dots .\Lambda X_n. \sum_{i=1}^{n} X_i) \\ (\!|M\ N|\!) & = & (\!|M|\!)(\!|N|\!) \\ (\!|\lambda x.M|\!) & = & 1 \oplus (\Lambda (\!|x|\!).(\!|M|\!)) \end{array}$$

Interpretations of terms

$$n \oplus_{\bar{\mathbb{N}}} = \Lambda X.(n + X)$$

$$n \oplus_{(\!|T \to T'|\!)} : \Lambda F.\Lambda X.(n \oplus_{(\!|T'|\!)} F(X))$$

Definition (Interpretations)

$$\begin{array}{rcl} (\!|x|\!) & = & X \\ (\!|c|\!) & = & 1 \oplus (\Lambda X_1. \ldots .\Lambda X_n. \sum_{i=1}^{n} X_i) \\ (\!|M\ N|\!) & = & (\!|M|\!)(\!|N|\!) \\ (\!|\lambda x.M|\!) & = & 1 \oplus (\Lambda (\!|x|\!).(\!|M|\!)) \\ (\!|case\ M\ of\ \ \ldots c_i \to M_i \ldots|\!) & = & 1 \oplus \sqcup_i \{(\!|M_i|\!) R_i \mid (\!|c_i|\!) R_i \leq (\!|M|\!)\} \end{array}$$

Interpretations of terms

$$n \oplus_{\bar{\mathbb{N}}} = \Lambda X.(n + X)$$

$$n \oplus_{(\!|T \to T'|\!)} : \Lambda F.\Lambda X.(n \oplus_{(\!|T'|\!)} F(X))$$

Definition (Interpretations)

$$\begin{array}{rcl} (\!|x|\!) & = & X \\ (\!|c|\!) & = & 1 \oplus (\Lambda X_1. \ldots .\Lambda X_n. \sum_{i=1}^{n} X_i) \\ (\!|M\ N|\!) & = & (\!|M|\!)(\!|N|\!) \\ (\!|\lambda x.M|\!) & = & 1 \oplus (\Lambda (\!|x|\!).(\!|M|\!)) \\ (\!|case\ M\ of\ \ \ldots c_i \to M_i \ldots|\!) & = & 1 \oplus \sqcup_i \{(\!|M_i|\!) R_i \mid (\!|c_i|\!) R_i \leq (\!|M|\!)\} \\ (\!|letRec\ f = M|\!) & = & \sqcap \{F \mid F \geq 1 \oplus (\Lambda (\!|f|\!).(\!|M)|\!) F\} \end{array}$$

Interpretations of terms

$$n\oplus_{\bar{\mathbb{N}}} = \Lambda X.(n + X)$$

$$n\oplus_{(\!|T \to T'|\!)} : \Lambda F.\Lambda X.(n \oplus_{(\!|T'|\!)} F(X))$$

Definition (Interpretations)

$$\begin{array}{rcl}
(\!|x|\!) & = & X \\
(\!|c|\!) & = & 1 \oplus (\Lambda X_1.\ldots.\Lambda X_n.\sum_{i=1}^{n} X_i) \\
(\!|M\ N|\!) & = & (\!|M|\!)(\!|N|\!) \\
(\!|\lambda x.M|\!) & = & 1 \oplus (\Lambda(\!|x|\!).(\!|M|\!)) \\
(\!|case\ M\ of\ \ \ldots c_i \to M_i \ldots|\!) & = & 1 \oplus \sqcup_i \{(\!|M_i|\!) R_i \mid (\!|c_i|\!) R_i \leq (\!|M|\!)\} \\
(\!|letRec\ f = M|\!) & = & \sqcap\{F \mid F \geq 1 \oplus (\Lambda(\!|f|\!).(\!|M)|\!)F\}
\end{array}$$

$$(\!|op\ \mathtt{M}|\!) \geq (\!|[\![op]\!](\mathtt{M})|\!)$$

Properties of interpretations

Theorem

Any term M has an interpretation.

Knaster-Tarski: $\mathit{lfp}(\Lambda X.1 \oplus ((\Lambda (\!| f |\!).(\!| M |\!))X))$

$\rightarrow$ monotonic function over a complete lattice

Lemma

If $M \longrightarrow N$, then $(\!| M |\!) \geq (\!| N |\!)$.

If $M \longrightarrow_{\alpha} N$, $\alpha \neq op$, then $(\!| M |\!) > (\!| N |\!)$.

Lemma

If $M :: B$ and $M \neq \top$ then M terminates in time $O((\!| M |\!))$.

Example of Interpretation

$$(\!|\mathit{letRec}\ \mathit{map} = \lambda g.\lambda x.\mathit{case}\ x\ \mathit{of}\ \mathbf{c}\ y\ z \to \mathbf{c}\ (g\ y)\ (\mathit{map}\ g\ z)|\!)$$

Example of Interpretation

$$(\!| letRec\ map = \lambda g.\lambda x.case\ x\ of\ \mathbf{c}\ y\ z \to \mathbf{c}\ (g\ y)\ (map\ g\ z) |\!)$$

$$= \sqcap \{F \mid F \geq 1 \oplus (\Lambda(\!| f |\!).(\!| \lambda g.\lambda x.case\ x\ of\ \mathbf{c}\ y\ z \to \mathbf{c}\ (g\ y)\ (f\ g\ z) |\!))F\}$$

(*letRec*)

Example of Interpretation

$$(\!|letRec\ map = \lambda g.\lambda x.case\ x\ of\ \mathbf{c}\ y\ z \to \mathbf{c}\ (g\ y)\ (map\ g\ z)|\!)$$

$$= \sqcap \{F \mid F \geq 1 \oplus (\Lambda(\!|f|\!).(\!|\lambda g.\lambda x.case\ x\ of\ \mathbf{c}\ y\ z \to \mathbf{c}\ (g\ y)\ (f\ g\ z)|\!))F\}$$

(*letRec*)

$$= \sqcap\{F \mid F \geq 3 \oplus ((\Lambda(\!|f|\!).\Lambda(\!|g|\!).\Lambda(\!|x|\!).(\!|case\ x\ of\ \mathbf{c}\ y\ z \to \mathbf{c}\ (g\ y)\ (f\ g\ z)|\!))F)\}$$

(*lambda* $\times$ 2)

Example of Interpretation

$$(\!|\mathit{letRec}\ \mathit{map} = \lambda g.\lambda x.\mathit{case}\ x\ \mathit{of}\ \mathbf{c}\ y\ z \to \mathbf{c}\ (g\ y)\ (\mathit{map}\ g\ z)|\!)$$

$$= \sqcap \{F \mid F \geq 1 \oplus (\Lambda(\!|f|\!).(\!|\lambda g.\lambda x.\mathit{case}\ x\ \mathit{of}\ \mathbf{c}\ y\ z \to \mathbf{c}\ (g\ y)\ (f\ g\ z)|\!))F\}$$

(letRec)

$$= \sqcap\{F \mid F \geq 3 \oplus ((\Lambda(\!|f|\!).\Lambda(\!|g|\!).\Lambda(\!|x|\!).(\!|\mathit{case}\ x\ \mathit{of}\ \mathbf{c}\ y\ z \to \mathbf{c}\ (g\ y)\ (f\ g\ z)|\!))F)\}$$

$(\mathit{lambda} \times 2)$

$$= \sqcap\{F \mid F \geq 4 \oplus (\Lambda(\!|f|\!).\Lambda(\!|g|\!).\Lambda(\!|x|\!).\ \sqcup \{(\!|\mathbf{c}\ (g\ y)\ (f\ g\ z)|\!) \mid (\!|x|\!) \geq (\!|\mathbf{c}\ y\ z|\!)\}F)\}$$

(case)

Example of Interpretation

$$(\!|letRec\ map = \lambda g.\lambda x.case\ x\ of\ \mathbf{c}\ y\ z \to \mathbf{c}\ (g\ y)\ (map\ g\ z)|\!)$$

$$= \sqcap \{F \mid F \geq 1 \oplus (\Lambda (\!|f|\!).(\!|\lambda g.\lambda x.case\ x\ of\ \mathbf{c}\ y\ z \to \mathbf{c}\ (g\ y)\ (f\ g\ z)|\!))F\}$$

($letRec$)

$$= \sqcap\{F \mid F \geq 3 \oplus ((\Lambda(\!|f|\!).\Lambda(\!|g|\!).\Lambda(\!|x|\!).(\!|case\ x\ of\ \mathbf{c}\ y\ z \to \mathbf{c}\ (g\ y)\ (f\ g\ z)|\!))F)\}$$

($lambda \times 2$)

$$= \sqcap\{F \mid F \geq 4 \oplus (\Lambda(\!|f|\!).\Lambda(\!|g|\!).\Lambda(\!|x|\!). \sqcup \{(\!|\mathbf{c}\ (g\ y)\ (f\ g\ z)|\!) \mid (\!|x|\!) \geq (\!|\mathbf{c}\ y\ z|\!)\}F)\}$$

($case$)

$$= \sqcap\{F \mid F \geq 4 \oplus (\Lambda(\!|f|\!).\Lambda(\!|g|\!).\Lambda(\!|x|\!). \sqcup \{1 \oplus ((\!|g|\!)\ (\!|y|\!)) + ((\!|f|\!)\ (\!|g|\!)\ (\!|z|\!))$$
$$\mid (\!|x|\!) \geq 1 \oplus (\!|y|\!) + (\!|z|\!)\}F)\}$$

($cons \times 2$)

Example of Interpretation

$(\!|\mathit{letRec}\ \mathit{map} = \lambda g.\lambda x.\mathit{case}\ x\ \mathit{of}\ \mathbf{c}\ y\ z \to \mathbf{c}\ (g\ y)\ (\mathit{map}\ g\ z)|\!)$

$= \sqcap \{F \mid F \geq 1 \oplus (\Lambda(\!|f|\!).(\!|\lambda g.\lambda x.\mathit{case}\ x\ \mathit{of}\ \mathbf{c}\ y\ z \to \mathbf{c}\ (g\ y)\ (f\ g\ z)|\!))F\}$

(letRec)

$= \sqcap\{F \mid F \geq 3 \oplus ((\Lambda(\!|f|\!).\Lambda(\!|g|\!).\Lambda(\!|x|\!).(\!|\mathit{case}\ x\ \mathit{of}\ \mathbf{c}\ y\ z \to \mathbf{c}\ (g\ y)\ (f\ g\ z)|\!))F)\}$

($\mathit{lambda} \times 2$)

$= \sqcap\{F \mid F \geq 4 \oplus (\Lambda(\!|f|\!).\Lambda(\!|g|\!).\Lambda(\!|x|\!). \sqcup \{(\!|\mathbf{c}\ (g\ y)\ (f\ g\ z)|\!) \mid (\!|x|\!) \geq (\!|\mathbf{c}\ y\ z|\!)\}F)\}$

(case)

$= \sqcap\{F \mid F \geq 4 \oplus (\Lambda(\!|f|\!).\Lambda(\!|g|\!).\Lambda(\!|x|\!). \sqcup \{1 \oplus ((\!|g|\!)\ (\!|y|\!)) + ((\!|f|\!)\ (\!|g|\!)\ (\!|z|\!)))$

$\mid (\!|x|\!) \geq 1 \oplus (\!|y|\!) + (\!|z|\!)\}F)\}$

($\mathit{cons} \times 2$)

$= \sqcap\{F \mid F \geq 5 \oplus (\Lambda(\!|g|\!).\Lambda(\!|x|\!). \sqcup \{(((\!|g|\!)\ (\!|y|\!)) \oplus (F\ (\!|g|\!)\ (\!|z|\!)))$

$\mid (\!|x|\!) \geq 1 \oplus (\!|y|\!) \oplus (\!|z|\!)\}\}$

Relaxing interpretations

$$(\!|\mathit{letRec}\ \mathit{map} = \lambda g.\lambda x.\mathit{case}\ x\ \mathit{of}\ \mathbf{c}\ y\ z \rightarrow \mathbf{c}\ (g\ y)\ (\mathit{map}\ g\ z)|\!)$$

$$= \sqcap\{F \mid F \geq 5 \oplus (\Lambda G.\Lambda X. \sqcup \{((G\ Y) \oplus (F\ G\ Z)) \mid X \geq 1 \oplus Y \oplus Z\}\}$$

Relaxing interpretations

$$(\!|letRec\ map = \lambda g.\lambda x.case\ x\ of\ \mathbf{c}\ y\ z \rightarrow \mathbf{c}\ (g\ y)\ (map\ g\ z)|\!)$$

$$= \sqcap\{F \mid F \geq 5 \oplus (\Lambda G.\Lambda X. \sqcup \{((G\ Y) \oplus (F\ G\ Z)) | \ X \geq 1 \oplus Y \oplus Z\}\}$$

$$\leq \sqcap\{F \mid F \geq 5 \oplus (\Lambda G.\Lambda X.((G\ (X-1)) \oplus (F\ G\ (X-1))))\}$$

(constraint upper bound)

Relaxing interpretations

$$(\!| letRec\ map = \lambda g.\lambda x.case\ x\ of\ \mathbf{c}\ y\ z \rightarrow \mathbf{c}\ (g\ y)\ (map\ g\ z) |\!)$$

$$= \sqcap\{F \mid F \geq 5 \oplus (\Lambda G.\Lambda X. \sqcup \{((G\ Y) \oplus (F\ G\ Z)) | \ X \geq 1 \oplus Y \oplus Z\}\}$$

$$\leq \sqcap\{F \mid F \geq 5 \oplus (\Lambda G.\Lambda X.((G\ (X-1)) \oplus (F\ G\ (X-1))))\}$$

(constraint upper bound)

$$\leq \Lambda G.\Lambda X.(5 + G\ X)) \times X$$

(min upper bound)

A characterization of BFF_i

Order

Order 1 $f : \mathbb{N} \to \mathbb{N}$

Order 2 $f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$

Order n $f : \underbrace{(((\mathbb{N} \to \mathbb{N}) \to \mathbb{N})... \to \mathbb{N})}_{n} \to \mathbb{N}$

Formally:

$order(b) = 0 \quad order(T \to T') = \max(order(T) + 1, order(T'))$

Example

map is of order 2.

$apply : f, x \mapsto f(x)$ is of order 2.

$compose : f, g \mapsto (x \mapsto f(g(x)))$ is of order 2.

$isNorm : f \mapsto \begin{cases} 1 & \text{if } f \text{ is a norm} \\ 0 & \text{otherwise} \end{cases}$ is of order 3.

Order

Order 1 $f : \mathbb{N} \to \mathbb{N}$

Order 2 $f : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$

Order n $f : \underbrace{(((\mathbb{N} \to \mathbb{N}) \to \mathbb{N})... \to \mathbb{N})}_{n} \to \mathbb{N}$

Formally:

$order(b) = 0 \quad order(T \to T') = \max(order(T) + 1, order(T'))$

Example

map is of order n.

$apply : f, x \mapsto f(x)$ is of order n.

$compose : f, g \mapsto (x \mapsto f(g(x)))$ is of order n.

$isNorm : f \mapsto \begin{cases} 1 & \text{if } f \text{ is a norm} \\ 0 & \text{otherwise} \end{cases}$ is of order 3.

Higher-order polynomial

$$P_1 ::= c \in \mathbb{N} | X_0 | P_1 + P_1 | P_1 \times P_1$$
$$P_{i+1} ::= P_i | P_{i+1} + P_{i+1} | P_{i+1} \times P_{i+1} | X_i(P_{i+1})$$

Definition

Let FP_i, $i > 0$, be the class of polynomial functionals at order i that consist in functionals computed by closed terms M such that:

- $order(M) = i$
- $(\!|M|\!)$ is bounded by an order i polynomial ($\exists P_i,\ (\!|M|\!) \leq P_i$).

Bounded Typed Loop Program (BTLP)

Definition (BTLP)

A Bounded Typed Loop Program (BTLP) is a non-recursive and well-formed procedure defined by the following grammar:

$$
\begin{aligned}
(\text{Procedures}) \ni P \quad & ::= v^{\tau_1 \times \ldots \times \tau_n \to \mathbb{N}}(v_1^{\tau_1}, \ldots, v_n^{\tau_n}) P^* V I^* \ \textbf{Return}\ v_r^{\mathbb{N}}\ \textbf{End} \\
(\text{Declarations}) \ni V \quad & ::= \textbf{var}\ v_1^{\mathbb{N}}, \ldots, v_n^{\mathbb{N}}; \\
(\text{Instructions}) \ni I \quad & ::= v^{\mathbb{N}} := E; \mid \textbf{Loop}\ v_0^{\mathbb{N}}\ \textbf{with}\ v_1^{\mathbb{N}}\ \textbf{do}\ I^*\ \textbf{EndLoop}\ \textbf{;} \\
(\text{Expressions}) \ni E \quad & ::= 1 \mid v^{\mathbb{N}} \mid v_0^{\mathbb{N}} + v_1^{\mathbb{N}} \mid v_0^{\mathbb{N}} - v_1^{\mathbb{N}} \mid v_0^{\mathbb{N}} \# v_1^{\mathbb{N}} \mid \\
& \qquad v^{\tau_1 \times \ldots \times \tau_n \to \mathbb{N}}(A_1^{\tau_1}, \ldots, A_n^{\tau_n}) \\
(\text{Arguments}) \ni A \quad & ::= v \mid \lambda v_1, \ldots, v_n . v(v_1' \ldots, v_m') \quad \text{with } v \notin \{v_1, \ldots, v_n\}
\end{aligned}
$$

BFF_i is the class of order i functionals computable by a BTLP program.

Results

Define the Safe Feasible Functionals at order i, SFF_i by:

$$SFF_1 = BFF_1,$$
$$\forall i \geq 1,\ SFF_{i+1} = BFF_{i+1 \restriction SFF_i}$$

Theorem (Hainry Péchoux)
For any order i, $FP_i = SFF_i$.

In particular, FP_1 is FPtime and FP_2 is BFF with FPtime oracles.

Conclusion

Conclusion

Results

- An interpretation theory for higher-order functional languages
- A characterization of well-known classes: BFF_i

Issues and future work

- BFF_i is known to be restricted
  $\rightarrow$ see Férée's phD manuscript (2014)
- The interpretation synthesis problem is known to be very hard.
- Interpretations for complexity analysis of real operators and real-based languages.
- Adapt the results to space: does it make sense?
- Adapt ICC techniques to characterize $P(\mathbb{R})$.