Higher order complexity and application in computable analysis

Hugo Férée

Shonan meeting on ICC and applications

UNIVERSITÉ DE LORRAINE
Loria
Laboratoire lorrain de recherche en informatique et ses applications

Background on higher order complexity

Higher order strategies

Higher order Turing machines

Computable analysis

Perspectives

Higher order complexity today:

- ✓ Order 1 ($\mathbb{N} \to \mathbb{N}$) computability and complexity
- ✓ Order 2 ($(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$): Oracle Turing Machines (OTM)
- ✗ Order 3 and above:
  - BFF
  - but it is a far smaller class than the continuous functionals (contrary to orders 1 and 2).
  - no satisfying machine model
  - no general notion of complexity
  - several (incomparable) notions of computability, but one natural and robust class: the Kleene-Kreisel functionals.
- How to define a general and meaningful notion of complexity at all finite types?

Basic Feasible Functionals (BFF)

Definition

PV^ω = simply-typed λ-calculus + PTIME + $\mathcal{R}$

$\mathcal{R}$ is a second order bounded recursion on notation:

$$\mathcal{R}(x_0, F, B, x) = \begin{cases} x_0 \text{ if } x = 0 \\ t \text{ if } |t| \leq B(t) \\ B(t) \text{ otherwise.} \end{cases}$$

$$\text{with } t = F(x, \mathcal{R}(x_0, F, B, \lfloor \frac{x}{2} \rfloor))$$

BFF: functionals computed by closed PV^ω terms.

Example (Irwin, Kapron, Royer)

$$f_x(y) = 1 \iff y = 2^x$$

$$\Phi, \Psi : \overbrace{((\mathbb{N} \to \mathbb{N}) \to \mathbb{N})}^{F} \times \overset{x}{\mathbb{N}} \to \mathbb{N}$$

$$\Phi(F, x) = \begin{cases} 0 & \text{if } F(f_x) = F(f_\infty) \\ 1 & \text{otherwise.} \end{cases}$$

$\Phi \in \mathrm{BFF}$

$$\Psi(F, x) = \begin{cases} 0 & \text{if } F(f_x) = F(f_\infty) \\ 2^x & \text{otherwise.} \end{cases}$$

$\Psi \notin \mathrm{BFF}$

The size issue

Definition (Output size)

If $F : \tau_1 \times \cdots \times \tau_n \to \mathbb{N}$, then $|F| : \tau_1 \times \cdots \times \tau_n \to \mathbb{N}$ and:

$$|F|(t) = \max_{|f| \leq t} |F(f)|$$

Theorem

The output size of every BFF *functional is well-defined.*

The size issue

Example

$$\Gamma(F) = \begin{cases} 0 \text{ if } \forall x, F(f_\infty) = F(f_x) \\ x \text{ minimal such that } F(f_\infty) \neq F(f_x) \text{ otherwise.} \end{cases}$$

$$\forall x, F_x(f) = \begin{cases} 1 \text{ if } f(x) = 1 \\ 0 \text{ otherwise} \end{cases}$$

$$\forall x, |F_x| \leq \mathbf{1}$$

$\Gamma \notin$ BFF since $|\Gamma(F_x)|$ is unbounded while $|F_x|$ is bounded.

Toward a machine model at higher types

Interaction with the argument in an Oracle Turing Machine:

- Machine: what is $f(n)$?
- Oracle: $f(n)$ is v!

Toward a machine model at higher types

Interaction with the argument in an Oracle Turing Machine:

- Machine: what is $f(n)$?
- Oracle: $f(n)$ is v!

- Machine: what is $f(x)$?
- Oracle: what is x?
- Machine: $x = n$!
- Oracle: $f(x) = v$!

Toward a machine model at higher types

Interaction with the argument in an Oracle Turing Machine:

- Machine: what is $f(n)$?
- Oracle: $f(n)$ is v!

- Machine: what is $f(x)$?
- Oracle: what is x?
- Machine: $x = n$!
- Oracle: $f(x) = v$!

Let's generalize this dialogue to all types: a functional is described by the way it interacts with input functionals.

- We first define dialogs as games following strategies.
- We then define HOTM playing such games.

Background on higher order complexity

Higher order strategies

Higher order Turing machines

Computable analysis

Perspectives

Higher order strategies

(Hyland & Ong, Nickau)

Finite types: $\tau = \mathbb{N} \mid \tau_1 \times \ldots \times \tau_n \to \mathbb{N}$

Given a finite type, give a name to each occurrence of $\mathbb{N}$:

$$\overbrace{(\underbrace{(\overset{x}{\mathbb{N}} \to \overset{f}{\mathbb{N}}) \to \overset{F}{\mathbb{N}})}_{\text{Opponent}} \to \overset{\phi}{\mathbb{N}}}^{\text{Player}}$$

Moves: $?^f$ or $!^f(v)$.

Strategy: partial function which given a list of previous moves, outputs a valid move.

Execution tree: tree representation of a strategy.

Examples

- $x = 3$

$$\frac{?^x}{} \; !^x(3)$$

Examples

- $x = 3$
- $f(x) = 2x + 1$

$$\frac{?^x}{\quad} \; !^x(3)$$

$$\frac{?^f}{\quad} \; ?^x \; \frac{!^x(n)}{\quad} \; !^f(2n+1)$$

Examples

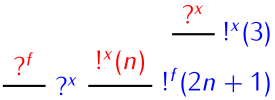
- $x = 3$
- $f(x) = 2x + 1$

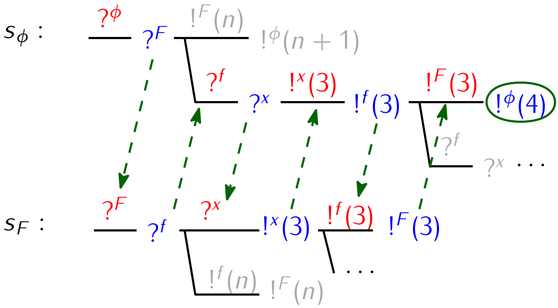
$$\frac{?^x}{}\ !^x(3)$$

$$\frac{?^f}{}\ ?^x\ \frac{!^x(n)}{}\ ?^x\ \frac{!^x(n)}{}\ ?^x\ \frac{!^x(n)}{}\ !^f(2n+1)$$

Examples

- $x = 3$
- $f(x) = 2x + 1$
- $((\overset{x}{\mathbb{N}} \rightarrow \overset{f}{\mathbb{N}}) \rightarrow \overset{F}{\mathbb{N}}) \rightarrow \overset{\phi}{\mathbb{N}}$





$\phi(F) = F(\lambda x.x) + 1$

$F(f) = f(3)$

$s_\phi[s_F] = 4$

Representation of functionals by strategies

Definition

$s_F[s_1, \ldots, s_n] = v$ if the game between s_F and $s_1, \ldots, s_n$ ends with $!_F(v)$.

Remark

A game may not end if:

- at some point a strategy is undefined
- or the dialogue is infinite

Definition

s_F represents $F : \tau_1 \times \cdots \times \tau_n \to \mathbb{N}$ if whenever $s_1, \ldots s_n$ represent $f_1, \ldots f_n, s_F[s_1, \ldots, s_n] = F(f_1, \ldots f_n)$.

Computability and continuity

Definition

- A strategy is computable if it is computable as an order 1 function.
- A function is computable if it is represented by a computable strategy.

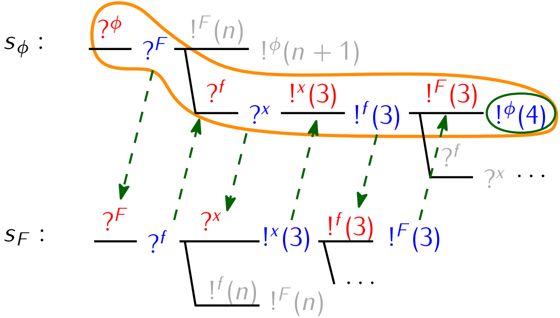
Proposition

A function is (Kleene-Kreisel) continuous if and only if it is represented by a strategy.

Proposition

A function is computable if and only if it is Kleene-Kreisel computable.

Size of a strategy



History: $H(s_\phi, s_F)$

Size of a strategy

Definition (Size)

By induction, the size of a strategy s over type τ is $S_s : \tau$

- If $s = \xrightarrow{?^x} !^x(n)$ then $S_s = |n| + c$.
- $S_s(b_1, \ldots, b_n) = \max_{(s_1, \ldots s_n) \in K_{b_1} \times \cdots \times K_{b_n}} |H(s, s_1, \ldots s_n)|$ with $K_b = \{s' \mid S_{s'} \preccurlyeq b\}$

Example

- $n \in \mathbb{N}$ has a strategy of size $\mathcal{O}(\log_2 n)$.
- $f : \mathbb{N} \to \mathbb{N}$ has a strategy of size $|f|(n) = n + \max_{|x| \leq n} |f(x)|$.
- The size of a strategy for $F : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ is at least its modulus of continuity.

Background on higher order complexity

Higher order strategies

Higher order Turing machines

Computable analysis

Perspectives

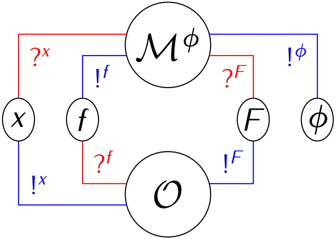
Higher order Turing machines

Definition (HOTM)

A HOTM is a kind of oracle Turing machine which plays a game versus strategies played by oracles.

If $\mathcal{M}^\phi$ computes

$\phi : ((\overset{x}{\mathbb{N}} \to \overset{f}{\mathbb{N}}) \to \overset{F}{\mathbb{N}}) \to \overset{\phi}{\mathbb{N}}$ then $\mathcal{M}^\phi$ has four special states denoted by "x","f","F","ϕ".



Running time of a HOTM: same as for an OTM.

Property

A strategy is computable $\iff$ it is represented by a HOTM.

Polynomial time complexity

Definition (Higher type polynomials)

HTP: simply-typed $\lambda-$calculus with $+, * : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.

Property

HTP *of type* 1 *and* 2 *are respectively the usual polynomials and the second-order polynomials.*

Definition (POLY)

$\phi \in$ POLY if ϕ is computed by a HOTM whose running time is bounded by a HTP.

Remark

- $\text{POLY}_1 = \text{FPTIME}$, $\text{POLY}_2 = \text{BFF}_2$ and $\forall i \geq 3, \text{BFF}_i \subsetneq \text{POLY}_i$.
- We can define other time (or space) complexity classes

Example

$$\Psi(F,x) = \begin{cases} 0 & \text{if } F(f_x) = F(f_\infty) \\ 2^x & \text{otherwise.} \end{cases}$$

The complexity of Ψ is about $\mathcal{F}, n \mapsto c \times \mathcal{F}(P(n))$, where $P(|x|)$ is the complexity of f_x.

Example

$$\Gamma(F) = \begin{cases} 0 \text{ if } \forall x, F(f_\infty) = F(f_x) \\ x \text{ minimal such that } F(f_\infty) \neq F(f_x) \text{ otherwise.} \end{cases}$$

The complexity of Γ is about $\mathcal{F} \mapsto \mathcal{F}(P_\infty) \times \mathcal{F}(P(\mathcal{F}(P_\infty)))$ where P_∞ is the complexity of f_∞.

Higher order polynomial time complexity

- ✓ Inputs: strategies
- ✓ Size of inputs
- ✓ Machine model
- ✓ Running time
- ✓ Polynomial time complexity class

Background on higher order complexity

Higher order strategies

Higher order Turing machines

Computable analysis

Perspectives

Complexity in computable analysis

- Order 1
  - Complexity of real functions (Ko, Friedman)
  - Generalization to σ-compact spaces (Weihrauch, Schröder)
- Order 2 (Kawamura and Cook)
  - Polynomial time complexity based on BFF_2
  - Allows to define notions of complexity over non σ-compact spaces like $\mathcal{C}([0,1],\mathbb{R})$
- Is order 2 always sufficient?

"Feasible" admissibility

Definition (Polynomial reducibility)

$\delta \leq_P \delta'$ if $\delta = \delta' \circ f$ with f polynomial time computable

Theorem (Kawamura & Cook)

$\delta_\square$ is the "largest" representation of $\mathcal{C}([0,1],\mathbb{R})$ making $Eval : \mathcal{C}([0,1],\mathbb{R}) \to [0,1] \to \mathbb{R}$ polynomial time computable.

"Feasible" admissibility

Definition (Polynomial reducibility)

$\delta \leq_P \delta'$ if $\delta = \delta' \circ f$ with f polynomial time computable

Theorem (Kawamura & Cook)

$\delta_\square$ is the "largest" representation of $\mathcal{C}([0,1],\mathbb{R})$ making $Eval : \mathcal{C}([0,1],\mathbb{R}) \to [0,1] \to \mathbb{R}$ polynomial time computable.

$\to$ For which spaces can we do the same?

Question (Kawamura)

For which spaces X the space $\mathcal{C}(X,\mathbb{R})$ admits a (maximal) representation making $Eval : \mathcal{C}(X,\mathbb{R}) \times X \to \mathbb{R}$ polynomial time computable?

First order representations are not sufficient

Theorem

Let X be a Polish space that is not σ-compact. Then there is no representation of $\mathcal{C}(X, \mathbb{R})$ making the time complexity of $Eval_{X,\mathbb{R}} : \mathcal{C}(X, \mathbb{R}) \times X \to \mathbb{R}$ well-defined.

($X = \mathcal{C}([0, 1], \mathbb{R})$ for example)

Lemma

There is no surjective partial continuous function $\phi : (\mathbb{N} \to \mathbb{N}) \to \mathcal{C}(\mathbb{N} \to \mathbb{N}, \mathbb{N})$ bounded by a total continuous function.

Corollary

"Higher order is required to define complexity-friendly representations."

Higher order representations

Definition (Kleene-Kreisel Spaces)

$KKS = [\mathbb{N}, \subseteq, \rightarrow, \times]$

Definition (Representation)

A representation δ of a space X with a KKS A is a surjective function from A to X.

Definition (Polynomial reduction)

$\delta_1 \leq_P \delta_2$ if $\delta_1 = \delta_2 \circ F$ for some polynomial time computable $F : A_1 \rightarrow A_2$.

Standard representation of $\mathcal{C}(X, Y)$

Definition

$\delta_{\mathcal{C}(X,Y)}(F) = f$ whenever $f \circ \delta_X = \delta_Y \circ F$

Property

$Eval : \mathcal{C}(X, Y) \times X \to Y$ is polynomial-time computable w.r.t. $(\delta_{\mathcal{C}(X,Y)}, \delta_X, \delta_Y)$

Theorem

It is the largest representation making Eval polynomial.

Background on higher order complexity

Higher order strategies

Higher order Turing machines

Computable analysis

Perspectives

Perspectives

- We have a robust definition of higher order complexity.
- This gives us new representation spaces.
- Some spaces can now be well represented.

- We need to understand the boundaries of the class of polynomial time functionals.
- Make further comparisons with BFF.
- Give implicit characterizations (e.g. function algebra like PV^{ω}).
- Study the extension of TTE with these new representations (e.g. admissibility)
- Find applications in other domains.