Game Semantics Approach to Higher-Order Complexity

Hugo Férée

September 2, 2016
LCC 2016

Introduction

- Computability and complexity are defined for first-order functions (i.e. $\Sigma^* \to \Sigma^*$) in many different ways
- Same for second-order functions,e.g. $(\Sigma^* \to \Sigma^*) \to \Sigma^*$ (to a lesser extent)
- For third-order functions and above?
    - a few incomplete attempts to define complexity
    - not even a unique notion of computability

Motivations

- First-order is sufficient to define computability/complexity for countable sets (e.g. integers, finite graphs, matrices, etc.)
- Second-order is required for computations over some uncountable sets (e.g. real numbers)
- Higher orders are required for some "larger spaces", especially for complexity
- Higher-order functions appear naturally in functional programming languages

First-order complexity

Several computational models, equivalent for complexity:

Machine models:

Complexity = a bound on running time given
a bound on the input size

Function algebras:

Cobham 1965 : bounded recursion on notation

Others

Logic-based (linear logic), program
interpretations, etc.

Second-order complexity

Definition (Mehlhorn (1976))

$\text{FPTIME}_2 = [\text{FPTIME}, \text{Application}, \text{Composition}, \text{Extension}, \mathcal{R}]$

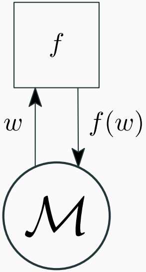
$$\mathcal{R}(x_0, F, B, x). \begin{cases} x_0 \text{ if } x = 0 \\ t \text{ if } |t| \leq B(x) \\ B(x) \text{ otherwise.} \end{cases}$$

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

Theorem (Kapron & Cook 1996)

FPTIME_2 is the class of functions computed by an oracle Turing machine in second-order polynomial time.

Oracle Turing machine



Definition

$F : (\Sigma^* \to \Sigma^*) \to \Sigma^*$ is computed by an oracle Turing machine $\mathcal{M}$ if for any oracle $f : \Sigma^* \to \Sigma^*$, $\mathcal{M}^f$ computes $F(f)$.

Second order complexity

Definition (Time complexity)

The complexity of a machine is an upper bound on its computation time w.r.t the size of its input.

- ✓ size of a finite word
- ? size of an order 1 function

Second order complexity

Definition (Time complexity)

The complexity of a machine is an upper bound on its computation time w.r.t the size of its input.

- ✓ size of a finite word
- ? size of an order 1 function

Definition (Size of a function)

The size of $f : \Sigma^* \to \Sigma^*$ is $|f| : \mathbb{N} \to \mathbb{N}$:

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

Second order polynomial time

Definition (Second order polynomials)

$$P := c \mid X \mid Y\langle P\rangle \mid P + P \mid P \times P$$

Example

$$P(X, Y) = (Y\langle X \times Y\langle X + 1\rangle\rangle)^2$$

Definition ($\textsc{fptime}_2$)

A second order function is computable in polynomial time if it is computed by an oracle Turing machine whose running time is bounded by a second order polynomial.

Some higher order models

- Kleene schemata
- Kleene associates
- Berry-Curien sequential algorithms
- …
- PCF (Scott, Plotkin)
  λ-calculus over $\mathbb{N}$ + fixpoint combinator.
  ✗ No simple underlying complexity notion.
- BFF (Cook, Urquhart)
  λ-calculus + FPTIME + $\mathcal{R}$ (2^{nd}-order bounded recursion)
  ✗ Defines only one complexity class (no EXPTIME, etc.)
  ✗ Misses some intuitively feasible functionals.

Example (Irwin, Kapron, Royer)

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

$$\Phi, \Psi : ((\mathbb{N} \to \mathbb{N}) \to \mathbb{N}) \times \mathbb{N} \to \mathbb{N}$$

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

$\Phi \in \textsc{bff}_3$

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

$\Psi \notin \textsc{bff}_3$, but Ψ is "as feasible as" Φ.

The size issue

Main issue: No notion of size for functions of order 2 and above

In particular, for second-order, the modulus of continuity should be taken into account

Solution:

- A model where the interaction between machine and input is a dialogue.
- Size $\simeq$ "length" of the dialogue.

Computation = dialogue

O: What is $f(x)$?

P: What is x?

O: x equals 4.

P: $f(x)$ equals 2.

Figure 1: Dialogue between a Turing machine (P) computing f and its opponent (O) computing x.

Computation = dialogue

O: What is $F(f)$?

P: What is $f(x)$?

O: What is x?

P: x equals 4.

O: $f(x)$ equals 2.

$\ldots$

P: What is $f(x)$?

O: What is x?

P: x equals 2.

O: $f(x)$ equals 9.

P: $F(f)$ equals 16.

Figure 1: Dialogue between an OTM (P) computing F and its opponent (O) computing f.

Computation = dialogue

O: What is $\Phi(F)$?

Second order dialogue ($F(f_1)$)

$\vdots$

Second order dialogue ($F(f_k)$)

P: $\Phi(F)$ equals 11.

Figure 1: Dialogue between an order 3 machine (P) computing Φ and its opponent (O) computing F.

Game Semantics

Origin: provide a fully abstract semantics for PCF

Solution: (Hyland & Ong, Nickau, Abramsky):

- functions $\leftrightarrow$ strategies
- function application $\leftrightarrow$ confrontation of strategies

Game Semantics

Origin: provide a fully abstract semantics for PCF

Solution: (Hyland & Ong, Nickau, Abramsky):

- functions $\leftrightarrow$ strategies
- function application $\leftrightarrow$ confrontation of strategies

Not made with effectivity/complexity in mind, although Nickau mentionned it.

Arena

An arena is a set of(player and opponent) moves (questions or answers) as well as a subset of initial questions and an enabling relation.

The answers enabled by an initial question are called final answers.

Arenas for finite types

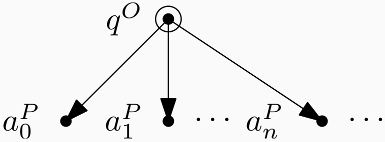


Figure 2: Arena for the base type $\mathbb{N}$

Arenas for finite types

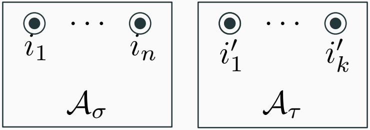


Figure 2: Arena $\mathcal{A}_{\sigma\times\tau}$ built from $\mathcal{A}_\tau$ and $\mathcal{A}_\sigma$

Arenas for finite types

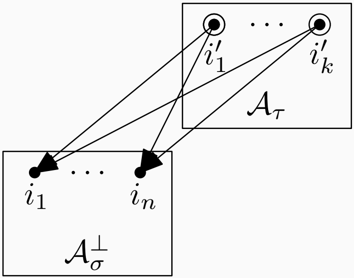


Figure 2: Arena $\mathcal{A}_{\sigma\to\tau}$ built from $\mathcal{A}_\tau$ and $\mathcal{A}_\sigma$

Arenas for finite types

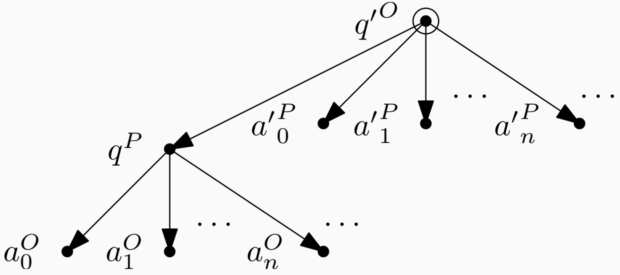


Figure 2: Arena for type $\mathbb{N} \to \mathbb{N}$

Arenas for finite types

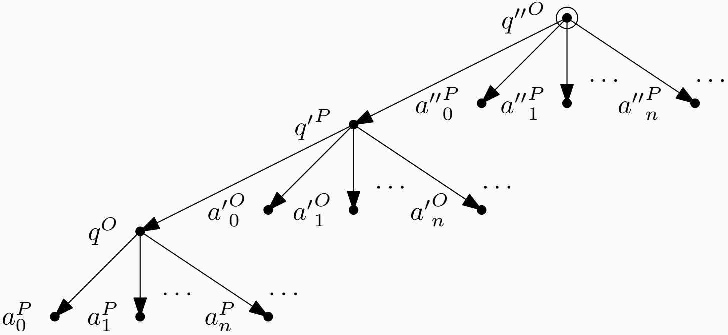


Figure 2: Arena for type $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$

Plays & Rules

Definition (Play)

A play is a list of named moves, i.e. $m[\alpha]$ ($m \in \mathcal{A}, \alpha \in \mathbb{N}$).

A play p is said to be:

- justified: if every non initial move is justified by a previous move in p ;
- well-opened: if there is only one initial move, at the beginning of p ;
- alternating: if two consecutive moves belong to different protagonists ;
- strictly scoped: if answering a question prevents further moves to be justified by this question ;
- strictly nested: if question/answer pairs form a valid bracketing.

Innocent Strategies

Definition (Strategy)

A strategy is a partial function from plays to moves.

Definition (Innocent strategy)

A strategy is innocent if its output only depends on its current view of the play.

Confrontation

If $\tau = \tau_1 \times \dots \tau_n \to \mathbb{N}$ and $s, s_1, \dots s_n$ are strategies on the arenas $\mathcal{A}_\tau, \mathcal{A}_{\tau_1}, \dots \mathcal{A}_{\tau_n}$, then the confrontation of s agains $s_1, \dots, s_n$ is defined this way:

- the play p starts with the initial question of $\mathcal{A}_\tau$
- the confrontation stops when s plays a final answer
- the play is successively extended this way:
  - if s is defined on p, then p is extended with $s(p)$
  - if $s(p)$ is not a final answer, it belongs to one of the $\mathcal{A}_i$ and p contains a play p_i in $\mathcal{A}_i$
  - the play is extended with $s_i(p_i)$ (+renaming)

The final play $H(s, s_1, \dots, s_n)$ is the history of the confrontation, and this defines a partial function $s[]$ from argument strategies to final answers.

Games for PCF

Given a finite type τ, the corresponding game is defined by innocent strategies playing justified, alternating, well-opened, strictly-nested, … plays in the arena $\mathcal{A}_\tau$.

Definition

Such a strategy s represents $F : \tau_1 \times \cdots \times \to \mathbb{N}$ if whenever $s_1, \ldots, s_n$ represent $f_1 : \tau_1, \ldots, f_n : \tau_n$, then

$s[s_1, \ldots, s_n]$ represents $F(f_1, \ldots, f_n)$

Effectivity

Our presentation of game semantics allows to define an explicit encoding of moves and names: for every game on a finite type τ,

- an answer of the form a_n (i.e. representing $n \in \mathbb{N}$) can be encoded by a binary word of size $\mathcal{O}(\log_2(n))$;
- the questions can be encoded by words of bounded size ;
- names are integers $\rightarrow$ usual binary encoding ;
- this encoding can be extended to plays ;
- a strategy s can be represented by a partial function $\overline{s}$ of type $\Sigma^* \rightarrow \Sigma^*$

Computability and complexity

Definition

A strategy is s is computable if $\overline{s}$ is computable.

Computability and complexity

Definition

A strategy is s is computable if $\overline{s}$ is computable.

Definition attempt

A function is computable in time t, if it is represented by a strategy s such that $\overline{s}$ is computable in time t.

Computability and complexity

Definition

A strategy is s is computable if $\bar{s}$ is computable.

Definition attempt

A function is computable in time t, if it is represented by a strategy s such that $\bar{s}$ is computable in time t.

Theorem

Every computable function has a polynomial strategy.

Proof.

s can gain time by asking many useless questions. □

Computability ~~and complexity~~

Definition

A strategy is s is computable if $\bar{s}$ is computable.

~~**Definition attempt**~~

~~*A function is computable in time t, if it is represented by a strategy s such that $\bar{s}$ is computable in time t.*~~

~~**Theorem**~~

~~*Every computable function has a polynomial strategy.*~~

~~**Proof.**~~

~~s can gain time by asking many useless questions.~~ □

Size of a strategy

Definition (Size of a play)

$=$ size of its binary encoding.

Definition (Size of a strategy)

The size S_s of a strategy s in the game of type $\tau = \tau_1 \times \cdots \times \tau_n \to \mathbb{N}$ is a bound on the size of the play H produced by the confrontation of s versus argument strategies:

$$S_s(b_1, \ldots, b_n) = \sup\{|H(s, s_1, \ldots s_n)| \;:\; \forall i, S_{s_i} \preccurlyeq_{\tau_i} b_i\}$$

Additionally, for all $F, B : \tau$, $F \preccurlyeq_\tau B$ if:

$$(\forall i, S_{s_i} \preccurlyeq_{\tau_i} b_i) \implies F(S_{s_1}, \ldots, S_{s_n}) \leq B(b_1, \ldots, b_n)$$

Examples

Example

- $k \in \mathbb{N}$ has a strategy of size about $\log_2(k)$
  (plays are of the form: $q, a_k[0]$)
- $g : \mathbb{N} \to \mathbb{N}$ has a strategy of size about
  $|g|(n) = \max_{|x| \leq n} |g(x)|$
  (plays are of the form: $q, q'[0], a'_x[1], a_{g(x)}[0]$)
- $F : (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ has a strategy whose size depends on its values and on its modulus of continuity.

Game machines

Definition (Game machine)

OTM which simulates a strategy:

- initial state $\leftrightarrow$ initial question
- oracle call $\leftrightarrow$ (encoded) player move
- oracle answer $\leftrightarrow$ (encoded) opponent move
- final state + tape's content $\leftrightarrow$ final answer

Proposition

s is simulated by a game machine $\iff$ s is computable.

Complexity

We can define the complexity of a strategy, and in particular:

$$\text{size} \preccurlyeq \text{complexity}$$

$$\text{size} \simeq \text{smallest relativized complexity}$$

Definition

$f \in$ PCF is computable in time T if there is a game machine simulating an innocent strategy for f in time T.

Remark

If s represents a PCF *function $f : \tau$, then the size and complexity functions for s have type τ.*

Higher order polynomials

Definition (Higher type polynomials)

HTP $=$ simply-typed λ-calculus, with $+$ and $\times$.

Remark

- *Order 1* HTP *= usual polynomials.*
- *Order 2* HTP *= second order polynomials.*

Higher order polynomial time complexity

Definition (POLY)

$f \in \text{PCF}$ is polynomial time computable ($f \in \text{POLY}$), if it has a strategy computed by a (higher order) polynomial time machine.

Results

Proposition

For every finite type τ, the complexity of the identity function of type $\tau \to \tau$ is about $\lambda b.2 \cdot b$.

Similarly, composition, projections and expansion also have polynomial time complexity.

Remark

If $b : \sigma$ and $B : \sigma \to \tau$ bound the complexity (resp. size) of $f : \sigma$ and $F : \sigma \to \tau$, then $B(b)$ bounds the complexity (resp. size) of $F(f)$.

Results

Proposition

Bounded recursion on notation is polynomial-time computable.

Proof.

It can be computed by $|x|$ iteration of F applied to x an input bounded by the size of B on x. Its complexity is bounded by:

$$\lambda n_0 \lambda G \lambda H \lambda n. \quad n \cdot G(H(n, B(n) + n_0)) + n_0. \qquad \square$$

Results

Theorem

- $\mathit{FPTIME} = \mathrm{BFF}_1 = \mathit{POLY}_1$
- $\mathit{FPTIME}_2 = \mathrm{BFF}_2 = \mathit{POLY}_2$
- $\mathrm{BFF} \subseteq \mathit{POLY}$
- $\mathrm{BFF}_3 \subsetneq \mathit{POLY}_3$ *(cf. example* Ψ*)*
- *POLY is stable by composition*

$\implies$ this complexity class is a good candidate for a generalization of FPTIME at all finite types.

Summary

We have defined:

- a notion of size for "PCF strategies" ;
- a machine model adapted to games ;
- a notion of complexity for PCF functions ;
- a polynomial time computable class for PCF.

This class verifies all the expected properties.

The notion of complexity is generic enough to define other classes like:

- exponential time
- sub-linear time (with smaller names?)
- space/non-deterministic complexity
- query/communication/... complexity

Extension to more games

These notions extend to other kind of games as soon as:

- names, moves and plays have a binary encoding
  ($\implies$ countable arena)
- the confrontation can be defined:
  - finite-depth acyclic arena
  - well-opened, alternating, justified plays

In particular, strictly-nested plays or innocent strategies are not required to define size and complexity.

Further work

- Further justify the relevance of POLY (e.g. provide characterizations)
- Study other (deterministic time) complexity classes
- Do higher-order classes help shed light on first-order classes?
- New complexity notions exclusive to higher-order
- Are there other meaningful "sequential games" than those for PCF?
- Generalize to other games, e.g. :
    - arenas with cycles $\implies$ inductive/co-inductive types
    - no alternating rule $\implies$ parallelism