

Towards a *Complexity-through-Realizability* Theory

Thomas Seiller

Proofs, Programs, Systems – Paris 7 University

DICE/FOPARA

April 12th, 2015

Contents of the talk

Present a new approach to (implicit) computational complexity theory.

Slogan

There is a correspondence between complexity constraints and algebras.

Two important motivations:

- obtain a uniform mathematical approach to CT
- gain new proof methods from mathematics

Contents of the talk

Present a new approach to (implicit) computational complexity theory.

Slogan

There is a correspondence between complexity constraints and (some kinds of) algebras (modulo isomorphisms).

Two important motivations:

- obtain a uniform mathematical approach to CT
- gain new proof methods from mathematics

Related work

Related works in ICC:

- Constrained Linear Logics (Girard, Baillot, Mazza, etc.);
- Quantitative Realizability (Hofmann-dal Lago, Brunel);
- Parsimonious Lambda-calculus (Mazza, Terui, etc.);
- GoI techniques in ICC (Aubert, Bagnol, Pistone, Seiller).

Where does it comes from?

It comes from the interplay between two lines of work:

- Interaction Graphs. A quantitative version of Girard's Geometry of Interaction (GoI);
- Recent results using GoI techniques to characterise complexity classes (joint work with Aubert, Bagnol, Pistone).

Warning

GoI is an involved framework. Quantitative semantics are much more involved constructions than quantitative models. Turns out that... mixing the two do **not** make things simpler.

However, it is an extremely rich theory.

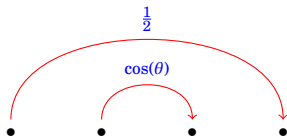
Basics

Basic hypothesis: Algorithms as graphings. So... what's a graphing?

Basics

Basic hypothesis: Algorithms as graphings. So... what's a graphing?

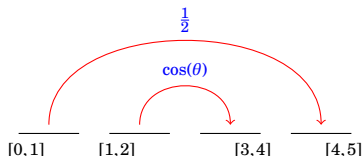
- Take your favourite directed graph, and put weights on the edges.



Basics

Basic hypothesis: Algorithms as graphings. So... what's a graphing?

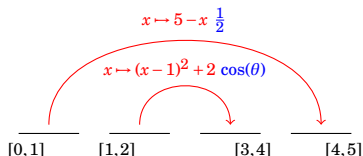
- Take your favourite directed graph, and put weights on the edges.
- Then, consider that vertices are measurable sets, e.g. intervals.



Basics

Basic hypothesis: Algorithms as graphings. So... what's a graphing?

- Take your favourite directed graph, and put weights on the edges.
- Then, consider that vertices are measurable sets, e.g. intervals.
- Finally, decide *how* the edges maps their source to their target.



- A microcosm $\mathfrak{m}$ is a monoid of measurable maps (for the composition law) describing a set of computational principles;

Definition (Graphing)

Let $\mathfrak{m}$ be a microcosm, Ω a monoid of weights, V^G a measurable subset of $\mathbf{X}$. A Ω -weighted $\mathfrak{m}$ -graphing G of support V^G is given by:

- a set of edges E^G ;
 - for each edge $e \in E^G$, source $s^G(e)$ and target $t^G(e)$ measurable subsets of V^G ;
 - for each edge $e \in E^G$, a *realiser* $\phi_e^G \in \mathfrak{m}$ such that $\phi_e^G(s^G(e)) = t^G(e)$;
 - for each edge $e \in E^G$, a *weight* $\omega^G(e)$.
- One defines, on the set of all Ω -weighted $\mathfrak{m}$ -graphing, a notion of execution. We therefore obtain an (untyped) model a computation.

Full models

- One can then define types *a posteriori*, using realizability techniques. Intuitively, say that a graphing is of type $\mathbf{Nat} \Rightarrow \mathbf{Nat}$ if the execution of G with an element $N \in \mathbf{Nat}$ results in an element of $\mathbf{Nat}$.
- This yields a model $\mathbb{M}[\Omega, m + n]$ of (at least!) multiplicative-additive linear logic MALL.
- We then consider models in which one can define (**very** weak) exponential connectives. Very weak in the sense that restrictions on the microcosm yields models where only some principles – or even none – governing exponential connectives hold.
- The approach, in this respect, can be thought of as a semantic version of contained linear logics, although the freeing from syntax makes it more flexible.

Varying the parameters of the construction and/or on graphings, we obtain a rich hierarchy of models.

Constraints on parameters

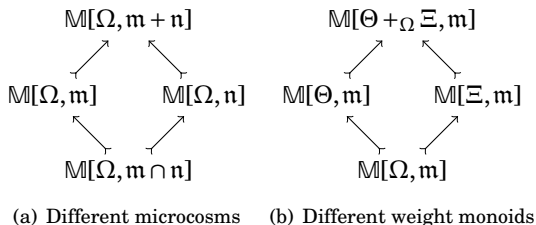


Figure: Inclusions of models

Constraints on graphings

Moreover, the quantitative aspects of the models allow the representation of several computational paradigms.

Example (Probabilistic graphings)

A Ω -weighted graphing G is *probabilistic* when:

- for all $e \in E^G$, $\omega_e^G \in]0, 1]$;
- the following set is of null measure:

$$\{x \in \mathbf{X} \mid \sum_{e \in E^G, x \in s^G(e)} \omega_e^G \neq 1\}$$

Constraints on graphings

Moreover, the quantitative aspects of the models allow the representation of several computational paradigms.

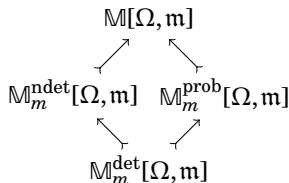


Figure: Different constraints on graphings

First results: Deterministic Models

Method: Consider an element A of the type $!\mathbf{Words}_{\{0,1\}}^{(2)} \multimap \mathbf{Bool}$. Then A defines a language defined as $\{w \mid A :: [w] = \text{true}\}$. Thus, the above type defines a complexity class depending (in particular) on m .

Theorem

There exists microcosms $m_1, m_2, \dots, m_\infty$ such that m_k characterizes the class of languages accepted by deterministic multihead automata with k heads (∞ means finite but unbounded).

First results: Non-Deterministic and Probabilistic Models

Consider an element A of the type $!\mathbf{Words}_{\{0,1\}}^{(2)} \multimap \mathbf{NBool}$. How can one define a language from this?

- Pick a *test* T , i.e. a set of objects in the model;
- Then A defines a language w.r.t. T defined as $\{\mathfrak{w} \mid A :: [\mathfrak{w}] \perp T\}$, where $\perp$ is an orthogonality relation used to define types.
- The above type then defines a complexity class w.r.t. T depending (in particular) on m .

First results: Non-Deterministic and Probabilistic Models

Theorem

There exists tests $T_{\text{may}}, T_{\text{must}}, T_{\text{prob}}$ such that $\mathfrak{m}_k$ characterizes:

- w.r.t. T_{may} , the class of languages accepted by non-deterministic multihead automata with k heads.*
- w.r.t. T_{must} , the complementary of the class of languages accepted by non-deterministic multihead automata with k heads.*
- w.r.t. T_{prob} , the class of languages accepted by probabilistic multihead automata with k heads.*

Summary

Microcosm	$M_m^{\det}$	M_m^{ndet}	M_m^{ndet}	M_m^{prob}	Logic	Machines
m_1	REG	REG	REG	STOC	MALL	Automata
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
m_k	D_k	N_k	coN_k	P_k	(...)	k -heads Aut.
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
m_∞	L	NL	CONL	PL	(...)	multihead-head Aut.
n	P	P	P	??	(...)	+ Pushdown Stack
p	P	NP	coNP	??	ELL	Ptime TM

A notion of compilation

Definition

A m is *compilable* in a microcosm n , written $m < n$ if every element $m \in m$ is almost everywhere equal to a finite union of restrictions of elements n , up to a Borel automorphism of the underlying space.

I.e. for every $m \in m$ there exists a finite partition $X_0, \dots, X_n$ of X and a Borel automorphism $\theta : X \rightarrow X$ such that the restriction of $\theta \circ m$ to X_i coincides (a.e.) with the restriction of an element n_i in n .

Definition

We write $\equiv$ the induced equivalence relation: $m \equiv n$ if $m < n$ and $n < m$.

The Conjecture

Pick a type of graphings (e.g. probabilistic), and write $\text{Pred}(m)$ the set of languages accepted by elements of $!\mathbf{Words}_{\{0,1\}}^{(2)} \multimap \mathbf{Bool}$.

Theorem

If $m \equiv n$, then $\text{Pred}(m) = \text{Pred}(n)$.

Conjecture

The converse holds, i.e. $\text{Pred}(m) = \text{Pred}(n)$ implies $m \equiv n$.

If this conjecture holds, it would provide new proof techniques for separation through (co)homological invariants, e.g. $\ell^{(2)}$ -Betti numbers:

$$\text{Pred}(m) = \text{Pred}(n) \Rightarrow m \equiv n \Rightarrow \ell^{(2)}(m) = \ell^{(2)}(n)$$

Future work

Lots of future work!

- Other complexity classes, e.g. NC_1 , PSPACE, EXP.
- Classes of functions, higher-order functionals, etc.
- Other computational paradigms, e.g. concurrent computation, cellular automata.
- Other approaches of complexity, e.g. term-rewriting (topological graphings),
- Prove the conjecture!