# Functionals using bounded information and the dynamics of algorithms

**Serge Grigorieff** & **Pierre Valarcher**

Université Paris 7 Université Paris-Est

*LICS 2012, Dubrovnik (Croatia), June 2012*

# Functionals using bounded information and the dynamics of algorithms

The framework of this talk is

1. the operational theory of computations
2. the Abstract State Machines computation model developed by Yuri Gurevich.

In the first part of this talk, we shall review this framework.
This will explain what is meant by "dynamics of algorithms"

Then we shall present our contribution.

# Algorithms & computation models

| | how data is structured | how data evolves |
|---|---|---|
| Turing mach. (1936)<br>How Gandy sees it: | infinite tape<br>$\equiv$ growing array | test & set program |
| Kolmogorov (1953)<br>Schönhage (1970) | dynamic graph<br>undirected, bounded degree<br>directed, bounded fan-out | test & set program |
| Cook & Reckow (1973)<br>RAM | Indirect access | test & set program |
| Gurevich (1984)<br>Abstract State Mach.<br>(aka Evolving Algebras) | Algebra on a multisort domain (= finite tuple of partial functions) | test & set program |

ultimate extension

# Algorithms & computation models

Out of the long list of existing computation models the table picks out four breakthroughs. They are turning points in the operational theory of computation.

Do not be outraged that lambda-calculus is not in our pick. We are just following how Gurevich came to ASMs.

Two features emerge

1. The data structure gets more and more involved.
   The last passage to algebras is a crucial move and an ultimate one, indeed.
2. On the opposite, programs keep very elementary: in imperative programming, this is conditional and assignment
   Of course, there is no loop in these programs since a loop instruction is NOT a single action. In fact, the loop is ousted in the computation run obtained as the repeated execution of the program (each execution gives one computation step).

# Three aspects of an ALGORITHM

| What is computed? | What is the detailed computation? | What is the computation rule? |
|---|---|---|
| denotational | operational | descriptional |
| Computable function | Gurevich 1st & 2d Postulates for algo.: $\Longrightarrow$ it is an Algebra<br>Sequence of tuples of partial functions over a multisort domain | Program<br>+<br>Primitive Operations<br>($\equiv$ Oracles)<br>Algorithm does NOT tell how they are computed |
| Semantics | Semantics | Syntax+Semantics |

| | |
|---|---|
| TURING MACHINE | Algebra = state, head position, tape contents<br>Primitive Oper. = read, write, move head, change state |
| LAMBDA CALCULUS | Primitive operation: substitution |
| MATRIX PRODUCT | Primitive operations: $+$, $\times$ on scalars |
| BSS MACHINE | null test on $\mathbb{R}$ is a primitive oper. Uncomputable! |

# Three aspects of an ALGORITHM

These three natural questions seem to exhaust the question

"What is an algorithm?"

---

The answer to the 1st question is well-known. That to the 2d question follows Gurevich's "Sequential time" and "Abstract state" Postulates. The answer to the 3rd question mixes syntax and semantics and is kind of a surprise. The reason is that algorithms are intrinsically ORACULAR. A priori, we expect them to tell everything about how the computation is done. This is false. They describe a mechanism of computation which uses some primitive operations to achieve its basic actions. The sole thing which is known about these primitive operations is their semantics. But the algorithm (that is its program) tells nothing about how the primitive operations are computed. Explaining the operationality of a primitive operation would introduce simpler primitive operations and so on...

---

In some computation models the basic actions are so simple that the primitive operations are simply ignored. Nevertheless, there do exist: cf. Turing machines. The last example, the Blum-Shub-Smale machine has a NON computable primitive operation.

# Denotational vs Operational Completeness

## What Complexity Theory (and other topics...) tells us

Operational strength discriminates among computation models with equal denotational strength

*(Palindrome recognition time discriminates 1-tape and 2-tape TM)*

Small-step = uniformly bounded work at each computation step

Small-step Operational completeness of ASMs

## Gurevich's Abstract State Machines Thesis

**Computations of small-step algorithms are matched step-by-step by ASMs**

## Lambda-calculus (benignly enriched) is operationally complete (Ferbus & SG, 2010)

Lambda-calculus (benignly enriched) matches ASMs
$k$ reductions to match 1 step ($k$ depends on the simulated algo.)

# Denotational vs Operational Completeness

As expected, comparison with respect to operational strength is much finer than comparison with respect to denotational strength. There are several ways to witness this fact. Complexity theory is a very powerful one.

---

Denotationally, there is a strongest computation model. Lots of them in fact. So, what for the operational strength? A bit of caution. Algorithm is a generic term which covers many different meanings: small-step, parallel, distributive, quantum,... So let us restrict to small-step algorithms.

The ASM Thesis is analogous to Church Thesis. Gurevich (& al.) checked that algorithms from all existing small-step computation models can be FAITHFULLY simulated by ASMs. FAITHFULLY means that the simulated environment is (up to isomorphism) that of the ASM and one transition is simulated by one ASM step.

---

Lambda-calculus is enriched by constants which represent the diverse primitive operations (and primitive constants) plus some reduction rules. The increased power of such a Lambda calculus is benign because the added reduction rules apply only to evaluate GROUND terms built with these new constants

# Proving the ASM Thesis

**Theorem (Gurevich, 1999)**

ASM Thesis proved from 3 Postulates

1. "State Transition System" — already considered
2. "States are Algebras" — already considered
3. "Bounded exploration":
   basic actions are defined via terms

We improve this theorem

by replacing the 3rd Postulate by a "lighter" one which is purely semantical

# Proving the ASM Thesis

Of course, the ASM Thesis cannot be proved from scratch. Some POSTULATES are required.

We improve Gurevich's theorem by removing the syntactical contents of the "Bounded exploration" Postulate.

# TRANSITION FUNCTIONALS

| What is computed? | What is the detailed computation? | What is the computation rule? |
|---|---|---|
| denotational | operational | descriptional |
| Computable function | Sequence of tuples of partial functions on multisort domain | Program + Primitive Operations |
| | | Syntax+Semantics |

$\mathbf{T} = \mathbf{T}_1 \times \ldots \times \mathbf{T}_\ell$

$\mathbf{T_i}$ set of partial functions over the sorts

**TRANSITION FUNCTIONAL**

$$\mathbf{\Psi : T \longrightarrow T}$$
$$(f_1, \ldots, f_\ell) \longmapsto (f'_1, \ldots, f'_\ell)$$

WHAT ARE THESE TRANSITION FUNCTIONALS?

(for small-step algorithms)

**This is the subject of this talk**

# TRANSITION FUNCTIONALS

Let us go back to our second slide.
The answer to the third question is a mix of syntax and semantics.

We can remove the syntactic part by looking at the transition functional.
This functional maps a tuple of partial functions over the sorts of the multidomain data structure to another such tuple

This transition functional can be viewed as an answer to both the operational and descriptional aspects of an algorithm

What are the properties of these transition functionals?
Can we characterize them?
This is the subject of this talk.

# Characterizing transition functionals

## Our contribution (for small-step algorithms)

- Proof of the ASM Thesis from set-theoretical conditions on the transition functionals
- Topological characterization of transition functionals

We present our results for
transition functional restricted to total functions
(assuming that total functions are mapped to total functions)
and assume countable sorts

## But our results...

extend to transition functionals on partial functions

# Characterizing transition functionals

The transition functional maps tuples of partial functions to tuples of partial functions.
Why do we restrict to TOTAL functions?

- This restriction is NOT necessary. The general case with partial functions is also relevant to analogous topological methods
- However, the case of "total functions" is by far much simpler to present. It avoids adding to the (already not so simple) proof the notions relevant to the consideration of partial functions.
- Let us say how things are modified in the case of partial functions.
  1. The usual total function spaces with their Hausdorff topology have to be replaced by their non Hausdorff (but $T_0$) Scott domain avatar.
  2. The Vuillemin & Milner notion of sequentiality index comes in

Why do we restrict to countable sorts? Useful only for effective versions.
Non effective results do not need any countability assumption

# Two properties of transition functionals $\Psi(f_1, \ldots, f_\ell) = (f'_1, \ldots, f'_\ell)$ (small-step algorithms)

| Small-step algorithms | do | bounded work at each step |
|---|---|---|

$\Downarrow$

## Bounded Effect property

$\exists k \quad \forall \vec{f} \quad f'_i$ is $f_i$ modified at $\leq k$ points $\qquad i = 1, \ldots, \ell$

## Bounded Cause property

$\exists k \quad \forall \vec{f} \quad \forall \vec{x} \quad \Big| \quad f'_i(\vec{x})$ depends on the values of $f_1, \ldots, f_\ell$ on $\leq k$ points $\qquad i = 1, \ldots, \ell$

# Two properties of transition functionals $\Psi(f_1, \ldots, f_\ell) = (f'_1, \ldots, f'_\ell)$ (small-step algorithms)

These properties witness the two facets of "small-step" algorithms

- local modification of the current environment
- and to do so, local query of the current environment

(since query is part of the work. . . )

# From $\Psi : T \to T$ to $\Phi : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$

We focus on the Bounded Cause property

Recall: | all sorts are countable
environment consists of total functions

$T = T_1 \times \cdots \times T_\ell$ $T_i$ = function space over the sorts $\equiv \mathbb{N}^{\mathbb{N}}$

**normalize** **uncurryfy**

| $\Psi : T \to T$ <br> $(f_1, \ldots, f_\ell) \mapsto (f'_1, \ldots, f'_\ell)$ | $\Theta : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ | $\Phi : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ |
|---|---|---|

**Bounded Cause for $\Psi$ $\iff$ Bounded Cause for $\Phi$**

$\exists k \quad \forall \vec{f} \quad \forall \vec{x}$ | $f'_i(\vec{x})$ depends on the values of $f_1, \ldots, f_\ell$ on $\leq k$ points $\qquad i = 1, \ldots, \ell$

IF AND ONLY IF

$\exists k \quad \forall f \quad \Phi(f)$ depends on the values of $f$ on $\leq k$ points

# From $\Psi : T \to T$ to $\Phi : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$

The bounded cause property for $\Psi : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ has some global character: it is a bounded cause for each value of the image tuple of functions.

To facilitate the study of the Bounded Cause property, it is convenient to flatten the target space $T$ to a discrete space. In fact,
the bounded cause property for functionals $\mathbb{N}^{\mathbb{N}} \to \mathbb{N}$
is much easier to handle than for functionals $\mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$

What is $\Phi$ ? In the simple case $\ell = 1$, $T = T_1 = \mathbb{N}^{\mathbb{N}}$, we have

$$\begin{array}{rcll} \Psi(g)(a) & = & \Phi(f) & \textit{where } f(0) = a \quad f(x+1) = g(x) \\ \Phi(f) & = & \Psi(g)(f(0)) & \textit{where } g(x) = f(x+1) \end{array}$$

# BOUNDED CONTINUITY

Basic clopen of $\mathbb{N}^{\mathbb{N}}$ : $[u] = \{g \in \mathbb{N}^{\mathbb{N}} \mid g \text{ extends } u\}$

Domain size of $[u] = |X|$ : $u : X \to \mathbb{N}$ with $X$ FINITE

Continuity as "Finite Cause"

$\Phi : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ is continuous if and only if $\mathbb{N}^{\mathbb{N}}$ can be covered / partitioned by basic clopens on which $\Phi$ is constant

**Definition** Bounded continuity (= Bounded Cause)

$\Phi : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ is $k$-continuous iff $\mathbb{N}^{\mathbb{N}}$ can be covered by basic clopens of domain size $\leq k$ on which $\Phi$ is constant

**Theorem** Witnessing $k$-continuity: From a covering to a partition

If $\Phi : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ is $k$-continuous then $\mathbb{N}^{\mathbb{N}}$ can be partitioned in basic clopens of domain size $\leq k^2$ on which $\Phi$ is constant

Effective version true with blow-up $2k^2 - k$

# BOUNDED CONTINUITY

Since $\Phi$ takes values in a discrete space, continuity can be expressed in terms of a covering by clopen sets on which $\Phi$ is constant. This covering can also be supposed to be a partition.

---

What this first theorem says is that the same passage from a covering to a partition is also possible for $k$-continuity modulo a quadratic blow-up of the domain-size of the involved clopens.

The $k^2$ blow-up in the theorem is optimal

# Example of a 3-continuous functional

$\Phi : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ $\qquad \Phi(f) = f(\, h(f(0), f(1))\,)$

($h : \mathbb{N}^2 \to \mathbb{N}$ fixed function)

$\Phi(f)$ known with 3 values of $f$ : at $0$, $1$, $h(f(0), f(1))$

$\Phi(f)$ is the value of the term $\mathtt{f}(\,\mathtt{h}(\mathtt{f}(\mathtt{a}), \mathtt{f}(\mathtt{b}))\,)$

when $\begin{cases} \mathtt{f} & \text{interpreted by } f \\ \text{auxiliary symbols } \mathtt{a}, \mathtt{b}, \mathtt{h} & \text{interpreted by } 0, 1, h \end{cases}$

We shall see that all $k$-continuous functionals $\mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ are analogous to the above one

# Example. A 3-continuous functional

A simple non trivial example of a 3-continuous functional

This functional is typically in the ASM spirit. In particular, its definition reduces to a term.

---

When looking for examples of $k$-continuous functionals, this kind of example is the sole that we can find. Indeed, there is a reason for that which is expressed by this rough and informal statement of the next Lemma.

# ASM Thesis vs Bounded effect & cause

**Lemma (Bounded continuity = Bndd Cause $\Longleftrightarrow$ Term Query Cause)**

Suppose $\Phi : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ had Bounded Cause.

Then there exist fixed $\begin{cases} \text{auxiliary functions } h_1, \ldots, h_n \\ \text{term } \gamma(\mathtt{f}, \mathtt{h}_1, \ldots, \mathtt{h}_n) \end{cases}$

such that $\forall f \quad \Phi(f)$ is the value of $\gamma(\mathtt{f}, \vec{\mathtt{h}})$

when $\mathtt{f}, \mathtt{h}_1, \ldots, \mathtt{h}_n$ are interpreted by $f, h_1, \ldots, h_n$

A proof of the ASM Thesis from semantical conditions

**Main Theorem 1 Bounded Effect & Cause $\Longleftrightarrow$ ASM**

Suppose $\Psi : T \to T$ had Bounded Effect & Cause.

Then $\Psi$ is the transition functional of some ASM.

In particular, Bounded Effect & Cause $\Longleftrightarrow$ ASM

The Lemma & Theorem have effective versions

# ASM Thesis vs Bounded effect & cause

**Lemma.**
If $\Phi$ has $k$-bounded cause then the term $\gamma(\mathtt{f}, \vec{\mathtt{h}})$ has $\leq k^2$ occurences of $f$.
That means that $\Phi(f)$ can be computed by querying $f$ at $\leq k^2$ points
For the effective version the blow-up is $2k^2 - k$.

**Theorem.**
From the Bounded effect & cause property, our proof finds terms (and also finds the auxiliary functions for the parameters in the term) which give the "Bounded exploration" Postulate.

If $\Psi$ has $k$-bounded cause and $p$-bounded effect then the ASM program will show (an apparent) $(kp)^2$-bounded effect and cause (though, its transition functional is $\Psi$ hence has $k$-bounded cause and $p$-bounded effect).

For the effective version the blow-up is $2(kp)^2 - kp$.

# Why is the problem non trivial?

Suppose
- (1) $\Phi : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ is computable
- (2) there exists a computably enumerable sequence $([u_i])_{i \in \mathbb{N}}$ of basic open sets with domain-size $\leq k$ on which $\Phi$ is constant.

- A priori, the considered algorithm to compute $\Phi(f)$ may query $f$ at more than $k$.
- To compute $\Phi(f)$, the simplest way to use the covering by the $[u_i]$'s is to query $f$ on $dom(u_0)$, $dom(u_1)$,... up to find $i$ such that $f$ agrees with $u_i$ hence $f \in [u_i]$.
  But this is a `WHILE` loop which may query $f$ on more than $k$

Our theorems tell that we can bound the `WHILE` loop to $O(k^2)$.

# Uniform topological spaces

Bounded continuity $\equiv$ uniform continuity

*(but not relative to a metric...)*

Topology is defined via neighborhoods of points
Neighborhood of a point $x \equiv$ set of points "close" to $x$

Uniformity is defined via entourages of the diagonal
Entourage of the diagonal $\equiv$ set of pairs of "close" points

## Paradigmatic Example: Metric spaces

Basic neighborhoods of $x$ : $\{y \mid d(x,y) < \varepsilon\}$

Basic entourages: $\{(x,y) \mid d(x,y) < \varepsilon\}$

There are axioms for the general (non metric) notions of topology on a set and of uniformity on a set

# Uniform topological spaces

It turns out that $k$-continuity is relevant to uniform continuity

*(but not relative to a metric...)*

Thus, we have to use the general theory of uniform spaces.

As is the case for topology, uniformities as best understood by starting with the metric case.

We have no time to detail the axioms for uniformities

# Bounded Information Uniformity on $\mathbb{N}^{\mathbb{N}}$

**Definition. Bounded Information Uniformity on the Baire space**

Basic entourages: equivalence relations $\bigcup_{i\in\mathbb{N}} [u_i] \times [u_i]$ where $([u_i])_{i\in\mathbb{N}}$ is a partition by basic clopen sets all of which have domain size $\leq k$, for some $k$.

- The associated topology is the usual product topology
- This uniformity is transitive ($\approx$ ultrametric)
- This uniformity does not come from any metric though it refines the usual metric uniformity on $\mathbb{N}^{\mathbb{N}}$

# Bounded Uniformity on the Baire space $\mathbb{N}^{\mathbb{N}}$

The Bounded Information uniformity is not a classical one (up to our knowledge).
The condition on the size of the basic clopen sets has motivation from computer science and not from mathematical analysis or algebra.

---

Let us stress that this uniformity is compatible with the usual topology on the Baire space but does not come from a metric.

# Bounded Cause and uniform continuity

$T$ product of spaces of total functions over sorts $\approx \mathbb{N}^{\mathbb{N}}$

**Theorem** *For any $\Phi : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$, the following cond. are equivalent*

1. $\Phi$ has Bounded Cause
2. $\Phi$ is $k$-continuous for some $k$
3. $\Phi$ is uniformly continuous $\left(\begin{array}{l}\text{bounded unif. on } \mathbb{N}^{\mathbb{N}} \\ \text{discrete one on } \mathbb{N}\end{array}\right)$

**Theorem** *For any $\Psi : T \to T$, the following cond. are equivalent*

1. $\Psi$ has Bounded Cause
2. $\Psi$ is "linearly" uniformly continuous ($\approx$ Lipschitz)

# Bounded Cause and uniform continuity

The bounded uniformity allows us to get rid of the parameter $k$

Lipschitz condition with a metric:

$$d(\Psi(\vec{f}), \Psi(\vec{g})) \leq N \; d(\vec{f}, \vec{g})$$

means that
the inverse image of the basic entourage $\{(\vec{f}, \vec{g}) \mid d(\vec{f}, \vec{g}) < \varepsilon\}$
contains the basic entourage $\{(\vec{f}, \vec{g}) \mid d(\vec{f}, \vec{g}) < \varepsilon/N\}$

Here linear uniform continuity means that
the inverse image of a basic entourage with domain size $\leq k$
contains a basic entourage which has domain size $\leq Nk$.

# Summing up $\Psi = (\Psi_1, \ldots, \Psi_\ell) : \mathbf{T} \to \mathbf{T}$

**Main theorem 2** *The following conditions are equivalent*

1. $\Psi$ is the transition functional of a small-step algorithm
2. $\Psi$ is an ASM functional
3. $\Psi$ has bounded effect and bounded cause
4. $\Psi$ has bounded effect
   and is "linearly" uniformly continuous ($\approx$ Lipschitz)
5. $\exists k \;\; \forall \vec{f} \;\; \Psi_i(\vec{f}) = f_i \oplus \psi_i(\vec{f})$ for some unif. continuous $\psi_i : T \to$ *partial functions with domain size* $\leq k$

**Effective versions are also equivalent**

$$dom(g \oplus h) = dom(g) \cup dom(h)$$
$$(g \oplus h)(\vec{x}) = IF \;\; \vec{x} \in dom(h) \;\; THEN \;\; h(\vec{x}) \;\; ELSE \;\; g(\vec{x})$$

# Summing up $\Psi = (\Psi_1, \ldots, \Psi_\ell) : \mathsf{T} \to \mathsf{T}$

Bounded effect is a set theoretical condition

The statement with the oplus equation embeds the "Bounded effect" into this equation

The oplus operator $\oplus$ (borrowed to Abrial's Z notation) can be described as follows: "the one who is right is the last to talk"

# THANK YOU
# FOR YOUR ATTENTION