Abstract geometrical computation: Turing-computing ability and unpredictable accumulations (extended abstract).

Jérôme Durand-Lose

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n^o 5668

Abstract geometrical computation: Turing-computing ability and unpredictable accumulations (extended abstract)

Jérôme Durand-Lose

Mars 2004

Research Report N^o 2004-09

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`

Abstract geometrical computation: Turing-computing ability and unpredictable accumulations (extended abstract)

Jérôme Durand-Lose

Mars 2004

Abstract

In the Cellular Automata (CA) literature, discrete lines inside (discrete) space-time diagrams are often idealized as Euclidean lines in order to analyze a dynamics or to design CA for special purposes. In this article, we present an analog model of computation corresponding to this abstraction: dimensionless signals are moving on a continuous space in continuous time generating Euclidean lines on (continuous) space-time diagrams. Like CA, this model is parallel, synchronous, uniform in space and time, and uses local updating. The main difference is that space and time are continuous and not discrete (*i.e.* $\mathbb{R}$ instead of $\mathbb{Z}$). In this article, the model is restricted to $\mathbb{Q}$ in order to remain inside Turing-computation theory. We prove that our model can carry out any Turing-computation through two-counter automata simulation. Because of the nature of time, Zeno phenomena, *i.e.* accumulations, can happen. By reducing the NTOT problem (*whether a recursive function is not total*), we prove that forecasting any accumulation is Σ^0_2-complete in the arithmetical hierarchy, *i.e.* totally undecidable.

Keywords: Abstract geometrical computation, Analog model of computation, Arithmetic hierarchy, Cellular automata, Geometry, Turing universality, Zeno phenomena.

Résumé

Dans la littérature des automates cellulaires (AC), les lignes discrètes apparaissant sur les diagrammes espace-temps sont souvent assimilées à des lignes du plan euclidien pour analyser une dynamique ou au contraire produire un AC particulier. Dans cet article, nous présentons un modèle de calcul analogique correspondant à cette abstraction : des signaux sans dimension se déplacent dans un espace continu et engendre de vraies droites sur les diagrammes espace-temps (continus). Comme pour les AC, le modèle de calcul est parallèle, synchrone, uniforme dans le temps comme l'espace et les mises à jours se font de manière locale. La principale différence est que le temps est continu et non plus discret. Dans cet article, nous nous restreignons à valeurs rationnelles afin de rester dans le cadre de la théorie de Turing. Nous prouvons qu'il y est possible d'effectuer n'importe quel calcul au sens de Turing par la simulation d'automates à deux compteurs. À cause de la nature continue du temps, des phénomènes de type Zénon, *i.e.* des accumulations, peuvent apparaître. En réduisant le problème NTOT (*une fonction récursive est-elle totale ?*), nous prouvons que la prévision d'une accumulation est Σ_2^0-complète dans la hiérarchie arithmétique, *i.e.* totalement indécidable.

Mots-clés: Abstract geometrical computation, Modèles de calcul analogiques, Hiérarchie arithmétique, Automates cellulaires, Géometrie, Turing universalité, Phénomène type Zénon.

Abstract geometrical computation: Turing-computing ability and unpredictable accumulations(extended abstract)

Jérôme Durand-Lose

March 2003

Abstract

In the Cellular Automata (CA) literature, discrete lines inside (discrete) space-time diagrams are often idealized as Euclidean lines in order to analyze a dynamics or to design CA for special purposes. In this article, we present an analog model of computation corresponding to this abstraction: dimensionless signals are moving on a continuous space in continuous time generating Euclidean lines on (continuous) space-time diagrams. Like CA, this model is parallel, synchronous, uniform in space and time, and uses local updating. The main difference is that space and time are continuous and not discrete (*i.e.* $\mathbb{R}$ instead of $\mathbb{Z}$). In this article, the model is restricted to $\mathbb{Q}$ in order to remain inside Turing-computation theory. We prove that our model can carry out any Turing-computation through two-counter automata simulation. Because of the nature of time, Zeno phenomena, *i.e.* accumulations, can happen. By reducing the NTOT problem (*whether a recursive function is not total*), we prove that forecasting any accumulation is Σ_2^0-complete in the arithmetical hierarchy, *i.e.* totally undecidable.

Key-words: Abstract geometrical computation, Analog model of computation, Arithmetic hierarchy, Cellular automata, Geometry, Turing universality, Zeno phenomena.

1 Introduction

Cellular automata (CA) form a well known and studied model of computation and simulation. Configurations are $\mathbb{Z}$-arrays of cells, the states of which belong to a finite set. Each cell can only access the states of its neighboring cells. All cells are updated iteratively and simultaneously. The main characteristics of CA are: parallelism, synchrony, uniformity and locality of updating. The trace of a computation, or the orbit starting from an initial configuration, is represented as a *space-time diagram*: a coloring of $\mathbb{Z} \times \mathbb{N}$ with states.

Discrete lines are often observed on these diagrams. They can be the keys to understanding the dynamics and correspond to so-called *particles* or *signals* as in, *e.g.*, [Ila01, pp. 87–94] or [BNR91, DL98, HSC01, JSS02]. They can also be the tool to design CA for precise purposes and then named *signals* and used for, *e.g.*, prime number generation [Fis65], firing squad synchronization [Got66, VMP70, LN90, Maz96] or reversible simulation [DL97, DL00]. These discrete lines systems have also been studied on their own [DM02, MT99]. The figures in cited papers all exhibit discrete lines which are explicitly refereed to –and are often idealized as Euclidean lines– for describing or designing. Many more articles could have been cited, the cited ones were randomly chosen in order to show the variety.

We want to consider Euclidean lines on their own –not as a passing point for analysis or conception– while remaining with the main characteristics of CA. As Euclidean lines belong to Euclidean spaces and not $\mathbb{Z} \times \mathbb{N}$, we change the supports of space and time to $\mathbb{R}$. Configurations (at a given time or the restriction of the space-time diagram to a given time) are not mappings from $\mathbb{Z}$ to a finite set of states but partial mappings from $\mathbb{R}$ to the union of a finite set of *meta-signals* and a finite set of *collision rules*, defined for finitely many positions. The time scale is $\mathbb{R}^+$ (not $\mathbb{N}$), so that there is no such thing as a "next configuration". The following configurations are defined by the uniform movement of each signal, the speed of which is defined by its associated meta-signal. When two or more signals meet, this produces a *collision* defined by a collision rule. Each collision rule is defined by a pair of sets of meta-signals:

(incoming meta-signals, outgoing meta-signals). There must be at least two incoming meta-signals and all sets of incoming meta-signals must differ (which means determinism). In the configurations following a collision, incoming signals are removed and outgoing signals corresponding to the outgoing meta-signals are added.

Due to the continuous nature of space and time and the uniformity of movements, the traces of signals form Euclidean lines on the space-time diagrams, which we freely call *signals*. Each signal corresponds to a meta-signal which indicates its slope. Since there are finitely many meta-signals, there are finitely many slopes. This limitation may seem restrictive and unrealistic, even awkward as a quantification inside an analog model of computation. Let us notice that, first, it comes from CA: once a discrete line is identified, wherever (and whenever) the same pattern appears, the same line is expected, thus with the same slope. Second, we give two pragmatic arguments: (1) laws to define the new slopes from the previous ones in collisions are not so easy to design and pretty cumbersome to manipulate; (2) there is already much computing power and scheming things (as presented in this paper and addressed in the conclusion).

Before presenting the results in this paper, we want to convince the reader that it is not just "one more analog model of computation". First, it does not come "out of the blue" because of its CA origin (where it is implicitly used). Second, let's do a brief tour of analog/super-Turing models of computation (a wider survey can be found in [DL03, Chap. 2]). To our knowledge, the closest model is the Mondrian automata of Jacopini and Sontacchi [JS90]. They also define dynamics and work on space-time diagrams which are mappings from $\mathbb{R}^n \times \mathbb{R}$ to a finite set of colors. Their diagrams should be bounded finite polyhedra; we are only addressing lines –(hyper-)faces are not considered– and our diagrams may be unbounded and accumulation points may appear (they just forbid them). Another close model is the piecewise-constant derivative system [AM95, Bou99]. Continuous space is partitioned into finitely many polygonal regions. The trajectory from a point is defined by a constant derivative on each region, thus an orbit is a sequence of (Euclidean) line segments. This model is very different from ours: it is sequential –there is only one "signal"– and the hyper-faces that delimit the regions are artifacts that do not exist in our model.

All the other models are based on totally different approaches. Some only define mapping over $\mathbb{R}$ like recursive analysis (type 2 Turing machines) [Wei00] or analog recursive function [Moo96]. Many use a discrete iterative time like the BSS model [BCSS98] or analog recurrent neural networks [SS95]. The models with continuous time mostly use differential equations, finite support (variables) with uncountably many possible values [Orp94, ŠO01, PE74, Bra95]. To our knowledge, our model is the only one that is a dynamical system with continuous time and space but finitely many local values.

In this paper, space and time are restricted to rationals. This is possible since all the operations used preserve rationality. Moreover, this allows manipulation by, *e.g.*, Turing machines, and is enough for Turing-computing capability. All intervals should be understood over $\mathbb{Q}$, not $\mathbb{R}$. Extending the definitions to real values is automatic but only the rational case is addressed here.

After formally defining our model in Sect. 2, we rapidly show that any Turing-computation can be carried out through the simulation of two-counter automata in Sect. 3. The counters are encoded in unary and the instructions are going forth and back between the two groups of signals. The nature of space is used here: any finite number of signals can be enclosed in a bounded interval of $\mathbb{Q}$.

In Sect. 4, we show how to bound temporally a computation that is already spatially bounded. This method is constructive and relies on the continuous nature of space and time: no matter how many signals there are, there is always room for more everywhere. The construction generates an accumulation point. We define the Accumulation forecasting problem (*given a signal machine and an initial configuration, will there be any accumulation?*) and prove that it belongs to the Σ_2^0 level of the arithmetical hierarchy [Odi99].

In Sect. 5, we apply this temporal bounding construction to our two-counter automata simulation and make little modifications so that if the simulated computation stops, everything is wiped out, hence forbidding any accumulation. We prove this way that Accumulation forecasting is co-recursively enumerable-hard, *i.e.* Π_1^0-hard in the arithmetical hierarchy.

In Sect. 6, we show that this problem is in fact Σ_2^0-complete by reducing NTOT (*whether a recursive function is not total*). Testing the definition for one entry can be done with a bounded portion of the space-time diagram as in Sect. 5; we add the necessary structure to start all the possible simulations.

Conclusion, remarks and perspectives are gathered in Sect. 7.

2 Definitions

Our abstract geometrical computations are defined by the following machines:

Definition 1 A *signal machine* is defined by (M, S, R) where M is a finite set, S is a mapping from M to $\mathbb{Q}$, and R is a subset of $\mathcal{P}(M)\times\mathcal{P}(M)$ that corresponds to a partial mapping of the subsets of M of cardinality at least 2 to the subsets of M (both domain and range are restricted to elements of different S-images).

The elements of M are called *meta-signals*. Each instance of a meta-signal is a *signal* which corresponds to a line segment in the space-time diagram. The mapping S assigns rational *speeds* to meta-signals, *i.e.* the slopes of the segments. The set R defines the *collision rules*, *i.e.* what happens when two or more signals meet. It also defines the intersections of the segments. The signal machines are deterministic because R must correspond to a mapping; if it were just a relation, then the machine would be non-deterministic.

Definition 2 A *configuration*, c, is a partial mapping from $\mathbb{Q}$ to the union of M and R such that the set $\{\, q \in \mathbb{Q} \mid c(q) \text{ is defined} \,\}$ is finite.

Let us define the dynamics:

A signal corresponding to a meta-signal μ at a position q, *i.e.* $c(q) = \mu$, is moving uniformly with constant speed $S(\mu)$. A signal must start in the initial configuration or be generated by a collision. It must end in a collision or in the last configuration. This corresponds to condition 1. in Def. 3.

A collision corresponds to a collision rule $\rho = (\rho^-, \rho^+)$, also noted as $\rho^-\!\to\!\rho^+$. All, and only, signals corresponding to the meta-signals in ρ^- (resp. ρ^+) must end (resp. start) in this collision. No other signal should be present. This corresponds to condition 2. in Def. 3.

Definition 3 The *space-time diagram*, or orbit, issued from an initial configuration c_0 and lasting for T, is a mapping c from $[0, T]$ to configurations (*i.e.* a partial mapping from $\mathbb{Q} \times [0, T]$ to $M \cup R$) such that, $\forall (q, t) \in \mathbb{Q} \times [0, T]$:

1. if $c_t(q){=}\mu$ then $\exists t_i, t_f{\in}[0, T]$ with $t_i{<}t{<}t_f$ or $0{=}t_i{=}t{<}t_f$ or $t_i{<}t{=}t_f{=}T$ s.t.:
   - $\forall t' \in (t_i, t_f)$, $c_{t'}(q + S(\mu)(t' - t)) = \mu$,
   - $t_i = 0$ or $c_{t_i}(q_i) \in R$ and $\mu \in (c_{t_i}(q_i))^+$ where $q_i = q + S(\mu)(t_i - t)$,
   - $t_f = T$ or $c_{t_f}(q_f) \in R$ and $\mu \in (c_{t_f}(q_f))^-$ where $q_f = q + S(\mu)(t_f - t)$;
2. if $c_t(q) = \rho^-\!\to\!\rho^+ \in R$ then $\exists \varepsilon$, $0 < \varepsilon$, $\forall t' \in [0, T]$,
   - if $t - \varepsilon < t' < t$ then $\forall \mu \in \rho^-$, $c_t(q + S(\mu)(t' - t)) = \mu$,
   - if $t < t' < t + \varepsilon$ then $\forall \mu \in \rho^+$, $c_t(q + S(\mu)(t' - t)) = \mu$,
   - $\exists \alpha$, $0 < \alpha$, $c_{[t-\varepsilon,t)}([q - \alpha, q + \alpha]){=}\rho^-$ and $c_{(t,t+\varepsilon]}([q - \alpha, q + \alpha]){=}\rho^+$.

This definition can easily be extended to the case where $T = \infty$.

The traces of signals represent line segments whose directions are defined by $(S(.), 1)$ (1 is the temporal coordinate). Collisions correspond to the extremities of these segments. Examples of space-time diagrams are provided by the various figures. Time is always increasing upwards.

3 Turing-computation capability

We prove the Turing-computation power of our model by simulating any two-counter automaton (a finite automaton couple with two counters, A and B). The possible actions on any counter are *add/subtract* 1 and *branch if non-zero*. These machines can be described with a six-operations (the three aforementioned ones for each of the two counters) assembly language with branching labels as on the left part of Fig. 2 (see [Min67] for more on two-counter automata).

The simulation is carried out with both counters unary encoded. Configurations are formed by, left to right: a left end-marking signal, a signals amounting for the value of A, one signal encoding the current instruction, b signals for B and a right end-marking signal. The only active signal (no collision ever happens without this signal being present) is the middle one; this signal both carries out the operation and encodes the line number. It goes forth and back between the signals encoding A and B. The end-markers are used when the value is zero, otherwise there would be no other signal on the side. They also provide a simple way to test for non-zero: an end marker is met if and only if the value is zero.

They are two kinds of meta-signals: five for the counters and end-markers and the ones generated for the code. The meta-signals of the first kind are: $\mathsf{endMarker}$, a and b of speed 0 used to mark the borders and to encode respectively A and B in unary, and aMv and bMv of speed $1/2$ and $-1/2$ used to increment A and B. The use of bMv is explained in the presentation of `B++` (Fig. 1). For the second kind, each line n of the program is converted to $\overrightarrow{n}$ and $\overleftarrow{n}$ of speed 1 and -1 (except for the test of B $\neq 0$ which generates $\overrightarrow{n}$, $\overleftarrow{n}_Y$, and $\overleftarrow{n}_N$). The program is encoded in the collision rules related to the second kind.

The full transformation of a program into a signal machine is not given. We only detail the collision rules generated for a `B++` instruction at line n:

$$\begin{array}{rcl@{\qquad}rcl}
\{\overrightarrow{n}, \mathsf{endMarker}\} & \rightarrow & \{\overleftarrow{n}, \mathsf{bMv}, \mathsf{endMarker}\}, & \{\mathsf{endMarker}, \overleftarrow{n}\} & \rightarrow & \{\mathsf{endMarker}, \overrightarrow{n{+}1}\}, \\
\{\overrightarrow{n}, \mathsf{b}\} & \rightarrow & \{\overleftarrow{n}, \mathsf{bMv}, \mathsf{b}\}, & \{\mathsf{a}, \overleftarrow{n}\} & \rightarrow & \{\mathsf{a}, \overrightarrow{n{+}1}\}, \\
\{\overrightarrow{n}, \mathsf{bMv}\} & \rightarrow & \{\overleftarrow{n}, \mathsf{bMv}, \mathsf{b}\}, & \{\mathsf{aMv}, \overleftarrow{n}\} & \rightarrow & \{\mathsf{a}, \overrightarrow{n{+}1}\}.
\end{array}$$

In the space-time diagram of Fig. 1, we suppose that instructions at lines $n{-}1$ and $n{+}1$ are not doing anything, except in the last case where the previous one is also `B++`, so that there is some bMv to set in position. Each instruction sets the moving bMv (or aMv), if any.

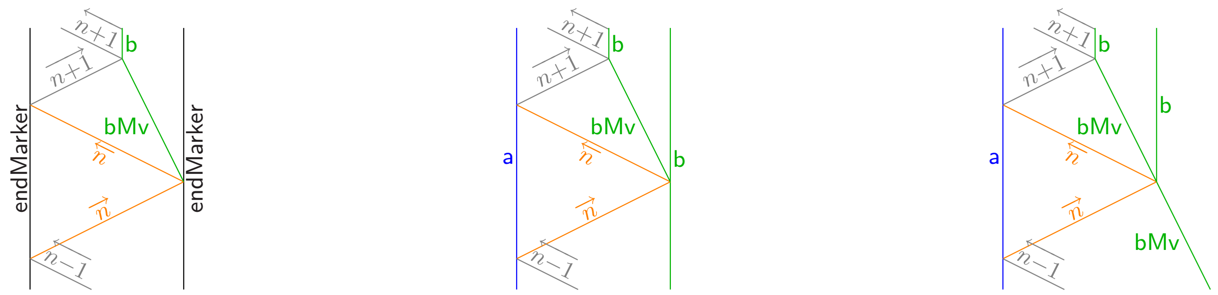

Figure 1: Implementation of `B++`.

The instruction `A++` is carried out similarly. The instructions `A--` and `B--` are just erasing exactly one corresponding signal, if any. The non-zero tests are done by simply noticing that the $\mathsf{endMarker}$ is met only if it is zero; branching is done on the collision on the left side.

Figure 2 provides three space-time diagrams associated to different initial values. The pictures are strained vertically in order to fit.

This way, any two-counter automaton can be simulated by a signal machine. Signal machines thus form a model of computation which has at least Turing-computing capability.

4 Contracting construction

It is possible to partially strain any space-time diagram as schematized on Fig. 3. The idea is to decompose the upper part according to two non-collinear vectors. One vector is used as a boundary (here the one of speed β). A change of scale is done on the second one (here multiplication by 3 on the axis corresponding to speed α). This is a strain of a given ratio about the second axe. On Fig. 3, the dotted lines indicate how the images of two points are computed. The grey parts indicate the ongoing computation.

This geometrical transformation is easily implemented inside our model: by switching to strained signal on the boundary, all following computations mimic the unstrained one. The following meta-signals are added: one for the boundary, and one strained meta-signal for each initial meta-signal[1]. All the collision rules are duplicated so that strained signals behave exactly as unstrained ones. Collisions

[1]Its speed is computed by some $(ax + b)/(cx + d)$ formula, the coefficients depend on the parameters which have to satisfy some conditions (see [DL03, Chap. 7] for all details). These conditions are always satisfied in this paper.

```
 beg: B++
      A--
      A != 0  beg1
      B != 0  imp
beg1: A--
      A != 0  beg
pair: B--
      A++
      B != 0  pair
      A != 0  beg
 imp: B--
      A++
      A++
      B != 0  imp1
      A != 0  beg
imp1: B--
      A++
      A++
      A++
      B != 0  imp1
      A != 0  beg
```

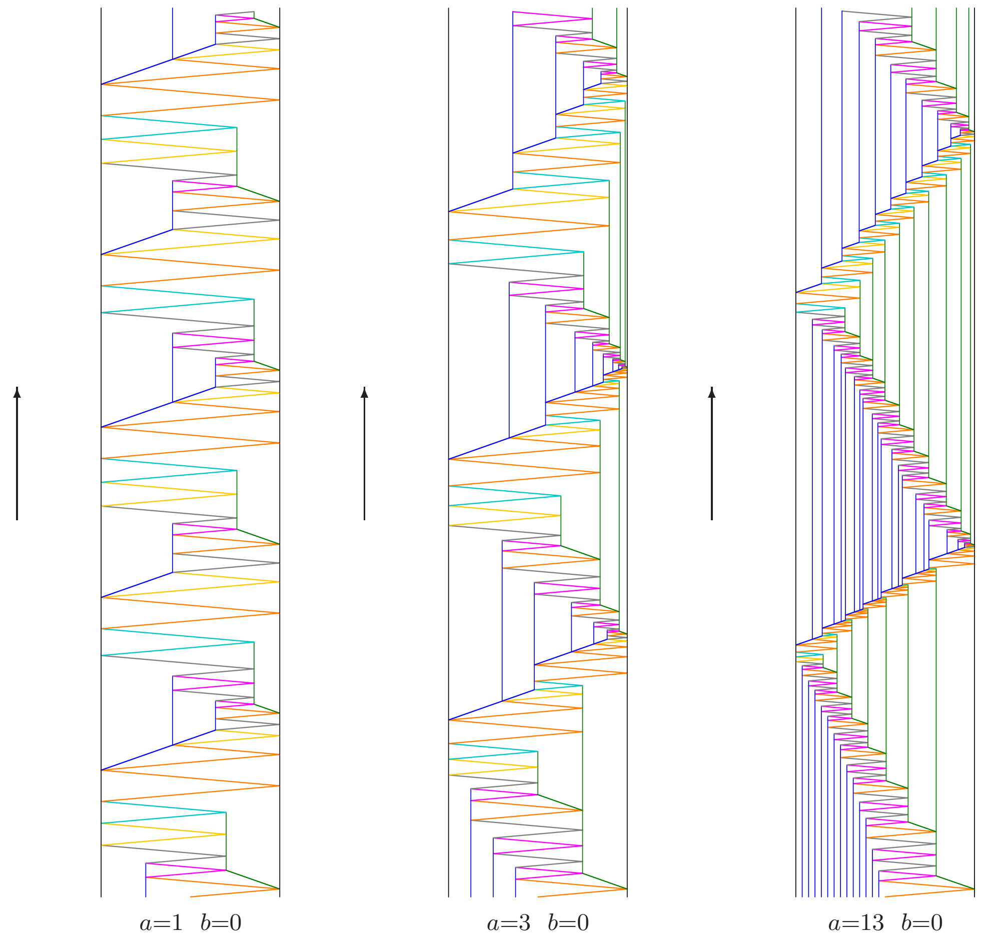

Figure 2: A two-counter automaton and its simulations for three different initial values.

of the form {*boundary and unstrained*}→{*boundary and strained*} are added. New rules are created to account for the possibility of the boundary to pass exactly on a collision.

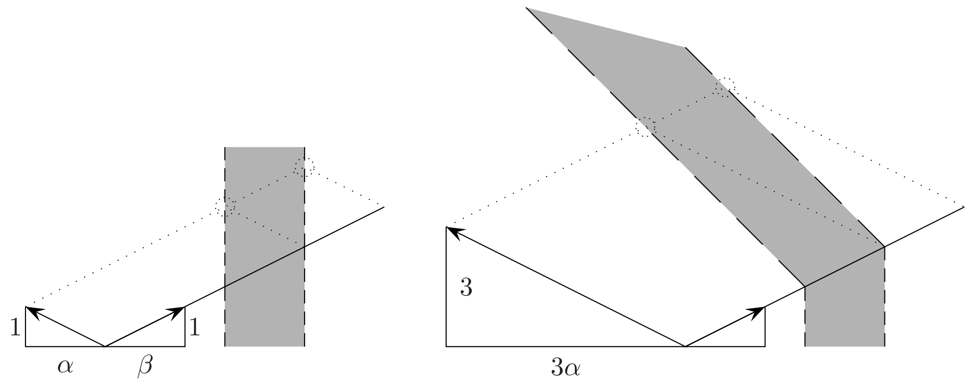

Figure 3: Strain principe.

With this construction, it is possible to build artifacts that scale by one half the rest of the computation as illustrated by Fig. 4. The two directions used correspond to v_0 and $4v_0$, where v_0 is big enough. In the left picture, nothing happens. In the middle picture, the lower signal is the boundary and a strain of ratio 1/2 is done about to the upper signal. In the right picture, a second strain takes place: the role of the directions are exchanged, and the ratio is still 1/2. After the two strains, the computation is scaled by 1/2 on both directions, thus on any direction. The whole computation is scaled by 1/2 and the original meta-signals can be used again since the computation undergoes no strain after the second one. This makes it possible to iterate the shrinking.

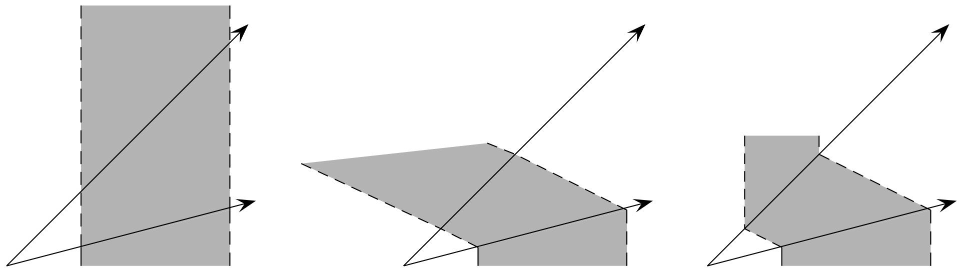
Figure 4: Shrinking principle.

From now on, only spatially bounded space-time diagrams are considered. This is sufficient to ensure that the computation remains inside the structure when shrinking is iterated. It is possible to add some extra signals to restart the shrinking each time as in the left part of Fig. 5. The middle picture represents the application of this structure to a simulation of a two-counter automaton.

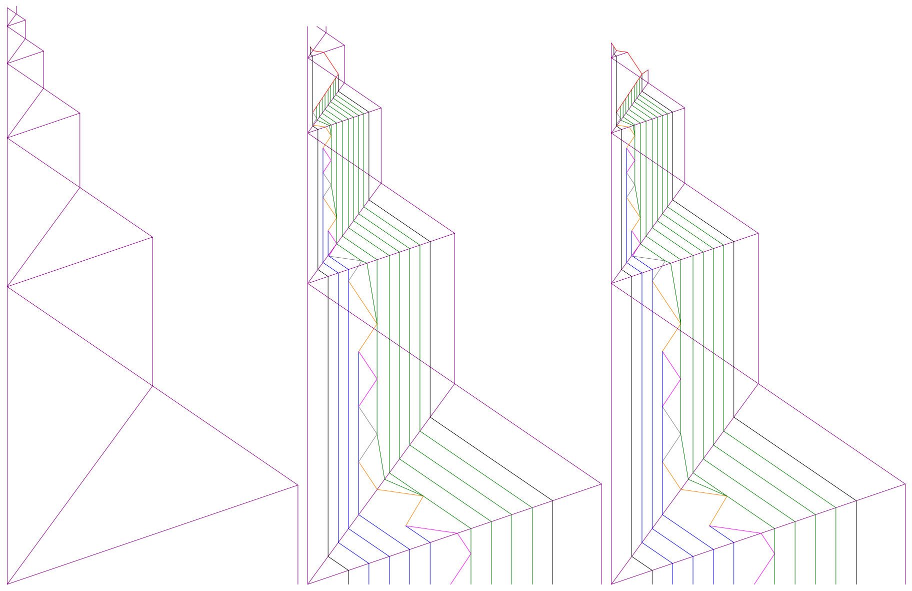
Figure 5: Iterated shrinking.

In the upper corner of the left and middle pictures of Fig. 5, there is an accumulation point: there are infinitely many collisions accumulating to the upper angle of the triangle. This is a "Zeno effect": finite (continuous) duration but infinitely many (discrete) instants.

Accumulation points are defined by infinitely many collisions accumulating to the position. Infinitely many of them should be in its light cone[2] ending there (and any light cone containing it).

The next sections are interested in the Accumulation forecasting decision problem: *given a signal machine and an initial configuration, will there be any accumulation?*

There is an accumulation if, and only if, *there exists* a position such that *for all* n, there are at least n collisions in the light cone ending in the position. Since all data are rational, it is possible to build a program always halting s.t. given the machine, the initial configuration, a position and n, it decides whether there is a least n collisions in the light cone. This means that Accumulation forecasting can be expressed as a total recursive predicate preceded by $\exists$ and $\forall$ quantifiers over $\mathbb{N}$ (possibly recursively encoding $\mathbb{Q}$). Thus, Accumulation forecasting is in Σ_2^0 in the arithmetical hierarchy (it could be easier but not harder).

[2]A light cone is formed by all positions that might have an influence on the extremity; it is bounded by the minimal and maximal speed.

5 Π^0_1-hardness of Accumulation forecasting

The class Π^0_1 corresponds to co-recursively enumerable decision problems/sets. The complement of the Halting problem is thus Π^0_1-complete. To prove that Accumulation forecasting is Π^0_1-hard, we co-reduce the Halting problem over two-counter automata (which are equivalent to Turing machines) to it. We provide a construction, from a two-counter automaton and an initial value, of a machine and an initial configuration such that there is an accumulation if, and only if, the computation of the automaton does not stop.

This is done quite easily: the previous construction provides a structure that always produces an accumulation, while our simulations of two-counter automata do not produce any accumulation on their own and are spatially bounded. The idea is to start the iterated shrinking on the simulation, with a slight modification. When the computation stops, the simulation stops and both simulation and structure are erased (as in the right picture of Fig. 5); so that there is no accumulation. If the computation does not stop, neither does the simulation, then nothing prevents the structure from producing an accumulation. Erasing is very simple to implement: a signal goes on each side, erasing any signal it meets until it reaches the border of the structure and then disappears.

6 Σ^0_2-completeness of Accumulation forecasting

Not only is Accumulation forecasting Π^0_1-hard, but it is Σ^0_2-hard (and thus complete) as we prove by reducing the following Σ^0_2-complete problem NTOT : *whether a recursive function is not total*, *i.e.* strictly partial [Odi99, p. 621]. This problem corresponds to *Whether there exists an infinite computation for a given two-counter automaton.*

The reduction is done very simply: all the possible entries are tried with previous reduction. If the computed function is not total, then the simulation for an undefined value produces the accumulation. If it is total, then none of the simulations produces an accumulation. There only remains to have a global structure that starts all the simulations without provoking any accumulation.

The construction in the previous section only needs bounded space and time; the bounds are determined by the shrinking structure. The localization of the computations is done according to the left of Fig. 6. On the right side, the scheme to generate 0 then 1... n... is presented: signals are bouncing inside a ribbon producing one extra signal each cycle. The signals leaving the ribbon are: two bounding signals (to start the simulation and the shrinking) and in-between them the unary value for the counter.

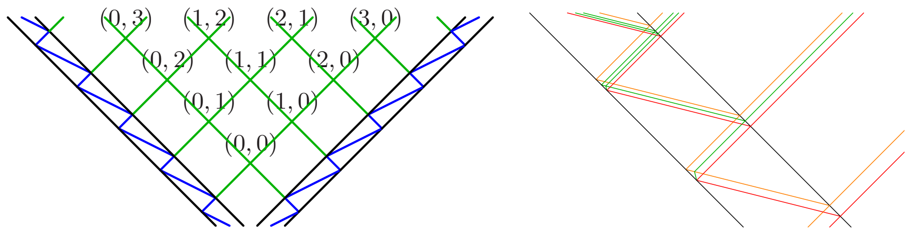

Figure 6: Localization of computations and mechanism for generating the values.

The way simulation is started and the initial counter values are set is presented on the right part of Fig. 7. On the right part is indicated how the shrinking structure is started.

Finally, all the pieces are gathered on Fig. 8. There can be seen the global structure and all the simulations. Let us note that the inner size of the squares is parametrized so that simulations do not overlap.

7 Conclusion

All the details of constructions and proofs can be found in [DL03], a long article in English is in preparation.

As long as the model is restricted to rationals, there are finitely many signals present at any instant and there is no accumulation, the model is Turing-universal and can be simulated by any Turing machine

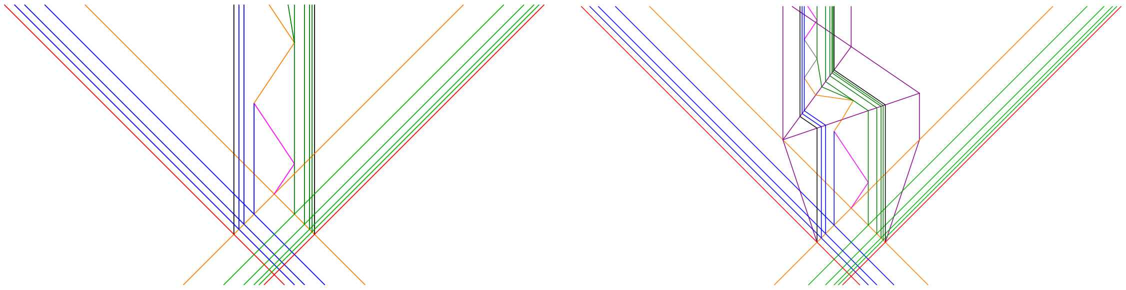

Figure 7: Starting simulation and iterated shrinking.

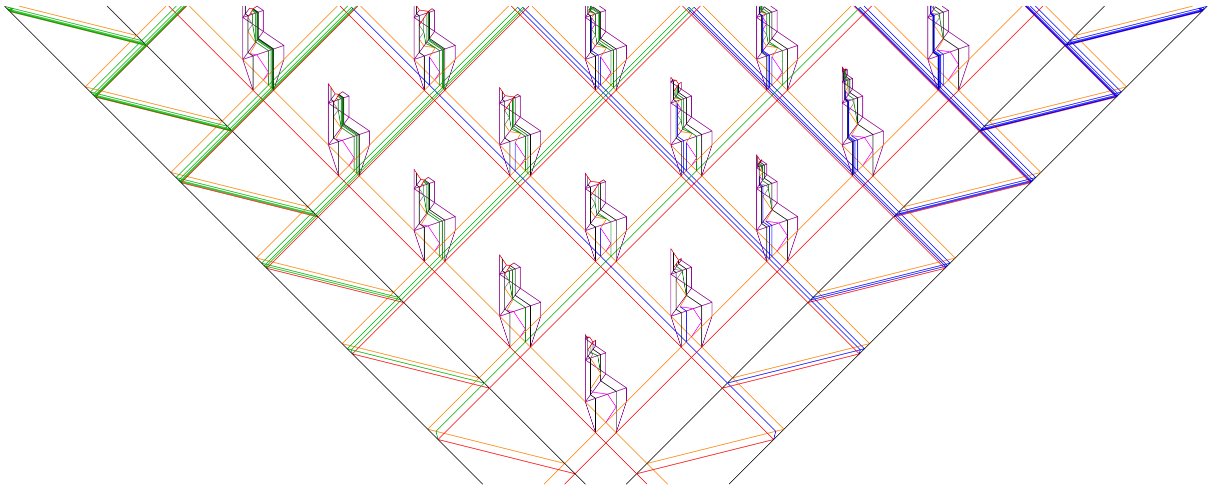

Figure 8: Simulating all the runs.

and is thus Turing-equivalent. If the "finite number of signals" condition is lifted, but signals are isolated, then this is a super-Turing model of computation following the "Infinity principle" of [EW03]. The analog power really appears when one of the remaining two constraints is lifted.

Allowing real values for speeds or positions is simple. These real values can be used as oracles and thus provide computing ability that goes beyond Turing-computation.

With a careful "treatment of accumulations", it is possible to access infinite Turing computation or computation on ordinals [Ham02]. Our model then becomes somehow similar to the black hole model [EN02]. With different levels of accumulation, we hope to climb the arithmetical hierarchy as in [AM95, Bou99].

References

[Ada02] A. Adamatzky, editor. *Collision based computing*. Springer, 2002.

[AM95] E. Asarin and O. Maler. Achilles and the Tortoise climbing up the arithmetical hierarchy. In *FSTTCS '95*, number 1026 in LNCS, pp. 471–483, 1995.

[BCSS98] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer, New York, 1998.

[BNR91] N. Boccara, J. Nasser, and M. Roger. Particle-like structures and interactions in spatio-temporal patterns generated by one-dimensional deterministic cellular automaton rules. *Phys. Rev. A*, 44(2):866–875, 1991.

[Bou99] O. Bournez. Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. *Theoret. Comp. Sci.*, 210(1):21–71, 1999.

[Bra95] M. S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoret. Comp. Sci.*, 138(1):67–100, 1995.

[DL97] J. Durand-Lose. Intrinsic universality of a 1-dimensional reversible cellular automaton. In *STACS '97*, number 1200 in LNCS, pp. 439–450. Springer, 1997.

[DL98] J. Durand-Lose. Parallel transient time of one-dimensional sand pile. *Theoret. Comp. Sci.*, 205(1–2):183–193, 1998.

[DL00] J. Durand-Lose. Reversible space-time simulation of cellular automata. *Theoret. Comp. Sci.*, 246(1–2):117–129, 2000.

[DL03] J. Durand-Lose. *Calculer géométriquement sur le plan — machines à signaux —*. Habilitation à diriger des recherches, École Doctorale STIC, Université de Nice-Sophia Antipolis, 2003. In French, http://perso.ens-lyon.fr/jerome.durand-lose/Hdr.

[DM02] M. Delorme and J. Mazoyer. Signals on cellular automata. in [Ada02], pp. 234–275, 2002.

[EN02] G. Etesi and I. Nemeti. Non-Turing computations via Malament-Hogarth space-times. *Int. J. Theor. Phys.*, 41:341–370, 2002. gr-qc/0104023.

[EW03] E. Eberbach and P. Wegner. Beyond Turing machines. *Bull. EATCS*, 81:279–304, 2003.

[Fis65] P. C. Fischer. Generation of primes by a one-dimensional real-time iterative array. *J. ACM*, 12(3):388–394, 1965.

[Got66] E. Goto. Ōtomaton ni kansuru pazuru [Puzzles on automata]. In T. Kitagawa, editor, *Jōhōkagaku eno michi [The Road to Information Science]*, pp. 67–92. Kyoristu Shuppan Publishing Co., Tokyo, 1966.

[Ham02] J. D. Hamkins. Infinite time Turing machines: Supertask computation. *Minds and Machines*, 12(4):521–539, 2002. math.LO/0212047.

[HSC01] W. Hordijk, C. R. Shalizi, and J. P. Crutchfield. An upper bound on the products of particle interactions in cellular automata. *Phys. D*, 154:240–258, 2001.

[Ila01] A. Ilachinski. *Cellular Automata –A Discrete Universe–*. World Scientific, 2001.

[JS90] G. Jacopini and G. Sontacchi. Reversible parallel computation: an evolving space-model. *Theoret. Comp. Sci.*, 73(1):1–46, 1990.

[JSS02] M. H. Jakubowsky, K. Steiglitz, and R. Squier. Computing with solitons: A review and prospectus. in [Ada02], pp. 277–297, 2002.

[LN90] K. Lindgren and M. G. Nordahl. Universal computation in simple one-dimensional cellular automata. *Complex Systems*, 4:299–318, 1990.

[Maz96] J. Mazoyer. On optimal solutions to the Firing squad synchronization problem. *Theoret. Comp. Sci.*, 168(2):367–404, 1996.

[Min67] M. Minsky. *Finite and Infinite Machines.* Prentice Hall, 1967.

[Moo96] C. Moore. Recursion theory on the reals and continuous-time computation. *Theoret. Comp. Sci.*, 162(1):23–44, 1996.

[MT99] J. Mazoyer and V. Terrier. Signals in one-dimensional cellular automata. *Theoret. Comp. Sci.*, 217(1):53–80, 1999.

[Odi99] P. Odifreddi. *Classical Recursion Theory. Volume 2.* Number 143 in Studies in Logic and the Foundations of Mathematics. Elsevier, Amsterdam, 1999.

[Orp94] P. Orponen. A survey of continuous-time computation theory. In D.-Z. Du and K.-J. Ko, editors, *Advances in Algorithms, languages and complexity*, pp. 209–224. Kluwer Academic Publisher, 1994.

[PE74] M. B. Pour-El. Abstract computability and its relation to the general purpose analog computer (some connections between logic, differential equations and analog computers). *Trans. Amer. Math. Soc.*, 199:1–28, 1974.

[ŠO01] J. Šíma and P. Orponen. Computing with continuous-time Liapunov systems. In *STOC '01*, pp. 722–731. ACM Press, 2001.

[SS95] H. T. Siegelmann and E. D. Sontag. On the computational power of neural nets. *J. Comput. System Sci.*, 50(1):132–150, 1995.

[VMP70] V. I. Varshavsky, V. B. Marakhovsky, and V. A. Peschansky. Synchronization of interacting automata. *Math. System Theory*, 4(3):212–230, 1970.

[Wei00] K. Weihrauch. *Introduction to Computable Analysis*. Texts in theoretical computer science. Springer, Berlin, 2000.