

# THE SIGNAL POINT OF VIEW: FROM CELLULAR AUTOMATA TO SIGNAL MACHINES

Jérôme DURAND-LOSE <sup>1</sup>

<sup>1</sup> Laboratoire d'Informatique Fondamentale d'Orléans, Université d'Orléans, B.P. 6759, F-45067 ORLÉANS Cedex 2.

*URL:* <http://www.univ-orleans.fr/lifo/Members/Jerome.Durand-Lose>

*E-mail address:* [Jerome.Durand-Lose@univ-orleans.fr](mailto:Jerome.Durand-Lose@univ-orleans.fr)

---

ABSTRACT. After emphasizing on the use of discrete signals in the literature on cellular automata, we show how these signals have been considered on their own. We then present their continuous counterpart: abstract geometrical computation and signal machines. Finally we relate them to other models of computation, classical and analog.

KEY-WORDS. abstract geometrical computation, analog/continuous computation, black hole model, cellular automata, computability and signal machines.

## 1. Introduction

Cellular automata (CA) were introduced in the 1950's and have been used as a model for self-replication, computation, hardware, physics, economics. . . They are composed of identical automata, called *cells* displayed on a regular lattice and communicating only with neighbors. The dynamics, defined by the transition function of a cell, is parallel and synchronous. Different points of view are commonly used: considering a single cell or an entire configuration or the whole space-time diagram.

In the past decades, a new point of view has emerged: the cells are just a substrata on which information travels. The atomic pieces of information are signals: patterns that keep repeating regularly. The dynamics can then be understood as well as conceived in terms of signals.

In this paper, we show that this signal approach is quite usual both for describing CA generated from modeling and for designing special purpose CA. Then we recall some constructions on discrete signals and present signals in a continuous setting, *abstract geometrical computation* before stating some of their computing capabilities.

Signals are embedded inside a discrete structure: both space and time are discrete. On space-time diagrams they correspond more or less to discrete lines. On the one side, the granularity of space is often exploited to get a natural scale and to get a halting condition (for example to finish a Firing squad synchronization). On the other side it imposes a quite cumbersome setting: discrete geometry; for example halfway between two cells is either on a cell or between two cells.

To generate special purposed CA, usually an Euclidean setting is used for conception and then brute force and technical skills are used to discretize. The other way round, to explain dynamics, one often forgets about the discreteness and moves on to a continuous setting for an easier explanation. Both approaches rely on scaling invariance: if the granularity is thinner or thicker, the same phenomenon happens so that it does not depend on the number of cells.

These observations led us to consider the continuous case on its own. On the one hand, there is no problem with finding the middle; all positions are available. On the other hand, there is no granularity on which to rely for an absolute scale or to ensure a correct stop.

The discrete and continuous cases have similarities; for example, both can compute (in the classical sense) and thus have numerous related undecidability problems. Nevertheless, the continuous model is quite different and addresses different topics. For example, the signal approach have been very fruitful to solve the Firing squad synchronisation problem but the constructions lead to “monsters” on the continuous side (like accumulations on a Cantor set).

Signal machines can compute anything in the classical understanding as well as CA. In the continuous setting, Zeno effect (infinitely many steps in a finite duration) can appear and be used efficiently to decide semi-decidable problems, whereas in CA it is impossible. In another direction, since it works in a continuous setting it can handle real numbers with exact precision and be related to other analog models of computation like the Blum, Shub and Smale one.

The paper is articulated as follows. Section 2 briefly recalls what cellular automata, space-time diagrams and discrete signals are. Section 3 provides examples from the literature where signals are used a mean of explanation. Section 4 presents the approach and results on discrete signals, whereas Sect. 5 presents its continuous counterpart and some results on its computing capabilities. Conclusion and perspectives are gathered in Sect. 6.

## 2. Cellular automata

This section grounds the notations. Only CA with one dimension are addressed, almost everything naturally extends to higher dimensions.

**Definition 2.1.** A *cellular automaton* (CA) is defined by  $(Q, r, f)$  where  $Q$  is a finite set of *states*, the *radius*,  $r$ , is a positive integer, and the local function,  $f$ , is a function from  $Q^{2r+1}$  to  $Q$ . A *configuration* is an element of  $Q^{\mathbb{Z}}$ . The *global function*,  $\mathcal{G} : Q^{\mathbb{Z}} \rightarrow Q^{\mathbb{Z}}$ , is defined by:

$$\mathcal{G}(c)_i = f(c_{i-r}, c_{i-r+1} \dots c_i \dots c_{i-r-1}, c_{i-r}) .$$

**Definition 2.2.** A *space-time diagram*,  $\mathcal{D}$ , is the orbit of a configuration,  $c_0$ . It is an element of  $Q^{\mathbb{Z} \times \mathbb{N}}$ ,  $\mathcal{D}(\cdot, 0)$  is  $c_0$  and  $\mathcal{D}(i, t)$  is  $\mathcal{G}^t(c)_i$ .

Unless noted otherwise, space-time diagrams are represented with time increasing upward. Each configuration is set right above the previous one.

The following definition is empirical. It may vary according to articles and is more conceptual than formal.

**Definition 2.3.** A *background* is a state pattern that may legally tile a whole space-time diagram. A *signal* is a pattern that is periodically repeating over a background. Its *speed* is defined as the spacial shift divided by the number of iterations to repeat.

For example, if 0 is a quiescent state (*i.e.*  $f(0, \dots, 0) = 0$ ) then a configuration filled with 0 is mapped onto itself and the resulting space-time diagram is filled with 0, so that 0 is a background. If the dynamics is such that a configuration with only 0's except for 1 on three consecutive cells, is regenerated every other iteration shifted by one cell on the right, then 111 is a signal and its speed is 1/2.

The speeds are bounded by the radius. This is a speed-of-light-like limitation. In space-time diagrams, the more a signal is vertical, the more it is slow. Signals have a *width*: the length of the pattern. The generated ones are generally 1-cell thin, but the ones observed are frequently not that thin.

### 3. Informal use of signals

Signals are information conveyors. The dynamics is driven by the information received, proceeded and sent.

#### 3.1. Analysing the dynamics

3.1.1. *Particles and solitons.* Signals can be thought of as moving objects. This leads to the vocabulary of “particles”. The term “soliton” is also used, but it implies that they can cross one another unaffected, like waves. Figure 1 shows some examples from the literature. This approach is important in physical modeling to ensure that studied objects could exist.

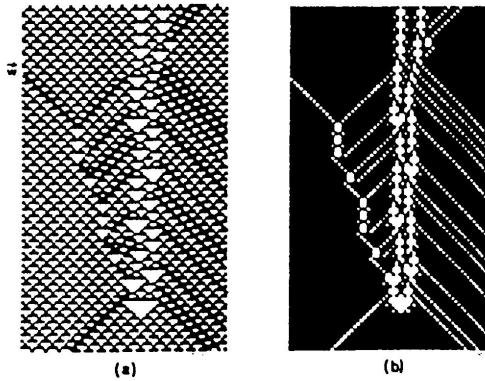


FIG. 7. Rule 54. (a) Annihilation of the radiating particle. (b) The same as (a) with the mapping defined in Fig. 6. (a) [BNR91, Fig. 7]

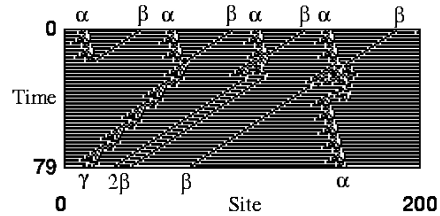


FIG. 7. The four different (out of 14 possible) interaction products for the  $\alpha + \beta$  interaction. (b) [HSC01, Fig. 7]

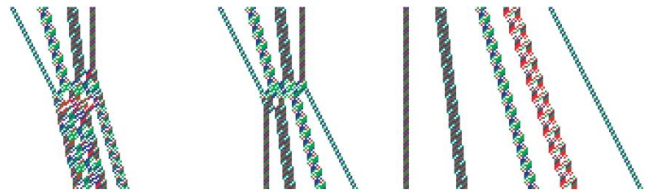


Figure 5. Two collisions of filtrons, and five free filtrons supported by the FPS model; ST diagram applies  $q = 1$ .

(c) [Siw01, Fig. 5]

(Time is increasing downward.)

Figure 1: Examples of particles and solitons.

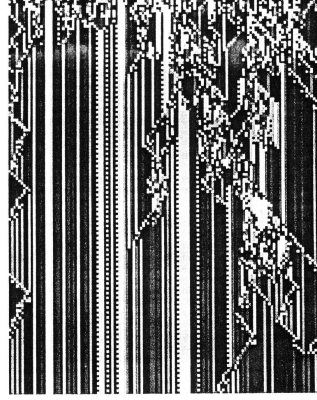



Figure 3: A simulation of the  $k = 7, r = 1$  universal CA of table 3 for an uncorrelated initial state (with a density of blanks equal to 0.76). Symbols  $g, 0, 1, A, B, \cup, \cap$  and  $T$  are represented by 

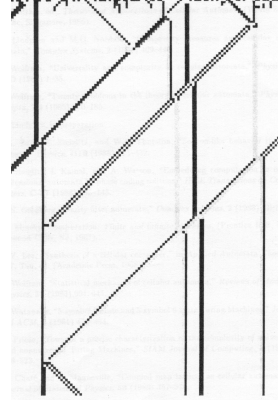



Figure 4: The  $k = 4, r = 2$  universal cellular automaton of table 4 simulated starting from a random initial state. The symbols  $0, 1, \cup$  and  $\cap$  are represented by 

(Time is increasing downward.)

Figure 2: Signals to build an universal Turing machine [LN90, Fig. 3 and 4].

**3.1.2. Computing capability.** In computer science, before computing *better* (whatever it means), one is interested in being able to compute. In [LN90] (Fig. 2), signals are found so as to provide states and tape to simulate Turing machines.

In the quest for minimal Turing-universal and intrinsically universal cellular automata [Oll02, Coo04, RO08], finding signals have often been the key to success.

## 3.2. Generating particular CA

The other way round, signals have also been used to design special purpose CA.

**3.2.1. Prime number generation.** One application is to generate the prime numbers as the iteration numbers with no signal on cell 0 (*i.e.* on the leftmost vertical line) as done on Fig. 3(a) [Fis65]. Other sets of natural numbers can be enumerated this way [MT99].

**3.2.2. Firing squad synchronization (FSS).** This is a typical synchronisation problem from distributed computing. The aim is to have all processors do something special for the first time simultaneously. They have no way to broadcast, nor a common clock to refer to. This is thought of as a line of soldiers that must shoot synchronously but are not aware of their number and have very poor means of communication: each one can only communicate with the closest soldier on each side. Two soldiers are particularised: the first is a general that will start the process and the last who knows that he is the last. In CA modeling, the cells represent the soldiers.

Most solutions work on a divide and conquer scheme. For example, Goto's algorithm (Fig. 3(b)) cuts the line of soldiers in half and restarts the process synchronously on each side. When the granularity of space is reached they shoot. A careful management of odd/even pieces ensures that granularity is reached everywhere synchronously.

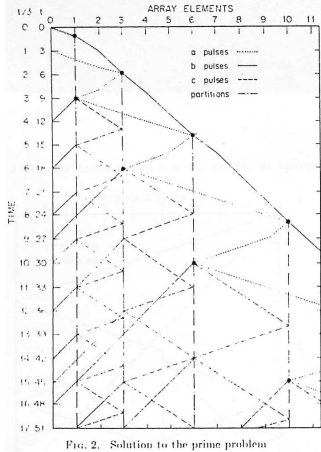


FIG. 2. Solution to the prime problem

(a) [Fis65, Fig. 2]

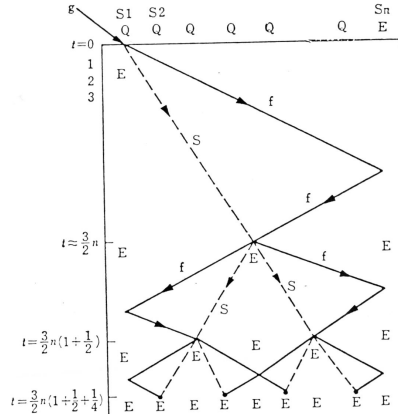


図 3-5 一斉射撃の問題 (連続近似)

(b) Goto's solution to FSS [Got66, Fig. 3+6]

(Time is increasing downward.)

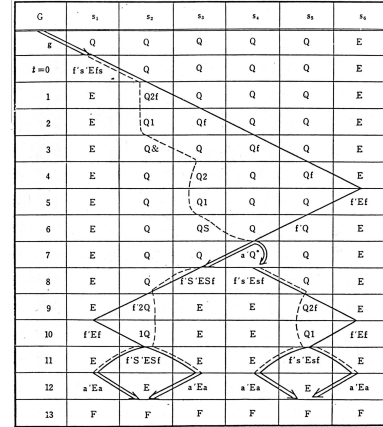


図 3-6 一斉射撃解 (n=6)

Figure 3: Geometric algorithms.

## 4. The world of discrete signals

### 4.1. Towards a definition

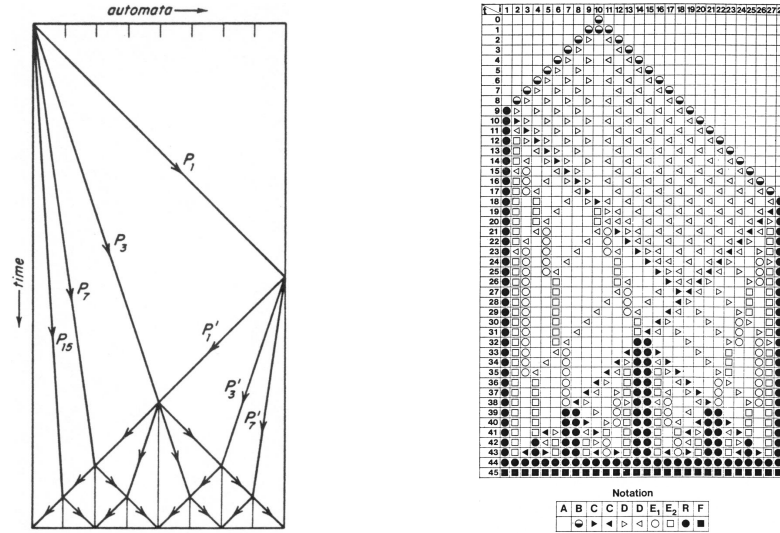
In the previous Section, we show that empirical notions of signals are used according to the context. It makes it natural to look at signals not as a tool but as a subject in itself. This is an important change of point of view. States and transitions are not considered to be the central place for the dynamics but rather some underlying layer, some byte code for a higher level language. Things are defined at the signals level, and then compiled into states and transitions. The compilation is more or less automatic depending what a signal is and what is expected from it.

For example, in the FSS construction of [VMP70] (Fig. 4), infinitely many signals and speeds are considered. One would expect it hard if not impossible to bring this forth with finitely many states. This turns out to be possible, not only because speeds are bounded but also because, basically, the movement is managed by the interactions between signals. This family can be decomposed with a few bricks: a signal for moving, one for not moving together with signals ordering to move or to stop. Each signal forward the order to move only half of the time so that the second one is half slower, the next one is half of half...

Considering this, one may think of some kind of jigsaw/tiling puzzle where thin pieces can be clipped on a board. Starting from the bottom, the board is filled upwards according to the way the pieces should be assembled together. This is the right level of abstraction. This can be implemented into CA and is abstract enough to design complex behaviours.

Let us propose a simple approach to compile signals of max speed 1 and width 1 (encoded on exactly one cell and with radius 1) to show its feasibility. Interaction only happens when signals are on the same cell. Signals are defined before their interactions.

A discrete signal is defined as a finite word over  $\{\leftarrow, -, \rightarrow\}$  that corresponds to its periodic movements. For example, a word  $\rightarrow$  would mean to move endlessly on the right, one cell at each iteration, whereas  $\rightarrow\rightarrow$  would mean right every other iteration. A signal



(Time is increasing downward.)

Figure 4: Geometrical Algorithm to solve the FSS [VMP70, Fig 1 and 3].

does not need to be anything like a discrete line segment on a period (e.g.  $\leftarrow\leftarrow\leftarrow\rightarrow\rightarrow\rightarrow$  is valid) although at a different scale it looks like a line.

Compilation is quite simple: in each cell there is a bit corresponding to each step of each signal. This means that every signal, at every step can be present in every cell! This generates a huge number of states. But signals can be handled very easily: if a signal is present with next move  $-$ , then it just goes to next step otherwise it is forgotten. If a signal is present on the left (resp. right) with next move  $\rightarrow$  (resp.  $\leftarrow$ ), it goes on the next step on the current cell. Since signals are split into steps, this is well defined. Signals are endlessly moving.

Interactions/collisions can now be defined by rules like “if these signals are present at these steps, then they are replaced by those at those steps”. Considering the vast amount of possibilities, one can imagine intended (and not extended) formulation and undefined cases could be handled by some superposition schemes and/or a default like: they cross unaffected or they disappear.

## 4.2. Achievements

This approach at the signal level together with an implementation (generally ad hoc and involving) has been quite fruitful: to give a improved solution to the FSS [Maz87], to design parabolas and circles [DMT99] and especially to develop a new kind of programing system with specific primitives.

For example, it is quite easy to have bits encoded by signals and have the dynamics carry out an addition or a multiplication [Maz96]. In these cases, the generated space-time diagrams look like the operation displayed as shown on Fig. 5.

There are ways to automatically have one computation twisted/bent so as to fit a portion of the diagram and to restart the computation in the room left. This produces recursion [DM02, Maz96, MT99] as displayed on Fig. 6.

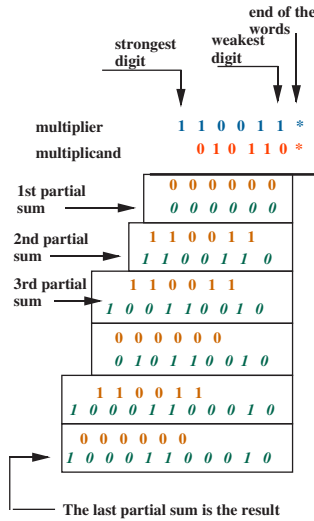


Figure 1: A human multiply

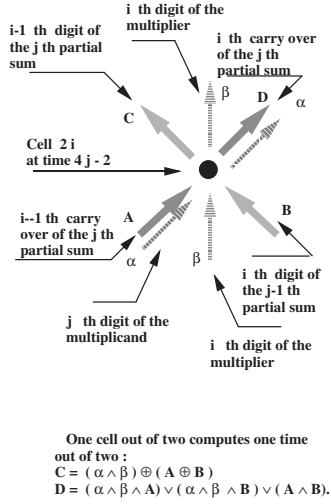


Figure 2: Computations done on one cell out of two, one unit of time out of two.

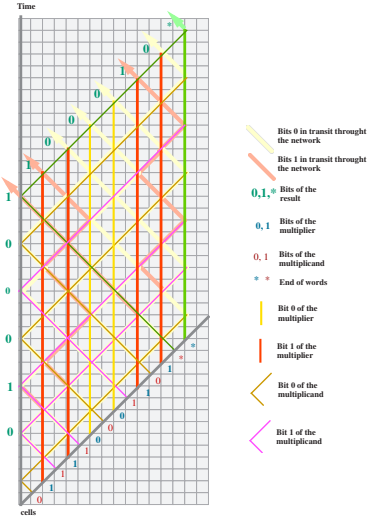


Figure 4: Multiplying 110011 by 10110.

Figure 5: Computation of a multiplication ([Maz96, Fig. 1, 3 and 4]).

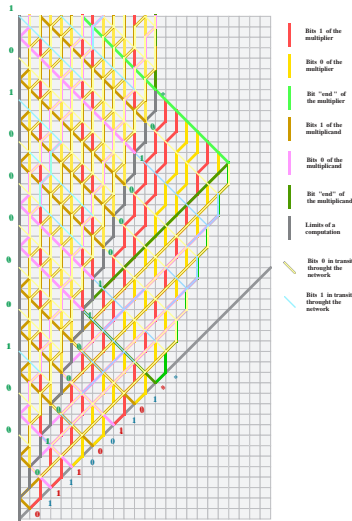


Figure 8: Computing  $(ab)^2$ .

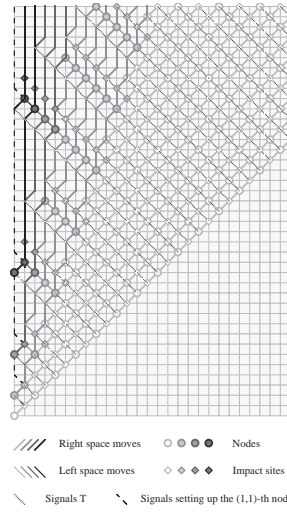


Figure 9: Setting up an infinite family of regular safe grids (the darkness of the grid indicates its rank).

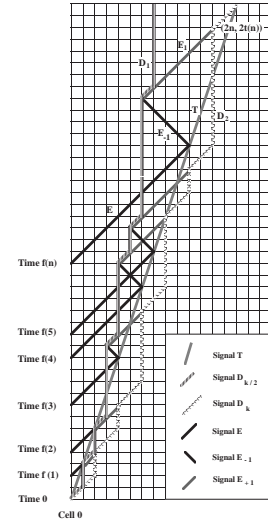


Figure 18: Characterization of the sites  $(n, f(n))$ .

Figure 6: Geometric computing ([Maz96, Fig. 8 and 19], and [MT99, Fig. 18]).

## 5. Signal machines

In an euclidean space-time  $(\mathbb{R} \times \mathbb{R}^+)$ , signals follow straight lines as illustrated on Fig. 7. They are dimension-less points (*i.e.* their width is zero). The dynamics of a single signal is not defined any more by a sequence of elementary displacements but by a constant speed. Thus its trace is a line segment in any space-time diagram.

The speed only depends on the nature of the signal since for discrete signals, it only depends on the pattern. Pragmatically, it simplifies everything; but nevertheless, the model is already very rich.

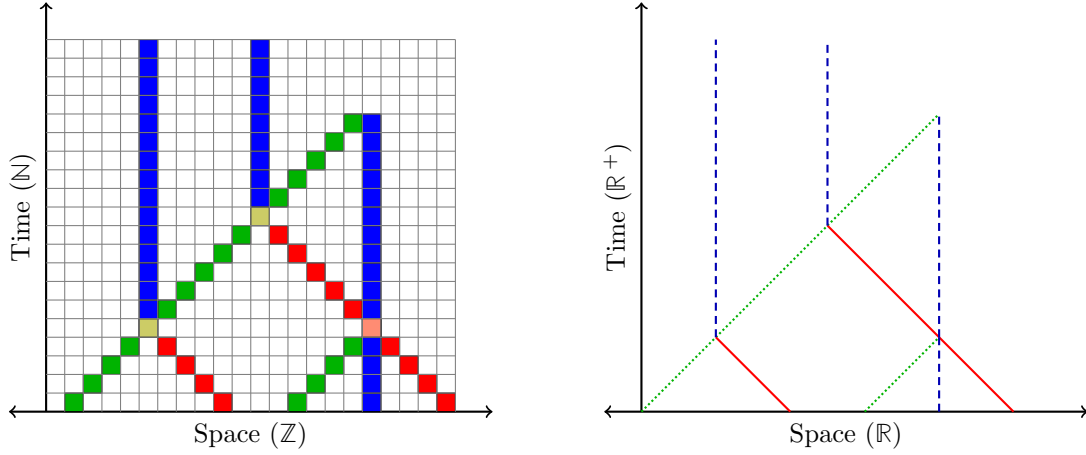


Figure 7: Space-time diagram of a cellular automaton and its signal machine counterpart.

The whole dynamics is driven by signal collisions. When two or more signals meet, they are replaced by other signals according to some rules.

### 5.1. Definition

**Definition 5.1.** A *signal machine* is defined by  $(M, S, R)$  where  $M$  is a finite set of meta-signals,  $S$  is a function  $(M \rightarrow \mathbb{R})$  that assigns speeds, and  $R$  defines the collision rules (a function from  $2^M$  to itself). A *signal* is an instance of a meta-signal.

There are finitely many signals which is not the case in some of above discrete examples. The reason is that otherwise the machine would not be finitely defined.

Let  $\mu \in M$ , if a  $\mu$ -signal is at position  $x$ , then it will be at position  $x + t.S(\mu)$  at time  $t$  if no other signal is met before.

A collision rule is denoted  $\rho^- \rightarrow \rho^+$ , if the meeting signals correspond to  $\rho^-$ , they are removed and replaced by signals corresponding to  $\rho^+$ . For example, in Fig. 7, whenever a dotted signal meets a dashed one they are replaced by a line one and a dashed one. Since  $R$  is a function, the dynamics is deterministic.

**Definition 5.2.** The *extended value set*,  $V$ , is the set of meta-signals plus two special values:  $\emptyset$  for the background (*i.e.* the absence of any signal) and  $\star$  for an accumulation. A *configuration* maps the underlying space to the extended set  $(\mathbb{R} \rightarrow V)$  such that there are finitely many non  $\emptyset$  positions.

The finitely many non  $\emptyset$  signals condition amounts for the finiteness of a configuration ensuring that the collisions are clearly defined.

**Definition 5.3.** Let  $S_{min}$  and  $S_{max}$  be the minimal and maximal speeds. The *causal past*, or (*backward*) *light-cone*, arriving at position  $x$  and time  $t$ ,  $I^-(x, t)$ , is defined by all the positions that might influence the information at  $(x, t)$  through signals, formally:

$$I^-(x, t) = \{ (x', t') \mid x - S_{max}(t-t') \leq x' \leq x - S_{min}(t-t') \} .$$

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



- (1)  $\forall t \in [0, T], \{x \in \mathbb{R} \mid ct(x) \neq \emptyset\}$  is finite,
- (2) if  $ct(x) = \mu \in M$  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.:
  - $\forall t' \in (t_i, t_f), ct'(x + S(\mu)(t' - t)) = \mu,$
  - $t_i = 0$  or  $(ct_i(x_i) = \rho^- \rightarrow \rho^+ \text{ and } \mu \in \rho^+)$  where  $x_i = x + S(\mu)(t_i - t),$
  - $t_f = T$  or  $(ct_f(x_f) = \rho^- \rightarrow \rho^+ \text{ and } \mu \in \rho^-)$  or  $ct_f(x_f) = \star$  where  $x_f = x + S(\mu)(t_f - t);$
- (3) if  $ct(x) = \rho^- \rightarrow \rho^+ \in R$  then  $\exists \varepsilon, 0 < \varepsilon, \forall t' \in [t - \varepsilon, t + \varepsilon] \cap [0, T], \forall x' \in [x - \varepsilon, x + \varepsilon],$ 
  - $(x', t') \neq (x, t) \Rightarrow c_{t'}(x') \in \rho^- \cup \rho^+ \cup \{\emptyset\},$
  - $\forall \mu \in M, c_{t'}(x') = \mu \Leftrightarrow$  or  $\begin{cases} \mu \in \rho^- \text{ and } t' < t \text{ and } x' = x + S(\mu)(t' - t), \\ \mu \in \rho^+ \text{ and } t < t' \text{ and } x' = x + S(\mu)(t' - t). \end{cases}$
- (4) if  $ct(x) = \star$  then  $\forall \varepsilon > 0,$  there is infinitely many signals in  $I^-(x, t) \cap ([x - \varepsilon, x + \varepsilon] \times [t - \varepsilon, t]).$

Rules handle the collision of isolated signals. So that other kind of “continuation” would have to be defined when infinitely many signals are spatially accumulating to ensure that the configuration at  $t + \varepsilon$  is defined. Among the monsters of Fig. 8; there is Goto’s FSS counterpart and a Cantor set generation. For CA, granularity ensures the correct achievement of the FSS, but there is no such thing here.

There is a tentative definition for the first kind of accumulation (leftmost case of Fig. 8) in [DL03, Chap. 9], simplified versions are used later on (nothing or a single signal is issued).

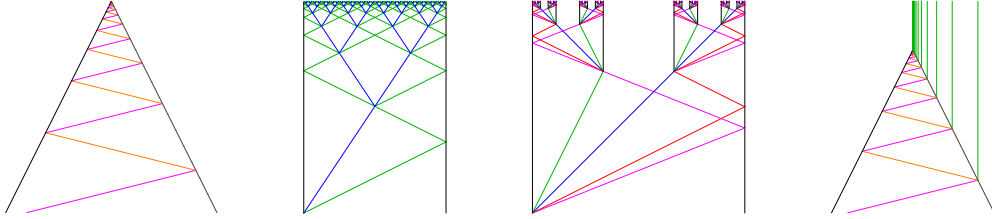


Figure 8: A simple accumulation and three unwanted phenomena.

## 5.2. Turing computing capabilities

Although Abstract geometrical computation relies on exact real values, with a simple restriction, it falls into the setting of classical computability. A signal machine is *rational* if it has only rational speeds and positions in any initial configuration. It is easy to see that, as long as there is no accumulation, all collisions happen at rational locations. Since rational numbers can be encoded and manipulated with exact precision on any computer (and the machine is finitely defined and there are finitely many signals in any configuration), implementation is possible (and has been done in Java to generate the illustrations). So that relating to Turing machine or any equivalent model makes sense.

In [DL05], it is proved that (rational) signal machine can simulate any counter automaton and thus have Turing power. This is still the case when only signal machines that are conservative and reversible are considered [DL06c]. Conservative means that each meta-signal has an positive energy and that each collision preserves this energy, so that the number of signals is bounded from the beginning. Reversible means that the collision rule is a bijection and that the signal machine can be run backward deterministically.

With computing capability comes undecidability, in the rational context many problems can be expressed in the classical setting. Some prediction problems (e.g. the apparition

of a signal, the extension of a configuration on the side) are straightforwardly undecidable [DL05] (this is not surprising since there are many such results as well as a Rice theorem for CA [Kar94]). Collision forecasting is not even semi-decidable, it is  $\Sigma_0^2$ -complete in the arithmetical hierarchy [DL06b].

Let us illustrate the computing capability as well as available geometrical operations with the proof of  $\Sigma_0^2$ -hardness. This is done by reducing the ( $\Pi_0^2$ -complete) Total problem: whether a computable function (as defined by a 2-counter automaton for example) is defined for all values. On the left of Fig. 9 there is a simulation of a two-counter automaton (the vertical lines represent the counters). It is possible to add signals and rules allowing a computation to go on while being bent one way then bent back producing a scaling by one-half. This superstructure can restart itself and iterates infinitely (scaling is done three times in the middle space-time diagram of Fig. 9). This computation is more and more compressed and accelerated by a fractal structure it is entangled in. Since each iteration corresponds to the same uncompressed duration, the embedded computation has an infinite time ahead of it. If the computation stops (as in the picture), the structure and computation is erased so that there is no accumulation, otherwise the accumulation takes place ( $\Pi_0^1$ -hardness is already reached by reducing Halt).

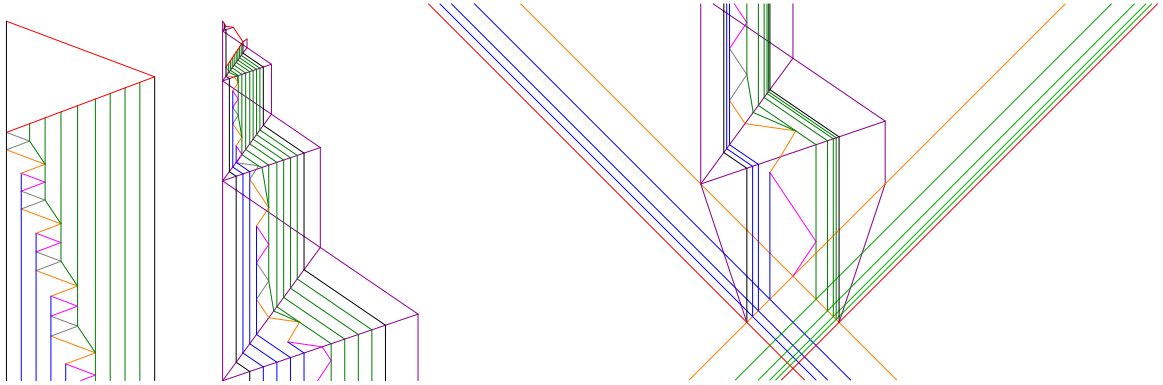


Figure 9: 2-counter automaton simulation, straight and contracted, and starting it.

On the right of Fig. 9, it is shown the way the value is provided and the computation started. On Fig. 10, a lattice is formed in order to try all the values of the counters.

With a proper use of simple accumulations (they just disappear leaving no trace), the so-called Black hole model of computation can be embedded inside rational signal machines and semi-decidable problems become decidable [DL06a]. The construction is as follows. The contracting computation is bounded by two signals. If the computation stops, a signal amounting for the answer is emitted and collected by the signals outside. When these two signals meet, if they had not received any signal they can assert that the computation never stopped.

### 5.3. Relations with the Blum-Shub-Smale model [BSS89].

Since the positions and speeds are any real number, computability issues refer to analog computation/computing on the continuous. Yet there is no analog Turing thesis and various incomparable definitions exist. Abstract geometrical computation has already been related to the BSS model: it is like a register machine where registers hold real numbers with exact

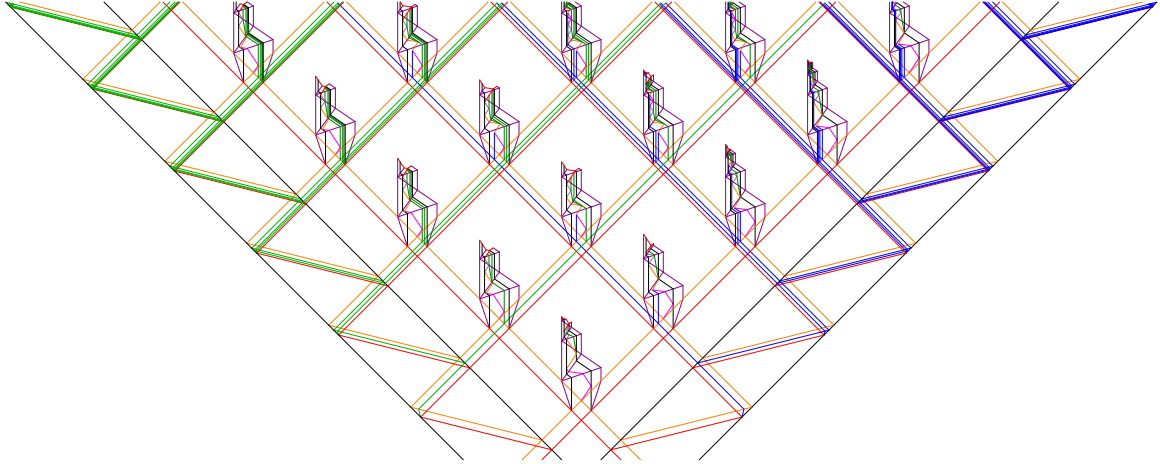


Figure 10: Trying all values.

addition, multiplication and sign test. Indirect addressing and infinitely many registers are available. Signal machines without any accumulation are equivalent to linear BSS, *i.e.* with the restriction that multiplication can only be by a constant [DL07]. The encoding of a linear BSS is done with constants in speeds and any real numbers held by a register as the distance between two parallel signals.

To achieve the full BSS, *i.e.* with internal multiplication, accumulations can be used [DL08]. Here accumulations results in a single signal where the accumulation takes place. Accumulations are used to compute infinite sums. The multiplication of two real numbers  $a$  and  $b$  is done by summing the products of  $a$  by the positive and negative powers of 2 according to the infinite binary expansion of  $b$ .

The resulting model is strictly more powerful than BSS since it can also performs square rooting. The formula/program for computing it uses only rational numbers so that the speeds of the machine are rational numbers. To compute the square root of 2, all signals are at rational positions. But the accumulation happens at the irrational position  $\sqrt{2}$ .

## 6. Conclusion

Abstract geometrical computation naturally arose from CA and is rich and promising.

The discrete constructions perfectly fit into the discrete word of CA as the continuous constructions perfectly fit the continuous word of SM. It would be interesting to investigate on how and in which cases the continuous side is a limit of the discrete one and the other way round, up to what amount can CA represent approximations of AGC. For example, on what conditions could there be an automatic discretisation of signal machine into CA in order to preserve some kind of properties?

Abstract geometrical computation still have to be studied on its own as well as related to computable analysis for its continuous aspects on one side and to transfinite computation since accumulation of order 2 and higher can be generated on the other side.

## References

- [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.
- [BSS89] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc.*, 21(1):1–46, 1989.
- [Coo04] M. Cook. Universality in elementary cellular automata. *Complex Systems*, 15:1–40, 2004.
- [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.
- [DL05] J. Durand-Lose. Abstract geometrical computation: Turing computing ability and undecidability. In B. S. Cooper, B. Löwe, and L. Torenvliet, editors, *New Computational Paradigms, 1st Conf. Computability in Europe (CiE '05)*, number 3526 in LNCS, pages 106–116. Springer, 2005.
- [DL06a] J. Durand-Lose. Abstract geometrical computation 1: Embedding black hole computations with rational numbers. *Fund. Inf.*, 74(4):491–510, 2006.
- [DL06b] J. Durand-Lose. Forecasting black holes in abstract geometrical computation is highly unpredictable. In J.-Y. Cai, B. S. Cooper, and A. Li, editors, *Theory and Applications of Models of Computations (TAMC '06)*, number 3959 in LNCS, pages 644–653. Springer, 2006.
- [DL06c] J. Durand-Lose. Reversible conservative rational abstract geometrical computation is Turing-universal. In A. Beckmann and J. V. Tucker, editors, *Logical Approaches to Computational Barriers, 2nd Conf. Computability in Europe (CiE '06)*, number 3988 in LNCS, pages 163–172. Springer, 2006.
- [DL07] J. Durand-Lose. Abstract geometrical computation and the linear Blum, Shub and Smale model. In B. S. Cooper, B. Löwe, and A. Sorbi, editors, *Computation and Logic in the Real World, 3rd Conf. Computability in Europe (CiE '07)*, number 4497 in LNCS, pages 238–247. Springer, 2007.
- [DL08] J. Durand-Lose. Abstract geometrical computation with accumulations: Beyond the Blum, Shub and Smale model. In A. Beckmann, C. Dimitracopoulos, and B. Löwe, editors, *Logic and Theory of Algorithms, 4th Conf. Computability in Europe (CiE '08) (abstracts and extended abstracts of unpublished papers)*, pages 107–116. University of Athens, 2008.
- [DM02] M. Delorme and J. Mazoyer. Signals on cellular automata. In A. Adamatzky, editor, *Collision-based computing*, pages 234–275. Springer, 2002.
- [DMT99] M. Delorme, J. Mazoyer, and L. Tougne. Discrete parabolas and circles on 2D cellular automata. *Theoret. Comp. Sci.*, 218(2):347–417, 1999.
- [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]*, pages 67–92. Kyoristu Shuppan Publishing Co., Tokyo, 1966.
- [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.
- [Kar94] J. Kari. Rice’s theorem for the limit sets of cellular automata. *Theoret. Comp. Sci.*, 127:229–254, 1994.
- [LN90] K. Lindgren and M. G. Nordahl. Universal computation in simple one-dimensional cellular automata. *Complex Systems*, 4:299–318, 1990.
- [Maz87] J. Mazoyer. A 6-states minimal-time solution to the Firing squad synchronisation problem. *Theoret. Comp. Sci.*, 50(2):183–237, 1987.
- [Maz96] J. Mazoyer. Computations on one dimensional cellular automata. *Ann. Math. Artif. Intell.*, 16:285–309, 1996.
- [MT99] J. Mazoyer and V. Terrier. Signals in one-dimensional cellular automata. *Theoret. Comp. Sci.*, 217(1):53–80, 1999.
- [Oll02] N. Ollinger. The quest for small universal cellular automata. In P. Widmayer, F. T. Ruiz, R. B. Morales, M. Hennessy, S. Eidenbenz, and R. Conejo, editors, *International Colloquium on Automata Languages and Programming (ICALP '02)*, number 2380 in LNCS, pages 318–329. Springer, 2002.

- [RO08] G. Richard and N. Ollinger. A particular universal cellular automaton. In T. Neary, D. Woods, A. K. Seda, and N. Murphy, editors, *The Complexity of Simple Programs*. National University of Ireland, Cork, 2008.
- [Siw01] P. Siwak. Soliton-like dynamics of filtrons of cycle automata. *Inverse Problems*, 17:897–918, 2001.
- [VMP70] V. I. Varshavsky, V. B. Marakhovsky, and V. A. Peschansky. Synchronization of interacting automata. *Math. System Theory*, 4(3):212–230, 1970.