Massively Parallel Automata in Euclidean Space-Time

Denys Duchier
LIFO, Université d'Orléans

Jérôme Durand-Lose
LIFO, Université d'Orléans

Maxime Senot
LIFO, Université d'Orléans

***Abstract*—In the cellular automata (CA) literature, discrete lines in discrete space-time diagrams are often idealized as Euclidean lines in order to design CA or analyze their dynamic behavior. In this paper, we present a parallel model of computation corresponding to this idealization: dimensionless particles move uniformely at fixed velocities along the real line and are transformed when they collide. Like CA, this model is parallel, uniform in space-time and uses local updating. The main difference is the use of the continuity of space and time, which we proceed to illustrate with a construction to solve Q-SAT, the satisfiability problem for quantified boolean formulae, in bounded space and time, and quadratic *collision depth*.**

***Index Terms*—Abstract geometrical computation; Signal machine; Continuous space-time; Cellular automata; Massive parallelism; Model of computation.**

I. Introduction

Since the introduction of cellular automata (CA) by J. Von Neumann in the forties, CA have been used as a model of computation [Kari, 2005], self-reproduction [Von Neumann, 1966], dynamical systems [Hedlund, 1969] and biological and physical phenomenae. CA are composed of identical and elementary *cells*. Each cell can exchange information only with its close neighbors and update its state according to a local rule of transition. Space and time are discrete, and information travels with a bounded speed. The evolution of the system is parallel and synchronous, and the local rule is applied uniformly to each cell. Different points of view are usually considered: we can focus on only one cell, a complete configuration of cells or the whole space-time diagram. An alternate point of view exists and explains diagrams by signals, particles and collisions. In this approach, the cells are just a substrata on which information travels. Signals become the basic objects and allow to implement computations in CA [Mazoyer and Terrier, 1999, Mazoyer, 1996].

In this paper, we consider *signal machines*, an abstract and geometrical model of computation, first introduced in [Durand-Lose, 2003], which extends CA into continuous space and time following the intuition outlined above. In this model, dimensionless *particles* move uniformly on the real axis. When a set of particles collide, they are replaced by a new set of particles according to a chosen collection of *collision rules*. We consider the temporal evolution of these systems through their *space-time diagram*, in which traces of the particles are materialized by line segments that we call *signals*. The space-time diagram of a signal machine constitutes a *geometrical computation*. Like CA, in which signal machines have their origins (see Fig. 1), the model of signal machines is massively parallel, uniform and admits a strong geometrical interpretation. The spatial aspect of this model is essential since computations are carried out by bits of information travelling and interacting when they meet.

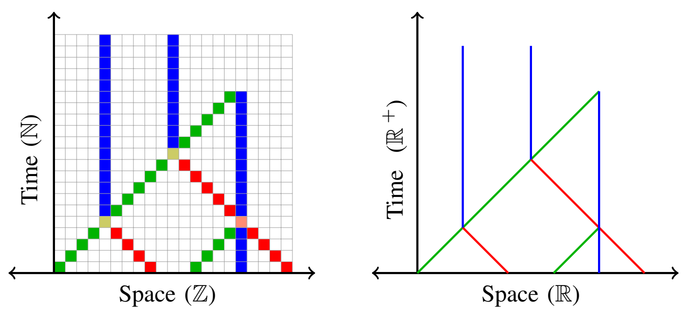


Fig. 1. From cellular automata to signal machines.

It is possible to do Turing-computation with signal machines [Durand-Lose, 2005] and even to do analog computation by a systematic use of the continuity of space and time [Durand-Lose, 2008, 2009a,b]. Of course, there are other *geometrical* models of computation: colored universe [Jacopini and Sontacchi, 1990], geometric machines [Huckenbeck, 1989, 1991], piece-wise constant derivative systems [Asarin and Maler, 1995, Bournez, 1997], optical machines [Naughton and Woods, 2001] ...

Most of the work to date in this domain, called *abstract geometrical computation* (AGC), has dealt with the simulation of sequential computations even though the model, as a continuous extension of cellular automata, is inherently parallel. In the present paper, we propose signal machines as a theoretical foundation for studying (certain classes of) massively parallel, spatially distributed algorithms, and the implications that potentially unbounded parallelism may have on the nature and structure of complexity classes. Our approach is based on distributing work across space following a fractal pattern. We illustrate this proposal with an example of how the continuity of space and time can be used to solve Q-SAT, the classical PSPACE problem of satisfiability of quantified boolean formulae, in bounded space and time. Since the amount of space and time used for a computation is no longer particularly meaningful in our model, we offer new measures of complexity based on the maximal length of chains and anti-chains in the space-time diagram of a computation

Meta-Signals	Speed
w	0
$\overrightarrow{\mathsf{div}}$, $\overrightarrow{\mathsf{lo}}$	3
$\overrightarrow{\mathsf{hi}}$	1
$\overleftarrow{\mathsf{back}}$	-3

Collision rules
{ w, $\overrightarrow{\mathsf{div}}$ } → { w, $\overrightarrow{\mathsf{hi}}$, $\overrightarrow{\mathsf{lo}}$ }
{ $\overrightarrow{\mathsf{lo}}$, w } → { $\overleftarrow{\mathsf{back}}$, w }
{ $\overrightarrow{\mathsf{hi}}$, $\overleftarrow{\mathsf{back}}$ } → { w }

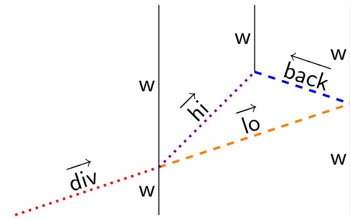


Fig. 2. Geometrically computing the middle.

regarded as directed graph.

The model of signal machines is introduced in Sect. II. Section III presents a solution of Q-SAT on signal machines. Discussions and remarks about the model and complexities are gathered in Sect. IV.

II. Definitions

In this section, we introduce the model of signal machines and illustrate it with an example for geometrically computing the middle.

A. Signals

Each *signal* is an instance of a *meta-signal*. The associated meta-signal defines its *velocity* and what happens when signals meet. Figure 2 presents a very simple space-time diagram. Time is increasing upwards and the meta-signals are indicated as labels on the signals. Existing meta-signals are listed on the left of Fig. 2.

Generally, we use over-line arrows to indicate the direction of propagation of a meta-signal. For example, $\overleftarrow{\mathsf{a}}$ and $\overrightarrow{\mathsf{a}}$ denotes two different meta-signals; but as can be expected, they have similar uses and behaviors. No over-line arrow indicates a stationary signal *e.g.* w in Fig. 2.

B. Collision rules

When a set of signals collide, they are replaced by a new set of signals according to a matching collision rule. A rule has the form:

$$\{\sigma_1, \ldots, \sigma_n\} \rightarrow \{\sigma'_1, \ldots, \sigma'_p\}$$

where all σ_i and σ'_j are meta-signals. A rule matches a set of colliding signals if its left-hand side is equal to the set of their meta-signals. By default, if there is no exactly matching rule for a collision, the behavior is defined to regenerate exactly the same meta-signals. In such a case, the collision is called *blank*. Collision rules can be deduced from space-time diagram as on Fig. 2. They are also explicitly listed on the left of this diagram.

C. Signal machine

A signal machine is defined by a set of meta-signals, a set of collision rules, and an initial configuration, i.e. a finite set of particles placed on the real line. The evolution of a signal machine can be represented geometrically as a *space-time diagram*: space is always represented horizontally, and time vertically, growing upwards.

The example of Fig. 2 computes the middle: the new w is located exactly half way between the initial two w. This process does not depend on the initial location of the two walls, neither on the distance between them. Since the space is continuous, this geometrical algorithm always works and spatially marks the middle by a stationary signal.

III. Q-SAT in bounded space-time

As an illustration of massively parallel computations with signal machines, we outline a construction for solving Q-SAT, the satisfiability problem for quantified boolean formulae (QBF), in bounded space-time.[1] Combinatorial computations self-distribute across space following a fractal pattern and information travels at fixed velocities.

A QBF is a closed formula of the form:

$$\phi \;=\; Qx_1Qx_2\ldots Qx_n \quad \psi(x_1, x_2, \ldots, x_n)$$

where $Q \in \{\exists, \forall\}$ and ψ is a quantifier-free formula of propositional logic. A recursive algorithm for solving Q-SAT is:

$$\begin{aligned} \mathsf{qsat}(\exists x\ \phi) &= \mathsf{qsat}(\phi[x \leftarrow \mathsf{false}]) \vee \mathsf{qsat}(\phi[x \leftarrow \mathsf{true}]) \\ \mathsf{qsat}(\forall x\ \phi) &= \mathsf{qsat}(\phi[x \leftarrow \mathsf{false}]) \wedge \mathsf{qsat}(\phi[x \leftarrow \mathsf{true}]) \\ \mathsf{qsat}(\beta) &= \mathsf{eval}(\beta) \end{aligned}$$

where β is a ground boolean formula. This is exactly the structure of our construction: each quantified variable splits the computation in 2, $\mathsf{qsat}(\phi[x \leftarrow \mathsf{false}])$ is sent to the left and $\mathsf{qsat}(\phi[x \leftarrow \mathsf{true}])$ to the right, and subsequently the recursively computed results that come back are combined (with $\vee$ for $\exists$ and $\wedge$ for $\forall$) to yield the result for the quantified formula. This process can be viewed as an instance of *Map/Reduce*, where the *Map* phase distributes the combinatorial exploration of all possible valuations across space using a binary decision tree, and the *Reduce* phase collects the results and aggregates them using quantifier-appropriate boolean operations. As our running example, we will use $\phi \;=\; \exists x_1 \forall x_2 \forall x_3 \quad x_1 \wedge (\neg x_2 \vee x_3)$.

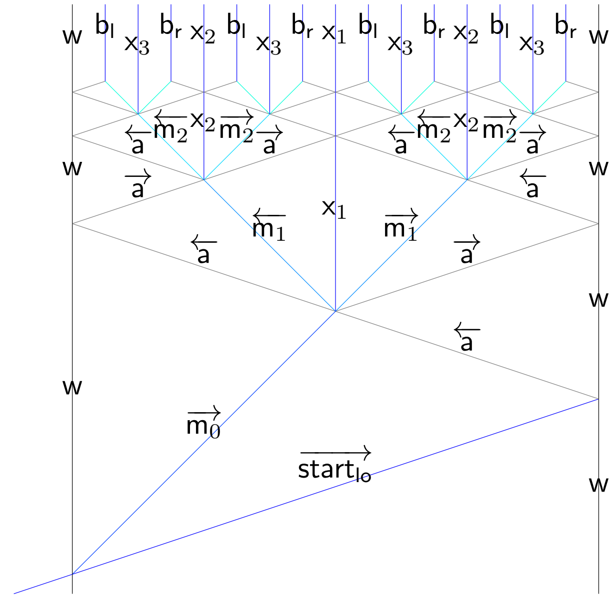
A. Combinatorial comb

The first step of our construction is to put into place the decision points for constructing the binary decision tree. The intuition is that the decision for variable x_i will be represented by a stationary signal: the space to the left should

[1] We do not go into the details both for lack of space and because the formal details of this construction have been submitted elsewhere.

$$\{\ \overrightarrow{\mathsf{start}},\ \mathsf{w}\ \} \rightarrow \{\ \mathsf{w},\ \overrightarrow{\mathsf{start_{lo}}},\ \overrightarrow{\mathsf{m}_0}\ \}$$
$$\{\ \overrightarrow{\mathsf{start_{lo}}},\ \mathsf{w}\ \} \rightarrow \{\ \overleftarrow{\mathsf{a}},\ \mathsf{w}\ \}$$
$$\{\ \mathsf{w},\ \overleftarrow{\mathsf{a}}\ \} \rightarrow \{\ \mathsf{w},\ \overrightarrow{\mathsf{a}}\ \}$$
$$\{\ \overrightarrow{\mathsf{a}},\ \mathsf{w}\ \} \rightarrow \{\ \overleftarrow{\mathsf{a}},\ \mathsf{w}\ \}$$
$$\{\ \overrightarrow{\mathsf{m}_i},\ \overleftarrow{\mathsf{a}}\ \} \rightarrow \{\ \overleftarrow{\mathsf{a}},\ \overleftarrow{\mathsf{m}_{i+1}},\ \mathsf{x}_i,\ \overrightarrow{\mathsf{m}_{i+1}},\ \overrightarrow{\mathsf{a}}\ \}$$
$$\{\ \overrightarrow{\mathsf{a}},\ \overleftarrow{\mathsf{m}_i}\ \} \rightarrow \{\ \overleftarrow{\mathsf{a}},\ \overleftarrow{\mathsf{m}_{i+1}},\ \mathsf{x}_i,\ \overrightarrow{\mathsf{m}_{i+1}},\ \overrightarrow{\mathsf{a}}\ \}$$
$$\{\ \overrightarrow{\mathsf{m}_n},\ \overleftarrow{\mathsf{a}}\ \} \rightarrow \{\ \mathsf{b_r}\ \}$$
$$\{\ \overrightarrow{\mathsf{a}},\ \overleftarrow{\mathsf{m}_n}\ \} \rightarrow \{\ \mathsf{b_l}\ \}$$

(a) Collision rules



(b) Division process

Fig. 3. Combinatorial comb.

be interpreted as $x_i = \mathsf{false}$ and the space to the right as $x_i = \mathsf{true}$. The resulting set of stationary signals form what we call the *combinatorial comb* and its construction for our example is shown in Fig 3. Everywhere, in our entire construction (here and later), signal velocities have absolute values 0, 1, or 3.

B. Compilation into a beam

The intuition for this next phase is that a formula of propositional logic can be viewed as a tree (Fig 4(a)) whose nodes are labeled by symbols (connectives and variables). One signal is generated for each node in this tree. In order to facilitate the naming of these signals, we decorate each node with the unique *path* to it from the root. The signals for all subformulae are generated and sent along parallel trajectories to form a beam (Fig 4(b)). The beam is then propagated through the binary decision tree to explore in parallel all possible valuations (Fig 4(c)).

C. Propagation

The beam is propagated down the decision tree. For each decision point (a stationary signal for a quantified variable x_i), the beam is duplicated: one part goes through, the other is reflected. Except for the sign of their velocity, most signals remain identical in both branches; most, except those corresponding to occurences of x_i: those become false in the left branch, and true in the right branch (Fig 5(a)).

D. Evaluation

Eventually, the beam reaches the bottom of the decision tree where stationary signals $\mathsf{b_l}$ or $\mathsf{b_r}$ initiate the evaluation process of the, by now, ground boolean formula (Fig 5(c)). When $\overrightarrow{\mathsf{t}^{\text{ll}}}$ reaches $\mathsf{b_r}$, it gets reflected as $\overleftarrow{\mathsf{T}^{\text{ll}}}$. The change from lowercase to uppercase indicates that the subformula's signal is now able to interact with the signal of its parent connective. The stacking order ensures that reflected signals of subformulae will interact with the incoming signal of their parent connective before the latter reaches $\mathsf{b_r}$.

A connective is evaluated by colliding with the (uppercased) boolean signals of its arguments. For example, when the disjunction collides with its first argument, depending on the value of the latter, it becomes either the one-argument identity function or the constant `true`. This is the way the rules of Fig. 5(b) should be understood.

Note how path decorations are essential to ensure that the right subformulae interact with the right occurrences of connectives. Conjunctions and negations are handled similarly. Finally, $\overrightarrow{\mathsf{store}}$ projects the truth value of the formula root on $\mathsf{b_r}$ where it is temporarily stored until $\overrightarrow{\mathsf{collect}}$ starts the aggregation of the results.

E. Collecting the results

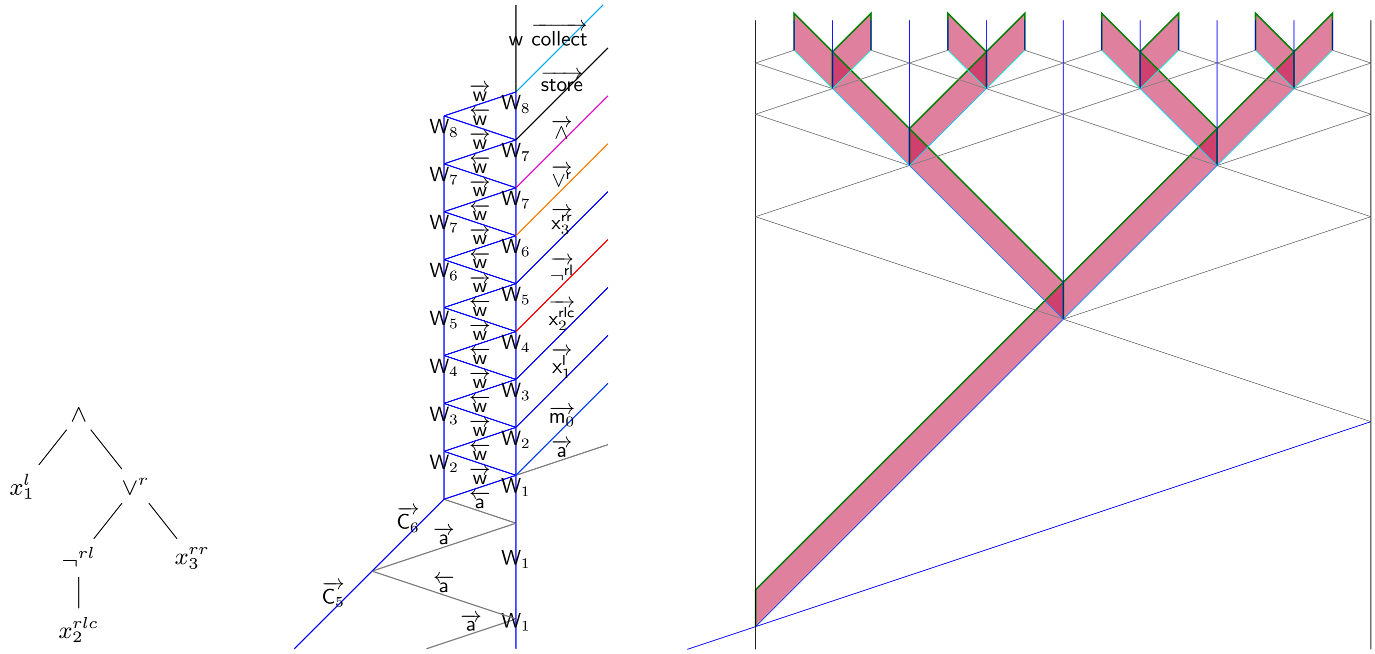
Collection of the stored results is initiated by $\overrightarrow{\mathsf{collect}}$, which is top-most in the beam and, therefore, last. This phase folds the tree back together, combining the two incoming results with $\vee$ for $\exists$ and with $\wedge$ for $\forall$ (Fig 6).

F. Size of the signal machine

The signal machine depends on the formula Compiling a QBF into a signal machine can be done in quadratic time by a Turing machine, and results in a signal machine with a quadratic number of rules and an intial state with 3 signals.

IV. Conclusion

In this paper, we briefly illustrated how the propagation of information – modeled by signals – and its interactions



(a) Labeled tree (b) Generation of the beam (c) Propagation of the beam

Fig. 4. Compilation of the formula into a beam of signals

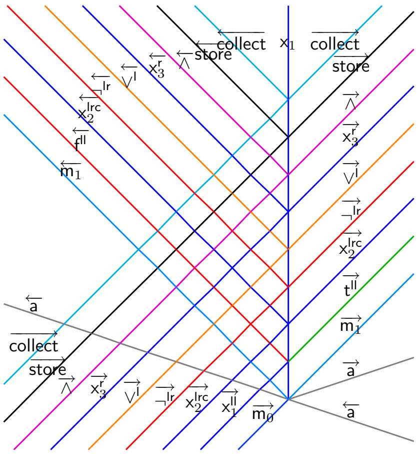
– modeled by collisions – can be used methodically to solve hard problems. We (geometrically) described a signal machine that solves Q-SAT in bounded time and space. Similar constructions can solve SAT (obviously, since a SAT formula is a special instance of Q-SAT), MAX(SAT), #SAT.

The fact that signal machines can solve PSPACE and NP problems in bounded time and space should not be understood as a collapse of the classical complexity hierarchy, but illustrates the fact that complexity classes crucially depend on the choice of a model of computation: classical classes such as P, NP and PSPACE are defined in terms of Turing machines. It also suggests that Turing machines are not well suited to the study of massively parallel, spatially distributed computations.

While width and height are respectively the traditional measures for space-complexity and time-complexity in the discrete world of cellular automata, they clearly loose here all pertinence. Indeed, considering the width and the height of our construction as complexity measures does not take in account the continuous nature of dimensions, resulting in a size of execution independent of the size of the input. Furthermore, the constant space and time used by a computation can be made as small as desired because of the continuity of space-time (it is enough to multiply all velocities by an appropriate constant or to start with a smaller initial configuration).

Instead we should regard our construction as a computational device transforming inputs into outputs. The inputs are given by the initial state of the signal machine at the bottom of the diagram. The output is the computed result that comes out at the top. The transformation is performed in parallel by many threads: a thread here is an ascending path through the diagram from an input to the output. The operations that are "performed" by the thread are all the collisions found along the path. If we view a space-time diagram of signal machine as a directed acyclic graph (directed by the relation of causality between collisions), then we can propose complexity notions adapted to this construction. We define the time-complexity as the maximal length of a chain and the space-complexity as the maximal length of an antichain. It corresponds to the maximal number of collisions along a path for the measure of time-complexity, and to the maximal number of signals simultaneously existing in the machine for the measure of space-complexity. According to these new definitions, the time-complexity of the construction of Sect. III is quadratic in the size of the formula and the space-complexity is exponential. It should also be pointed out that meta-signals and rules (but not speeds) of the signal machine depends on the formula but the compilation of a formula into a rational signal machine can be done in quadratic time by a fixed Turing machine.

Signal machines (and abstract geometrical computations) remains a model of computation that is resolutely theoretical. It illustrates the use of the continuous nature of space-time to implement efficient solutions to hard problems. This is achieved while keeping reasonable assumptions such as that information has finite density everywhere and travels at bounded (indeed, fixed) velocities. As in the discrete world of CA, space is methodically used to propagate information and signal machines are inherently parallel, providing a new abstract model for parallel programming.



(a) Split

$$\{\ \overrightarrow{\vee^{l}},\ \overleftarrow{\mathsf{T}^{ll}}\ \} \rightarrow \{\ \overrightarrow{t()^{l}}\ \}$$

$$\{\ \overrightarrow{t()^{l}},\ \overleftarrow{\mathsf{T}^{lr}}\ \} \rightarrow \{\ \overrightarrow{\mathsf{t}^{l}}\ \}$$

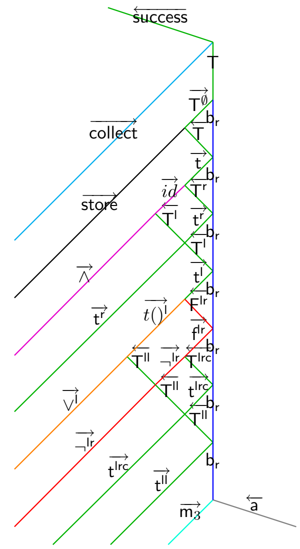
$$\{\ \overrightarrow{id^{l}},\ \overleftarrow{\mathsf{T}^{lr}}\ \} \rightarrow \{\ \overrightarrow{\mathsf{t}^{l}}\ \}$$

$$\{\ \overrightarrow{\vee^{l}},\ \overleftarrow{\mathsf{F}^{ll}}\ \} \rightarrow \{\ \overrightarrow{id^{l}}\ \}$$

$$\{\ \overrightarrow{t()^{l}},\ \overleftarrow{\mathsf{F}^{lr}}\ \} \rightarrow \{\ \overrightarrow{\mathsf{t}^{l}}\ \}$$

$$\{\ \overrightarrow{id^{l}},\ \overleftarrow{\mathsf{F}^{lr}}\ \} \rightarrow \{\ \overrightarrow{\mathsf{f}^{l}}\ \}$$

(b) Collision rules to evaluate the disjunction $\vee^{l}$



(c) Evaluation at the bottom of the comb

Fig. 5. Propagation and evaluation

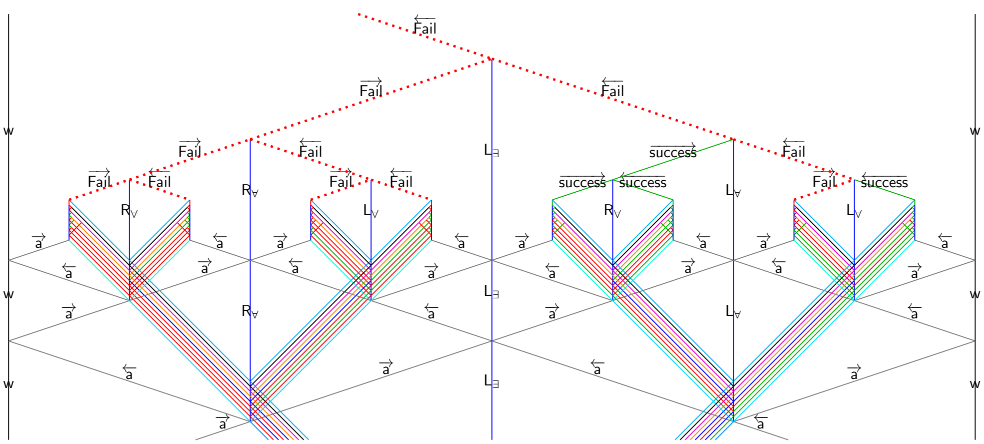


Fig. 6. Collecting the result.

All our research with abstract geometrical machines has been carried out on rational machines *i.e.* machines with only rational velocities and configurations, and all diagrams in this paper were automatically generated by Durand-Lose's Java-based simulator for signal machines.

We are currently furthering this research along two axes. First, we are considering how to tackle other complexity classes such as EXPTIME using abstract geometrical computation. Second, we would like to design a generic signal machine for Q-SAT, i.e. a single machine solving any instance of Q-SAT, where the formula is merely compiled into an initial configuration.

References

Eugene Asarin and Oded Maler. Achilles and the Tortoise climbing up the arithmetical hierarchy. In *FSTTCS '95*, number 1026 in LNCS, pages 471–483, 1995.

Olivier Bournez. Some bounds on the computational power of piecewise constant derivative systems. In *ICALP '97*, number 1256 in LNCS, pages 143–153, 1997.

Jérôme 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.

Jérôme 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.

Jérôme 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)*, pages 107–116. University of Athens, 2008.

Jérôme Durand-Lose. Abstract geometrical computation 3: Black holes for classical and analog computing. *Nat. Comput.*, 8(3):455–572, 2009a.

Jérôme Durand-Lose. Abstract geometrical computation and computable analysis. In J.F. Costa and N. Dershowitz, editors, *International Conference on Unconventional Computation 2009 (UC '09)*, number 5715 in LNCS, pages 158–167. Springer, 2009b.

Gustav A. Hedlund. Endomorphisms and automorphisms of the shift dynamical systems. *Mathematical Systems Theory*, 3(4):320–375, 1969.

Ulrich Huckenbeck. Euclidian geometry in terms of automata theory. *Theoret. Comp. Sci.*, 68(1):71–87, 1989.

Ulrich Huckenbeck. A result about the power of geometric oracle machines. *Theoret. Comp. Sci.*, 88(2):231–251, 1991.

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

Jarkko Kari. Theory of cellular automata: a survey. *Theoret. Comput. Sci.*, 334(1):3–33, 2005.

Jacques Mazoyer. Computations on one-dimensional cellular automata. *Annals of Mathematics and Artificial Intelligence*, 16:285–309, 1996.

Jacques Mazoyer and Véronique Terrier. Signals in one-dimensional cellular automata. *Theoret. Comput. Sci*, 217 (1):53–80, 1999.

Thomas J. Naughton and Damien Woods. On the computational power of a continuous-space optical model of computation. In M. Margenstern, editor, *Machines, Computations, and Universality (MCU '01)*, number 2055 in LNCS, pages 288–299, 2001.

John Von Neumann. Theory of self-reproducing automata. University of Illinois Press, 1966.