4 rue Léonard de Vinci
BP 6759
F-45067 Orléans Cedex 2
FRANCE
http://www.univ-orleans.fr/lifo

Rapport de Recherche

Solving Q-SAT in bounded space and time by geometrical computation (extended version)

Denys DUCHIER, Jérôme DURAND-LOSE and Maxime SENOT
LIFO, Université d'Orléans

Rapport n^o **RR-2011-03**

Solving Q-SAT in bounded space and time by geometrical computation

Denys Duchier, Jérôme Durand-Lose, Maxime Senot

LIFO, University of Orleans, BP 6759, F-45067 ORLEANS Cedex 2, FRANCE.

Abstract. Abstract geometrical computation can solve PSPACE-complete problems efficiently: any quantified boolean formula, instance of Q-SAT – the problem of satisfiability of quantified boolean formula – can be decided in bounded space and time with simple geometrical constructions involving only drawing parallel lines on an Euclidean space-time. Complexity as the maximal length of a sequence of consecutive segments is quadratic. We use the continuity of the real line to cover all the possible boolean valuations by a recursive tree structure relying on a fractal pattern: an exponential number of cases are explored simultaneously by a massive parallelism.

Keywords. Abstract geometrical computation; Signal machine; Q-SAT; Fractal computation; Massive parallelism; Unconventional computation.

1 Introduction

When defining and studying a new model of computation, especially an unconventionnal one, these questions arise naturally: what can we compute (in terms of decidability), how can we compute it, and what does it cost (in terms of complexity)? Answers could be found by taking representative problems of classical complexity classes, e.g. SAT for NP or Q-SAT for PSPACE, and coding them in the new computation model. This was done for NP-problems with active membranes system [Păun, 2001] and with an hyperbolic space of cellular automata [Margenstern and Morita, 2001]. Similarly, some solutions for Q-SAT were proposed with P-systems and membranes [Alhazov and Pérez-Jiménez, 2007], and with closed timelike curves in relativistic computation. We showed in [Duchier et al., 2010] that *signal machines*, a geometrical and abstract model of computation, are capable of solving SAT in bounded space and time. In the present paper, we extend this result to the higher complexity class PSPACE by describing a geometrical construction solving Q-SAT through *fractal parallelization*, still in constant space and time.

We also offer a more pertinent, model-specific, notion of time-complexity, namely *collision depth*, which is quadratic for our proposed construction.

The geometrical context proposed here is the following: 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 lines segment that we call *signals*. The space-time diagram of a signal machine constitued a *geometrical computation*.

Models of computation, conventional or not, are frequently based on mathematical idealizations of physical concepts and investigate the consequences, on computational power, of such abstractions (quantum, membrane, closed time-like curves, black holes. . .). However, oftentimes, the idealization is such that it must be interpreted either as allowing information to have infinite density (e.g. an oracle), or to be transmitted at infinite speed (global clock, no spatial extension. . .). On this issue, the model of signal machines stands in contradistinction with other abstract models of computation: it respects the principle of causality, density and speed of information are finite, as are the sets of objects manipulated. Nonetheless, it remains a resolutely abstract model with no apriori ambition to be physically realizable; it deals with theoretical issues such as computational power.

It is possible to do Turing-computation with such a system [Durand-Lose, 2005] and even to do analog computation by a systematic use of the continuity of space and time [Durand-Lose, 2009a,b]. Other *geometrical* models of computation exist and allow to compute: colored universes [Jacopini and Sontacchi, 1990], geometric machines [Huckenbeck, 1989], piece-wise constant derivative systems [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, seen as a continuous extension of cellular automata, is inherently parallel (see Fig. 1). In the present paper, we describe a massively parallel evaluation of all possible valuations for a given propositional formula and we provide a way to collect the results. This is the first time that parallelism is really used in AGC.

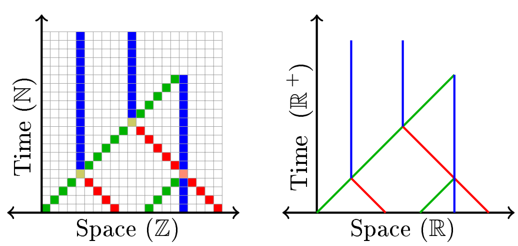


Fig. 1. From cellular automata to signal machines.

To achieve massive parallelism, we follow a fractal pattern to a depth of n (for n propositional variables) in order to partition the space in 2^n regions corresponding to the 2^n possible valuations of the unquantified formula. We call the resulting geometrical construction the *combinatorial comb* of propositional assignments. With a signal machine, such an exponential construction fits in bounded space and time regardless of the number of variables.

Once the combinatorial comb is in place, it is used to implement a binary decision tree for evaluating the formula, where all branches are explored in parallel. Finally, all the results are collected and aggregated respecting the quantifiers of the Q-SAT formula to yield the final answer. Our construction proceeds in stages: we generate and calibrate a beam of signals encoding the formula, making sure that it fits in the combinatorial comb, we propagate it through the binary decision tree, we compute the truth value when reaching each valuation, and finalize the answer at the top of the diagram.

Signal machines are presented in Section 2. Sections 3 to 7 detail step by step our geometrical solution to Q-SAT: splitting the space, coding the formula, broadcasting the formula, evaluating it and finalizing the answer by collecting the results. Complexities are discussed in Section 8 and conclusion and remarks are gathered in Section 9.

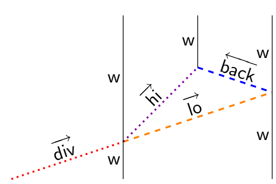
2 Definitions

Satisfiability of quantified boolean formulae. Q-SAT is the satisfiability problem for quantified boolean formulae (QBF). A QBF is a closed formula of the form: $\phi = Qx_1 Qx_2 \dots Qx_n \quad \psi(x_1, x_2, \dots, x_n)$ where $Q \in \{\exists, \forall\}$ and ψ is a quantifier-free formula of propositional logic. SAT is the fragment of Q-SAT using only the existential quantifier.

Q-SAT is *PSPACE-complete* [Stockmeyer and Meyer, 1973]: it can be solved by a polynomial-space algorithm and any PSPACE-problem can be reduced in polynomial time to Q-SAT. The classical algorithm is recursive: given a formula $Qx\ \phi(x)$, it recursively determines the satisfiability of $\phi(\texttt{true})$ and $\phi(\texttt{false})$, then aggregates the results with $\vee$ if $Q = \exists$ or with $\wedge$ if $Q = \forall$.

Signal machines. Signal machines are an extension of cellular automata from discrete time and space to continuous time and space. Dimensionless signals/particles move along the real line and rules describe what happens when they collide. *Signals.* Each *signal* is an instance of a *meta-signal*. The associated meta-signal defines its *velocity* and what happen 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. Meta-signals are listed on the left of Fig. 2.

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



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

Fig. 2. Computing the middle

Generally, we use over-line arrows to indicate the direction of propagation of a meta-signal. For example, $\overleftarrow{\mathsf{a}}$ and $\overrightarrow{\mathsf{a}}$ denote two different meta-signals; but as can be expected, they have similar uses and behaviors. Similarly $\mathsf{b_r}$ and $\mathsf{b_l}$ are different; both are stationary, but one is meant to be the version for right and the other for left.

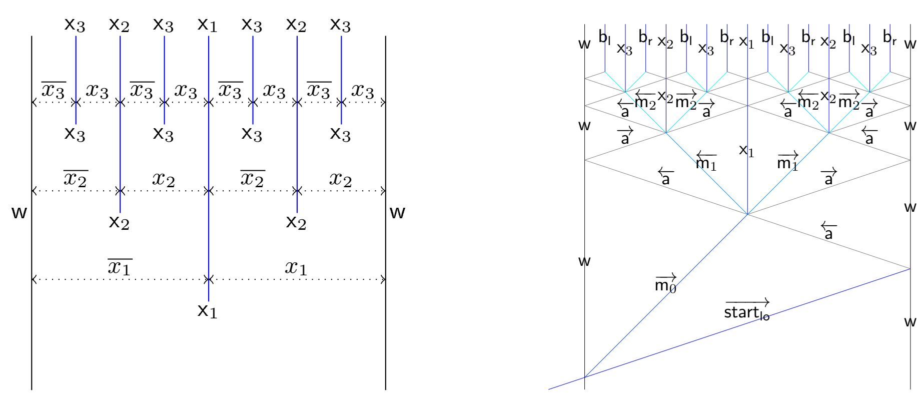
Collision rules. When a set of signals collide, they are replaced by a new set of signals according to a matching collision rule $\{\sigma_1, \ldots, \sigma_n\} \to \{\sigma'_1, \ldots, \sigma'_p\}$ where all σ_i and σ'_j are meta-signals. A rule matches a set of colliding signals if its left-hand side 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 listed on the right of this figure.

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 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 halfway between the initial two w.

3 Combinatorial comb

In order to determine by brute force whether a unquantified propositional formula with n variables is satisfiable, 2^n cases must be considered. These cases can be recursively enumerated using a binary decision tree.

The intuition is that the decision for variable x_i will be represented by a stationary signal: the space on the left should be interpreted as $x_i = \texttt{false}$, and the space on the right as $x_i = \texttt{true}$. Then we will similarly subdivide the spaces to the left and to the right, with stationary signals for x_{i+1}, and so on recursively for all variables as illustrated in Fig. 3(a).



(a) Cases identification (b) Division process

Fig. 3. Combinatorial comb for 3 variables.

Starting with two bounding signals w and an initiator $\overrightarrow{\text{start}}$, space is recursively divided as shown in Fig. 3(b). The first step works exactly as in Fig. 2, but then continues on to a depth of n: the counting is realized by using successively $\overrightarrow{\mathsf{m}_0}$, $\overrightarrow{\mathsf{m}_1}$, $\overrightarrow{\mathsf{m}_2}$ … The necessary rules and meta-signals are summarized in Tab.1.

Meta-Signal	Speed
$\overrightarrow{\mathsf{start}}$, $\overrightarrow{\mathsf{start}_{\mathsf{lo}}}$, $\overrightarrow{\mathsf{a}}$	3
$\overrightarrow{\mathsf{m}_0}$, $\overrightarrow{\mathsf{m}_1}$, $\overrightarrow{\mathsf{m}_2}$ …	1
x_1, x_2, x_3 …	0
$\overleftarrow{\mathsf{m}_0}$, $\overleftarrow{\mathsf{m}_1}$, $\overleftarrow{\mathsf{m}_2}$ …	-1
$\overleftarrow{\mathsf{a}}$	-3
$\mathsf{b_l}$, $\mathsf{b_r}$	0

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

Table 1. Meta-Signals and collision rules to build the comb.

Since each level of the tree is half the height of the previous one, the full tree can be constructed in bounded time regardless of its size. Also, note that the bottom level of the tree is not x_n but $\mathsf{b_r}$ and $\mathsf{b_l}$. These are used both to evaluate the formula and to aggregate the results as explained later.

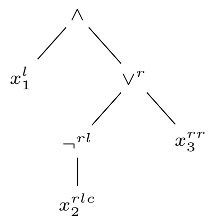
4 Formula encoding

In this section, we will explain how to represent the formula as a set of signals. This is illustrated with a running example:

$$\phi \ = \ \exists x_1 \forall x_2 \forall x_3 \quad x_1 \wedge (\neg x_2 \vee x_3) \quad .$$

We consider the quantifier-free subformula of ϕ: $x_1 \wedge (\neg x_2 \vee x_3)$ which can be viewed as a tree whose nodes are labeled by symbols (connectives and variables). The evaluation of the formula for a given assignment is a bottom-up process that percolates from the leaves toward the root. In order to model that process, we shall represent each node of the tree by a signal. In Fig. 4(a), each node is additionally decorated with a *path* from the root uniquely identifying its position in the tree: thus we are able to conveniently distinguish multiple occurrences of the same symbol. These decorated symbols provide convenient names for the required meta-signals (see Fig. 4(b)). Thus a formula of size t requires the definition of $2t$ meta-signals.

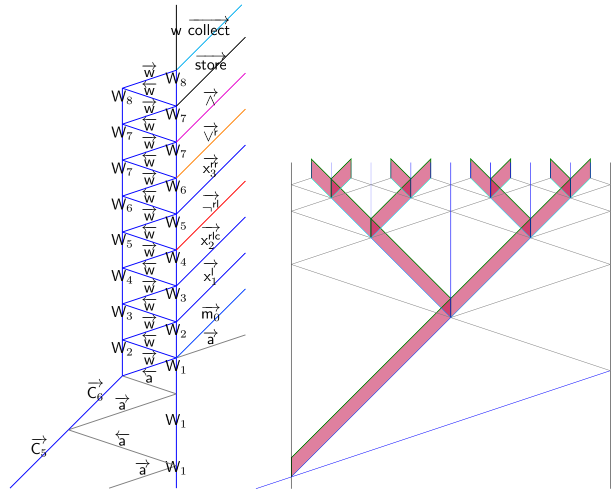
The signals for all subformulae are sent along parallel trajectories and form a *beam*. They are stacked in the diagram in order of nesting, inner-most subformulae first. This order is important for the process of percolation that will take place at the end. The width of the beam must be calibrated to have a proper propagation through the tree: it must be sufficiently narrow to fit in the top level . We explain and prove in Appendix A how to obtain geometrically this condition of narrowness.



(a) Labeled tree

Meta-Signal	Speed
$\overleftarrow{\wedge}$, $\overleftarrow{\vee^{\mathsf{r}}}$, $\overleftarrow{\neg^{\mathsf{rl}}}$	-1
$\overleftarrow{\mathsf{x}_1^{\mathsf{l}}}$, $\overleftarrow{\mathsf{x}_2^{\mathsf{rlc}}}$, $\overleftarrow{\mathsf{x}_3^{\mathsf{rr}}}$	-1
$\overrightarrow{\wedge}$, $\overrightarrow{\vee^{\mathsf{r}}}$, $\overrightarrow{\neg^{\mathsf{rl}}}$	1
$\overrightarrow{\mathsf{x}_1^{\mathsf{l}}}$, $\overrightarrow{\mathsf{x}_2^{\mathsf{rlc}}}$, $\overrightarrow{\mathsf{x}_3^{\mathsf{rr}}}$	1

(b) Generated signals



(c) Initial displaying (d) Corridors

Fig. 4. Compiling the formula

5 Propagating the beam

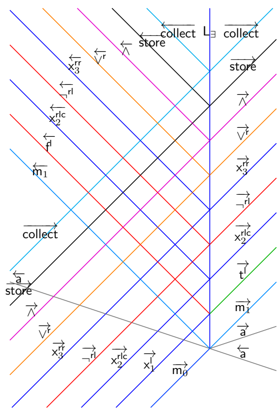
The formula's beam is now propagated down the decision tree. For each decision point, the beam is duplicated: one part goes through, the other is reflected. Thus, by construction, every branch of the beam tree encounters a decision point for every variable at least once. If we make the beam sufficiently narrow, the guarantee become "exactly once," as shown in Fig. 4(d).

When the beam encounters a decision point (a stationary signal for a variable x_i), then a split occurs producing two branches. Except for the sign of their velocity, most signals remain identical in both branches; most, except those corresponding to occurrences of x_i: those become `false` in the left branch and `true` in the right branch. Fig. 5(a) shows the beam intersecting the decision signal for variable x_1. Note how the incident signal $\overrightarrow{\mathsf{x}_1^{\mathsf{l}}}$ becomes $\overleftarrow{\mathsf{f}^{\mathsf{l}}}$ on the left and $\overrightarrow{\mathsf{t}^{\mathsf{l}}}$ on the right; the path decoration is preserved since, as we shall see, it is essential later for the percolation process. This is achieved by the collision rule: $\{\overrightarrow{\mathsf{x}_1^{\mathsf{l}}}, \mathsf{x}_1\} \rightarrow \{\overleftarrow{\mathsf{f}^{\mathsf{l}}}, \mathsf{x}_1, \overrightarrow{\mathsf{t}^{\mathsf{l}}}\}$. Since a decision point is encountered exactly once for each variable on each branch of the lane, at the bottom of the tree, all signals corresponding to occurrences of variables have been assigned a boolean value.

6 Evaluating the formula

Remember how, at the very bottom of the decision tree, we added an extra division using signals $\mathsf{b_l}$ or $\mathsf{b_r}$: their purpose is to initiate the percolation process.

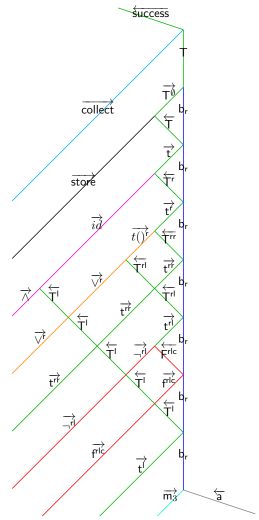
$\mathsf{b_l}$ is for starting the percolation process of a left branch, while $\mathsf{b_r}$ is for a right branch. Figure 5(c) zooms on one case of our example.



(a) Split

$$\{\ \overrightarrow{\vee^{\mathsf{r}}},\ \overleftarrow{\mathsf{T}^{\mathsf{rl}}}\ \} \to \{\ \overrightarrow{t()^{\mathsf{r}}}\ \}$$
$$\{\ \overrightarrow{t()^{\mathsf{r}}},\ \overleftarrow{\mathsf{T}^{\mathsf{rr}}}\ \} \to \{\ \overrightarrow{\mathsf{t}^{\mathsf{r}}}\ \}$$
$$\{\ \overrightarrow{id^{\mathsf{r}}},\ \overleftarrow{\mathsf{T}^{\mathsf{rr}}}\ \} \to \{\ \overrightarrow{\mathsf{t}^{\mathsf{r}}}\ \}$$
$$\{\ \overrightarrow{\vee^{\mathsf{r}}},\ \overleftarrow{\mathsf{F}^{\mathsf{rl}}}\ \} \to \{\ \overrightarrow{id^{\mathsf{r}}}\ \}$$
$$\{\ \overrightarrow{t()^{\mathsf{r}}},\ \overleftarrow{\mathsf{F}^{\mathsf{rr}}}\ \} \to \{\ \overrightarrow{\mathsf{t}^{\mathsf{r}}}\ \}$$
$$\{\ \overrightarrow{id^{\mathsf{r}}},\ \overleftarrow{\mathsf{F}^{\mathsf{rr}}}\ \} \to \{\ \overrightarrow{\mathsf{f}^{\mathsf{r}}}\ \}$$

(b) Collision rules to evaluate the disjunction $\vee^r$



(c) Evaluation at the bottom of the comb

Fig. 5. Split, evaluation process and rules for a connective.

The invariant is that all signals that reach $\mathsf{b_r}$ have determined boolean values. When $\overrightarrow{\mathsf{t}^{\mathsf{l}}}$ reaches $\mathsf{b_r}$, it gets reflected as $\overleftarrow{\mathsf{T}^{\mathsf{l}}}$. 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 $\mathsf{b_r}$. This enforces the invariant.

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

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

7 Collecting the results

At the end of the propagation phase, the results of evaluating the unquantified formula for all possible assignments have been stored as stationary signals replacing the $\mathsf{b_l}$ and $\mathsf{b_r}$ signals. We must now evaluate the quantified formula.

Remember that each level of the comb corresponds to a variable: level 1 stands for $x_1, \ldots$ For aggregating all the results, we will "undo" the construction process of the binary tree by mixing two by two the results of evaluations with respect to the initial quantifiers.

At each split of the beam – *i.e.* when it meets the stationary signals coding x_i at the i^{th} level of the tree – the signal $\overrightarrow{\mathsf{collect}}$ (resp. $\overleftarrow{\mathsf{collect}}$) (at the very top of the beam) changes the stationary x_i into a stationary $\mathrm{L}^i_{\mathrm{Q}_i}$ (resp. $\mathrm{R}^i_{\mathrm{Q}_i}$), where Q_i denotes the quantifier for x_i in ϕ, and L (resp. R) indicates the direction in which to emit the combined result of trying $x_i = \texttt{false}$ (coming from the left) and $x_i = \texttt{true}$ (coming from the right). The boolean connective to effect the combination depends on the type of the quantifier: $\vee$ for $\exists$ and $\wedge$ for $\forall$. This collection process is illustrated in Fig. 6. The required rules are given in Appendix B.

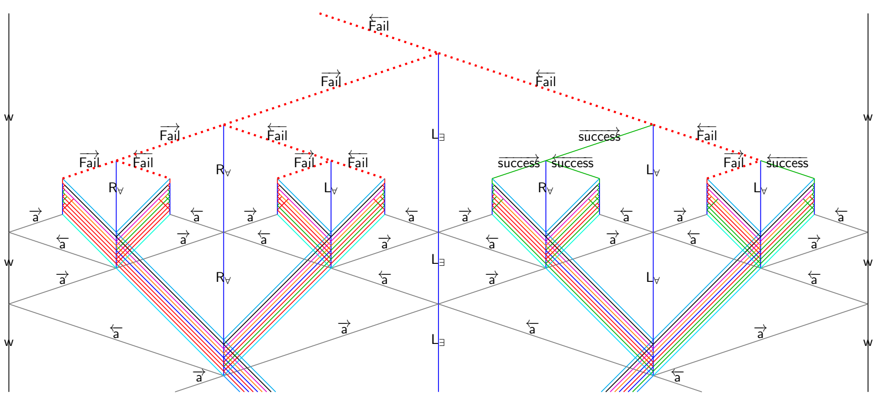


Fig. 6. Collecting the results.

The space-time diagram for the entire construction is shown in Fig. 7.

8 Complexities

We now turn to a crucial question: what is the complexity of our construction as a function of the size of the formula? What is a meaningful way to measure this complexity?

The width of the construction measures the space requirement: it is independent of the formula and can be fixed to any value we like. The height measures the time requirement: it is also independent of the formula because of the fractal construction and the continuity of space-time. If more variables are involved, the comb gains extra levels, but its height remains bounded by a fractal, the infinite binary tree.

As a consequence, while width (space) and height (time) are the natural continuous extensions of traditional complexity measures used in the discrete universe of cellular automata, in the context of abstract geometrical computations, they loose all pertinence.

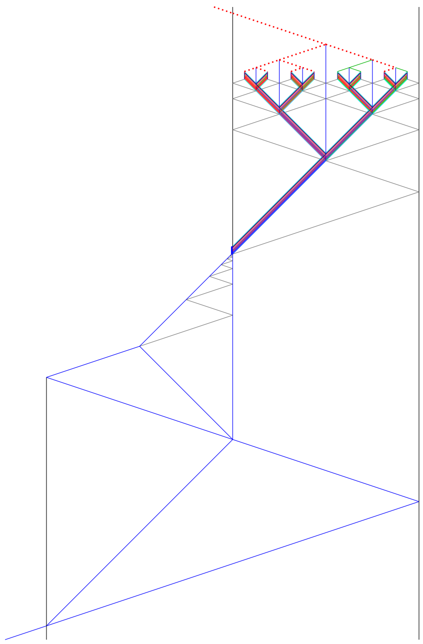


Fig. 7. The whole diagram.

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.

Thus, if we view the diagram as an acyclic graph of collisions (vertices) and signals (arcs), time complexity can then be defined as the maximal size of a chain of collisions and space complexity as the maximal size of an anti-chain *i.e.* a set of signals pairwise un-related.

Let t be the size of the formula and n the number of variables. At the bottom level of the comb, there is an anti-chain of length approximately $t2^n$, making the space complexity exponential. Generation of the comb, initiation, propagation, evaluation and aggregation contribute along any path a number of collisions at most linear in the size of the formula. However, intersections of incident and reflected branches at every level add $O(nt)$ because there are $O(n)$ levels and the beam consists of $O(t)$ parallel signals. Thus the time complexity is $O(nt)$.

It should also be pointed out that the compilation into a rational signal machine is done in polynomial time and only five distinct speeds are involved. Algorithms to generate the machine from the formula are given in App. B.

9 Conclusion

In this article, we have shown how to achieve massive parallelism with signal machines, by means of a fractal pattern. We call this *fractal parallelism* and it is a novel contribution in the field of abstract geometrical computation. We have described how this approach is able to solve Q-SAT in bounded space and time.

The complexity is not hidden inside the compilation of the machine (done in quadratic time) nor in the initial configuration (which is very simple) and all the speeds are constant. Since, clearly, time and space are no longer appropriate measures of complexity, we have also proposed to replace them respectively by the maximum size of a chain and an anti-chain in the space-time diagram regarded as a directed acyclic graph. According to these new definitions, our construction has exponential space complexity and quadratic time complexity.

It seems natural to draw a connexion between the construction we propose and Boolean circuits, and we plan to investigate the relation. However, signal machines address an additional concern, which Boolean circuits do not; namely, that computations take place not only over time, but also over physical space, and that, in this respect, they are potentially limited by any bound placed on the speed at which information may travel.

Any formula can be compiled into a signal machine. The next step is to provide a single signal machine that is generic for Q-SAT —*i.e.* it can solve any instance of Q-SAT— so that the formula would only have to be compiled into an initial configuration. Additionally, we will continue our investigations with complexity classes higher in the hierarchy, such as EXPTIME or EXPSPACE.

Bibliography

A. Alhazov and M. J. Pérez-Jiménez. Uniform solution of Q-SAT using polarizationless active membranes. In J. Durand-Lose and M. Margenstern, editors, *Machines, Computations, and Universality, 5th International Conference (MCU '07)*, number 4664 in LNCS, pages 122–133. Springer, 2007.

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

D. Duchier, J. Durand-Lose, and M. Senot. Fractal parallelism: Solving SAT in bounded space and time. In O. Cheong, K.-Y. Chwa, and K. Park, editors, *21st International Symposium on Algorithms and Computation (ISAAC '10)*, number 6506 (Part 1) in LNCS, pages 279–290. Springer, 2010.

J. Durand-Lose. Abstract geometrical computation: Turing computing ability and undecidability. In B. 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.

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

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

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

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

M. Margenstern and K. Morita. NP problems are tractable in the space of cellular automata in the hyperbolic plane. *Theoret. Comp. Sci.*, 259(1-2):99–128, 2001.

T. J. Naughton and D. 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.

G. Păun. P systems with active membranes: Attacking NP-Complete problems. *Journal of Automata, Languages and Combinatorics*, 6(1):75–90, 2001.

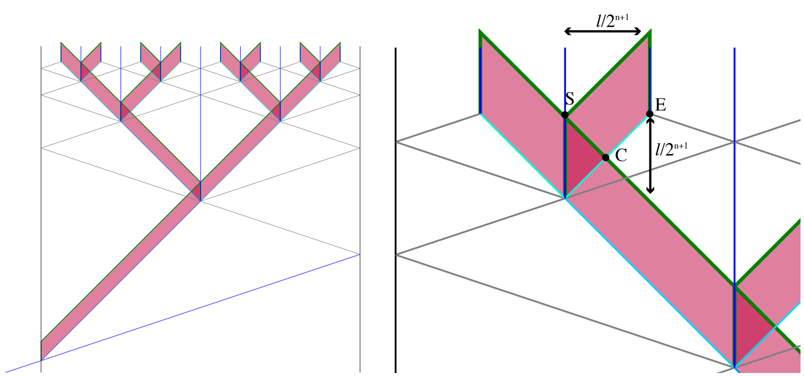
L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. In *5th ACM Symposium on Theory of Computing (STOC '73)*, volume 16, pages 1–9, 1973.

A Calibrating the beam

In this appendix, we formally prove conditions of validity for the construction and we detail the computation of the width of the beam.

The initiation follows two steps: we compute first the distance between the parallel signals composing the beam of the formula and then we generate this beam by creating successively the signals for each nodes of the formula, with respect to the spacing computed first.

The propagation of the beam through the tree is displayed in Fig. 8(a). To get a proper propagation and to ensure that signals corresponding to variables will be assignated a boolean value exactly once and that an evalution case will not interfere with another distinct one, the width of the beam must be calibrated in function of the height of the last level. Let l be the width of the comb and n the number of variables. At the bottom of the comb, space is split in 2^{n+1} equal parts, and so each part has a width $\frac{l}{2^{n+1}}$. Splitting signals have speed 1 (or -1) so that the delay between the last split and the evaluation level is also $\frac{l}{2^{n+1}}$. The condition of correctness of the propagation is that the width must be less than $\frac{l}{2^{n+1}}$. This condition on the propagation at the final level can be seen on Fig. 8(b): we want the splitting point S to happen before the starting of the evaluation marqued by E *i.e.* the temporal coordinate of S is less than the temporal coordinate of E. This ensures too that the crossing point C happens stricly before the evalution and that the splitting and the evaluation processes are independent in the time (evalution begins strictly after the end of the split). In this case, the evaluation of the beam for one case striclty takes place in its specific top area and does not affect the other evaluation cases.



(a) Through the tree. (b) At the top level.

Fig. 8. Conditions for a good propagation

This condition on the width of the beam is realized by the initiation step which computes geometrically the delay between signals forming the beam (this delay is the same between all the signals of the beam). Call n the number of variables involved in the formula ϕ and t its size. Let w_ϕ be the width of the beam.

All the signals required for compiling the formula have to be contained in the beam, so its width must be divised by $t+3$ to get the proper spacing between signals of the formula (we add 3 at the size t of the formula for signals useful for building the comb and collecting the final results). But instead of dividing the width w_ϕ exactly by $t+3$, we use our initial method for computing the middle, that is dividing by two, and we recursively divide by halves until we get a sufficiently narrow spacing. Thus the distance between two signals of the beam will be of the form $\frac{l}{2^i}$ where i is the number of consecutive divisions by half and so $w_\phi = (t+2).\frac{l}{2^i}$ (we have $t+2$ spacing in the beam since it is composed by $t+3$ parallel signals). For providing this, let consider

$$\tau_\phi \;=\; min\; \left\{ i \in \mathbb{N} \mid \tfrac{(t+2).2^{n+1}}{l} \le 2^i \right\} \;\; .$$

Such an integer τ_ϕ exists and is the smallest number of divisions by half ensuring a sufficient small size of the beam. Starting from a distance of l, we will get a spacing of $2^{-\tau_\phi}$, standing at the most right part of our bounded space of work as we can see in the right top of Fig. 9. We call $\delta = \frac{1}{2^{\tau_\phi}}$.

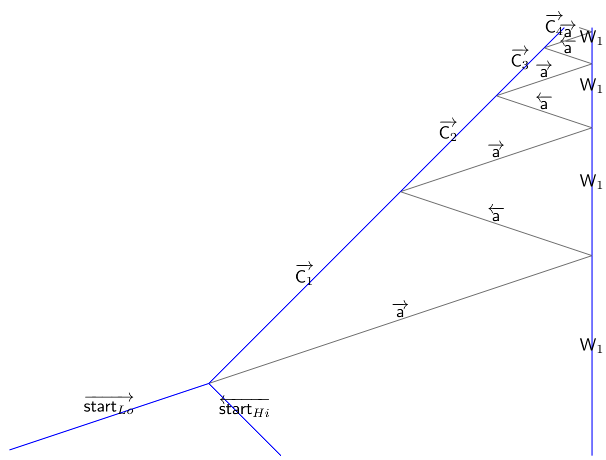
Meta-Signal	Speed
$\overrightarrow{\mathsf{start}_{Lo}}$, $\overrightarrow{\mathsf{a}}$, $\overrightarrow{\mathsf{w}}$	3
$\overrightarrow{\mathsf{c}_1} \ldots \overrightarrow{\mathsf{c}_{\tau_\phi}}$	1
W_1, W_2	0
$\overleftarrow{\mathsf{start}_{Hi}}$	-1
$\overleftarrow{\mathsf{a}}$	-3

(a) Meta-Signals

Collision rules

$$\{ \overrightarrow{\mathsf{start}_{Lo}}, \overleftarrow{\mathsf{start}_{Hi}} \} \to \{ \overrightarrow{\mathsf{c}_1}, \overrightarrow{\mathsf{a}} \}$$
$$\{ \overrightarrow{\mathsf{a}}, \mathsf{W}_1 \} \to \{ \overleftarrow{\mathsf{a}}, \mathsf{W}_1 \}$$
$$\{ \overrightarrow{\mathsf{c}_i}, \overleftarrow{\mathsf{a}} \} \to \{ \overrightarrow{\mathsf{c}_{i+1}}, \overrightarrow{\mathsf{a}} \}$$
$$\{ \overrightarrow{\mathsf{c}_{\tau\phi}}, \overleftarrow{\mathsf{a}} \} \to \{ \mathsf{W}_2, \overrightarrow{\mathsf{w}} \}$$

(b) Rules



(c) Computing the delay

Fig. 9. Computing the delay

This spacing between signals of the beam is suitable for ensuring a good propagation and the validity of the final evaluation. Indeed, the beam consists in $t+3$ parallel signals distant of $2^{-\tau_\phi}$, so its total width is $w_\phi = (t+2).2^{-\tau_\phi}$. By definition of τ_ϕ, we have:

$$\frac{(t+2).2^{n+1}}{l} \le 2^{\tau_\phi}$$

i.e.

$$\frac{t+2}{2^{\tau_\phi}} \le \frac{l}{2^{n+1}}$$

and therefore

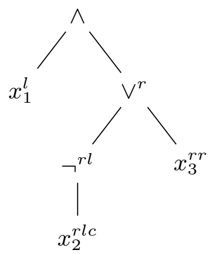
$$w_\phi \le \frac{l}{2^{(n+1)}} \;\; .$$

It has to be underlined that the number τ_ϕ can be computed in polynomial time. More precisely, it can computed at most in quadratic time in t, the size of ϕ.

This construction takes place in the left part of our workspace (the right one being used for the comb and evaluation of the formula).

After the step of computing the delay, the beam is generated as seen in Fig. 10(c): signals moving between the two wall spaced by δ create with a delay of $\frac{2}{3}.\delta$ the parallel signals encoding the formula (the delay is $\frac{2}{3}.\delta$ because of the back and forth of signals of speed 3 between walls).

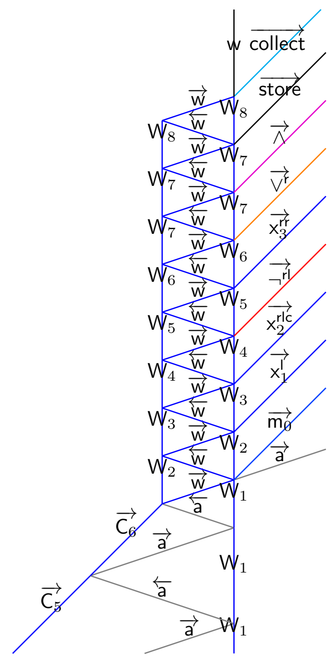
We get then parallel signals moving at speed 1 forming the formula's beam, which has the proper size to cover all the comb branches once and only once. The whole initiation process is displayed in Fig. 11.



(a) Labeled tree

Meta-Signal	Speed
$\overrightarrow{\mathsf{w}}$	3
$\overrightarrow{\wedge}$, $\overrightarrow{\vee^{\mathsf{r}}}$, $\overrightarrow{\neg^{\mathsf{rl}}}$, $\overrightarrow{\mathsf{x}_1^{\mathsf{l}}}$, $\overrightarrow{\mathsf{x}_2^{\mathsf{rlc}}}$, $\overrightarrow{\mathsf{x}_3^{\mathsf{rr}}}$, $\overrightarrow{\mathsf{store}}$, $\overrightarrow{\mathsf{collect}}$	1
W_1, W_2, W_3, W_4, W_5, W_6, W_7, W_8	0
$\overleftarrow{\wedge}$, $\overleftarrow{\vee^{\mathsf{r}}}$, $\overleftarrow{\neg^{\mathsf{rl}}}$, $\overleftarrow{\mathsf{x}_1^{\mathsf{l}}}$, $\overleftarrow{\mathsf{x}_2^{\mathsf{rlc}}}$, $\overleftarrow{\mathsf{x}_3^{\mathsf{rr}}}$, $\overleftarrow{\mathsf{store}}$, $\overleftarrow{\mathsf{collect}}$	-1
$\overleftarrow{\mathsf{w}}$	-3

(b) Generated signals



(c) Generating the formula

Fig. 10. Compiling the formula

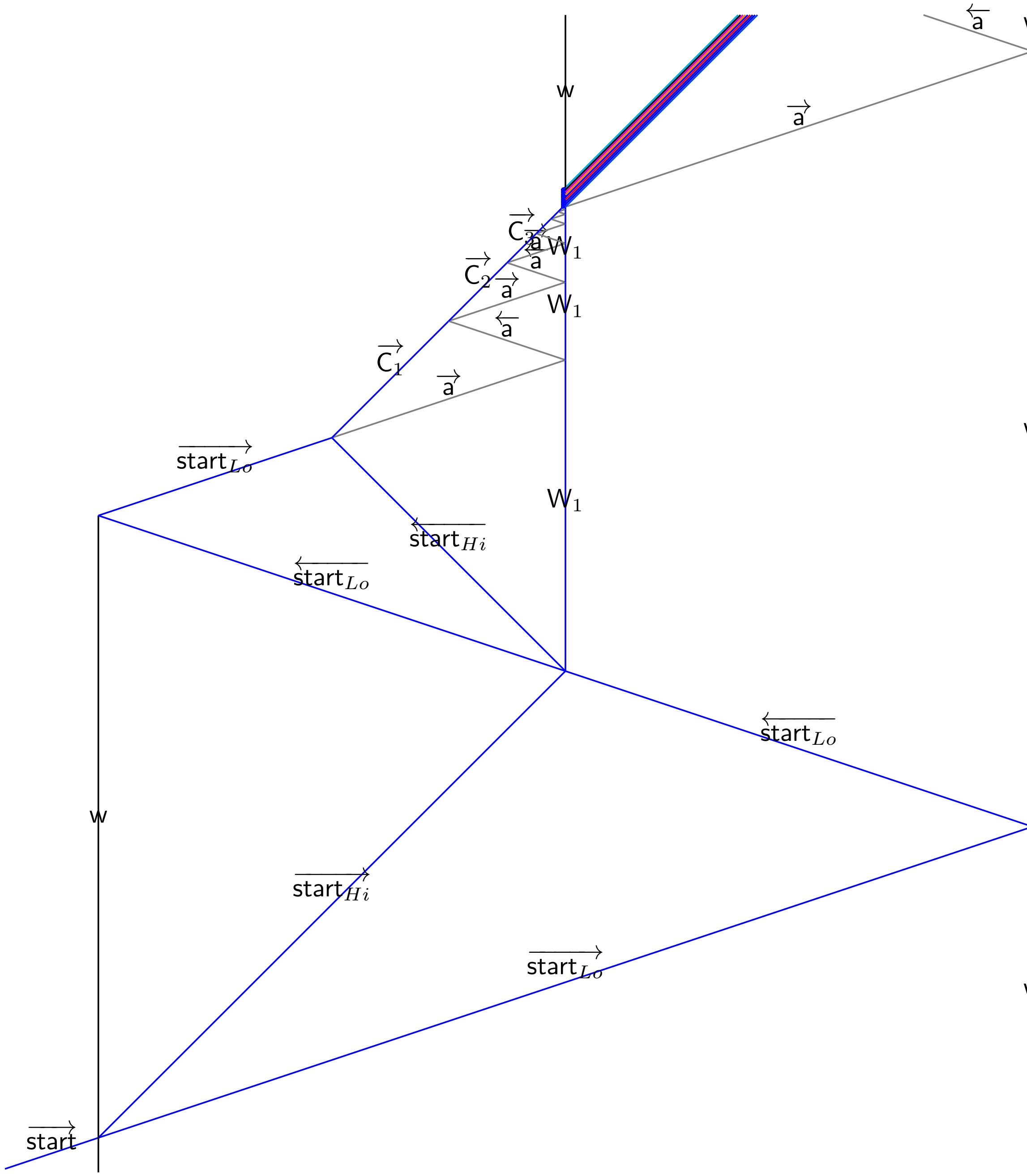


Fig. 11. The whole initiation.

B Compiling the signal machine

We give in this appendix the main algorithms to compile the signal machine corresponding to a Q-SAT formula. The algorithms given here completed by the tables of meta-signals and collision rules fully describe the signal machine compiled from a Q-SAT formula. We follow the steps described in the paper. Input of the algorithms are the signal machines obtained from the previous algorithms, so that the final signal machine that solves Q-SAT for a given instance ϕ is defined by all the meta-signals and collision rules generated by the tables and the algorithms. We eventually obtained a machine which depends on the Q-SAT formula, but which is generated in quadratic time in the size of the formula.

B.1 Constructing the comb

We provide in ths subsection the meta-signals and the rules used to build the combinatorial comb. Meta-signals and collision rules which are independent of the Q-SAT formula are gathered in Tab. 2. Others signals and rules depending on the number of variables n are generated by Algo. 1.

Meta-Signal	Speed
$\overrightarrow{\mathsf{start}}$, $\overrightarrow{\mathsf{start_{lo}}}$, $\overrightarrow{\mathsf{a}}$	3
$\overrightarrow{\mathsf{m_0}}$	1
$\overleftarrow{\mathsf{a}}$	-3
$\mathsf{b_l}$, $\mathsf{b_r}$	0

Collision rules
{ $\overrightarrow{\mathsf{start}}$, w } $\rightarrow$ { w, $\overrightarrow{\mathsf{start_{lo}}}$, $\overrightarrow{\mathsf{m_0}}$ }
{ $\overrightarrow{\mathsf{start_{lo}}}$, w } $\rightarrow$ { $\overleftarrow{\mathsf{a}}$, w }
{ w, $\overleftarrow{\mathsf{a}}$ } $\rightarrow$ { w, $\overrightarrow{\mathsf{a}}$ }
{ $\overrightarrow{\mathsf{a}}$, w } $\rightarrow$ { $\overleftarrow{\mathsf{a}}$, w }

Table 2. Meta-signals and rules independent of the formula to build the comb.

B.2 Initiation and calibrating the beam

We resume τ_ϕ the integer computed in Appendix A: τ_ϕ is the number of division by halves to get a correct spacing, used to generate the beam. Meta-signals and rules used in the calibrating process of the beam and the computation of the delay between signals of the beam – as described in Appendix A – are summarized in Tab.3. Table 4 lists the objects used in the pre-initition step, which consists in starting all the construction and in separating the space in two parts: one for calibrating the beam and one for the comb and the evaluation of the formula.

B.3 Compiling the formula

We explain next how a propositional formula can be viewed as a labeled tree and how it is then used to generate the meta-signals and the collision rules corresponding to the formula, what is done by Algo. 2 (in which t denotes the size of the propositional formula). We consider here only the quantifier-free subformula of the Q-SAT formula. The compiltaion for the runnig example is displayed in Fig. 10. Quantifiers are treated in the Subsect. B.6, during the collecting process.

input : $\mathcal{M}$ signal machine
output: $\mathcal{M}$ added with the rules to build the comb

`/* Meta-signals to build the comb */` ;
for *i=1 to n* **do**
  $\mathsf{x}_i \longleftarrow \mathcal{M}$.add.new_meta_signal_of_Speed(0) ;
  $\overrightarrow{\mathsf{m}_i} \longleftarrow \mathcal{M}$.add.new_meta_signal_of_Speed(1) ;
  $\overleftarrow{\mathsf{m}_i} \longleftarrow \mathcal{M}$.add.new_meta_signal_of_Speed(-1) ;
end

`/* Rules of collision to build the comb */` ;
for *i=1 to n* **do**
  $\mathcal{M} \longleftarrow$ add_rule($\{\overrightarrow{\mathsf{m}_i}, \overleftarrow{\mathsf{a}}\} \rightarrow \{\overleftarrow{\mathsf{a}}, \overleftarrow{\mathsf{m}_{i+1}}, \mathsf{x}_i, \overrightarrow{\mathsf{m}_{i+1}}, \overrightarrow{\mathsf{a}}\}$) ;
  $\mathcal{M} \longleftarrow$ add_rule($\{\overrightarrow{\mathsf{a}}, \overleftarrow{\mathsf{m}_i},\} \rightarrow \{\overleftarrow{\mathsf{a}}, \overleftarrow{\mathsf{m}_{i+1}}, \mathsf{x}_i, \overrightarrow{\mathsf{m}_{i+1}}, \overrightarrow{\mathsf{a}}\}$) ;
end
$\mathcal{M} \longleftarrow$ add_rule($\{\overrightarrow{\mathsf{m}_n}, \overleftarrow{\mathsf{a}}\} \rightarrow \{\mathsf{b_r}\}$) ;
$\mathcal{M} \longleftarrow$ add_rule($\{\overrightarrow{\mathsf{a}}, \overleftarrow{\mathsf{m}_n},\} \rightarrow \{\mathsf{b_l}\}$) ;

Algorithm 1: Building the comb

Meta-Signal	Speed	Collision rules
$\overrightarrow{\text{start}_{Lo}}$, $\overrightarrow{\mathsf{a}}$, $\overrightarrow{\mathsf{w}}$	3	$\{\overrightarrow{\text{start}_{Lo}}, \overleftarrow{\text{start}_{Hi}}\} \rightarrow \{\overrightarrow{\mathsf{c}_1}, \overrightarrow{\mathsf{a}}\}$
$\overrightarrow{\mathsf{c}_1} \ldots \overrightarrow{\mathsf{c}_{\tau_\phi}}$	1	$\{\overrightarrow{\mathsf{a}}, \mathsf{W}_1\} \rightarrow \{\overleftarrow{\mathsf{a}}, \mathsf{W}_1\}$
W_1, W_2	0	$\{\overrightarrow{\mathsf{c}_i}, \overleftarrow{\mathsf{a}}\} \rightarrow \{\overrightarrow{\mathsf{c}_{i+1}}, \overrightarrow{\mathsf{a}}\}$
$\overleftarrow{\text{start}_{Hi}}$	-1	$\{\overrightarrow{\mathsf{c}_{\tau_\phi}}, \overleftarrow{\mathsf{a}}\} \rightarrow \{\mathsf{W}_2, \overrightarrow{\mathsf{w}}\}$
$\overleftarrow{\mathsf{a}}$	-3	

Table 3. Meta-signals and rules for computing the delay

Meta-Signal	Speed	Collision rules
$\overrightarrow{\text{start}}$, $\overrightarrow{\text{start}_{Lo}}$	3	$\{\overrightarrow{\text{start}}, \mathsf{w}\} \rightarrow \{\mathsf{w}, \overrightarrow{\text{start}_{Hi}}, \overrightarrow{\text{start}_{Lo}}\}$
$\overrightarrow{\text{start}_{Hi}}$	1	$\{\overrightarrow{\text{start}_{Lo}}, \mathsf{w}\} \rightarrow \{\overleftarrow{\text{start}_{Lo}}, \mathsf{w}\}$
w, W_1	0	$\{\overrightarrow{\text{start}_{Hi}}, \overleftarrow{\text{start}_{Lo}}\} \rightarrow \{\overleftarrow{\text{start}_{Lo}}, \overleftarrow{\text{start}_{Hi}}, \mathsf{W}_1\}$
$\overleftarrow{\text{start}_{Hi}}$	-1	$\{\mathsf{w}, \overleftarrow{\text{start}_{Lo}}\} \rightarrow \{\overrightarrow{\text{start}_{Lo}}\}$
$\overleftarrow{\text{start}_{Lo}}$	-3	

Table 4. Meta-Signals and collision rules for the pre-initiation

We will now describe how to compile a propositional formula into a signal machine. This process exploits the fact that a propositional formula ϕ can be modeled as a *labeled tree*: its nodes can be identified by their paths from the root: a path π is a word over the alphabet $\mathbb{N}$, and they are labeled by connectives or propositional variables.

Labeled trees: We assume a signature Σ of function symbols f, g, $\ldots$, each of which is equipped with an arity $\mathsf{ar}(f) \geq 0$. We write $\mathbb{N}_0$ for $\mathbb{N} \setminus \{0\}$. A *tree domain* D is a finite subset of $\mathbb{N}_0^*$ which is closed for prefixes and for left-siblings; in other words it satisfies:

$$\begin{array}{ll} \forall \pi, \pi' \in \mathbb{N}_0^* & \pi\pi' \in D \;\Rightarrow\; \pi \in D \\ \forall \pi \in \mathbb{N}_0^*,\ \forall i, j \in \mathbb{N}_0 & i < j \wedge \pi j \in D \;\Rightarrow\; \pi i \in D \end{array}$$

A labeled tree $\tau = (D_\tau, L_\tau)$ consists of a tree domain D_τ and a labeling function $L_\tau : D_\tau \to \Sigma$ assigning a symbol to each node, respecting arities, *i.e.*:

$$\begin{array}{lcll} \mathsf{ar}(L_\tau(\pi)) = n & \Rightarrow & \pi n \in D_\tau \;\wedge\; \pi(n+1) \notin D_\tau & \forall \pi \in D_\tau, \forall n > 0 \\ \mathsf{ar}(L_\tau(\pi)) = 0 & \Rightarrow & \pi 1 \notin D_\tau & \forall \pi \in D_\tau \end{array}$$

Propositional formulae as labeled trees: We take the signature Σ to be formed from propositional variables (arity 0), the connective $\neg$ (arity 1), and the connectives $\wedge$ and $\vee$ (arity 2).

Compilation: Given a propositional formula $\phi = (D_\phi, L_\phi)$, we will emit a particle for each one of its nodes. Consider a node $\pi \in D_\phi$ with $L_\phi(\pi) = \ell$: its associated meta-signals will be noted $\overrightarrow{\ell^\pi}$ and $\overleftarrow{\ell^\pi}$ (respectively for the right-moving and the left-moving signals). We write $\mathcal{N}_\phi$ for the set of labelled nodes of ϕ, *i.e.*

$$\mathcal{N}_\phi = \{L_\phi(\pi)^\pi\}_{\pi \in D_\phi} = \{\ell^\pi\}_{\pi \in D_\phi}$$

The meta-signals coding the formula have all the same speed which is the speed of all the signals forming the beam: this speed is 1 (resp. -1) for the right-moving beam (resp. the left-moving).

We write $\prec$ for the lexicographic order on D_ϕ and $\lceil D_\phi \rceil$ for the $\prec$-maximal element in D_ϕ. We write $\lfloor \pi \rfloor$ for the $\prec$-predecessor of π in D_ϕ. If $\pi \prec \pi'$, then $\overrightarrow{L_\phi(\pi)^\pi}$ is emited later than $\overrightarrow{L_\phi(\pi')^{\pi'}}$.

Example: consider the propositional formula $\phi = x_1 \wedge (\neg x_2 \vee x_3)$ taken in example to generate all the diagram in this paper. It has six nodes and $\mathcal{N}_\phi = \{x_1^1,\ x_2^{211},\ \neg^{21},\ \vee^2,\ x_3^{22},\ \wedge^\varepsilon\}$.

To simplify the notations, we will write l, r, and c to designate respectively left, right and center in the path of a node of the formula (we will sometimes use ε to designate the empty word *i.e.* ε will be the path of the first node).
The previous example becomes $\mathcal{N}_\phi = \{x_1^l,\ x_2^{rlc},\ \neg^{rl},\ \vee^r,\ x_3^{rr},\ \wedge\}$.

input : $\mathcal{M}$ signal machine
output: $\mathcal{M}$ added with the rules of generation

```
/* Meta-signals coding the formula */ ;
foreach ℓ^π ∈ N_φ do
    →ℓ^π ⟵ M.add.new_meta_signal_of_Speed(1) ;
    ←ℓ^π ⟵ M.add.new_meta_signal_of_Speed(-1) ;
end
/* Meta-signals and walls to generate the beam */ ;
for i=1 to t+3 do
    W_i ⟵ M.add.new_meta_signal_of_Speed(0) ;
end
/* Rules for the bounces on left walls */ ;
for i=2 to t+2 do
    M ⟵ add_rule({W_i, ←w} → {W_{i+1}, →w}) ;
end
/* Rules for generating the beam */ ;
M ⟵ add_rule({→w, W_1} → {←w, W_2, →m0, →a}) ;
M ⟵ add_rule({→w, W_2} → {←w, W_3, →ℓ^⌈D_φ⌉}) ;
i ⟵ 2 ;
foreach ℓ^π ∈ N_φ \ {ℓ^⌈D_φ⌉} do
    if π = ε then
        M ⟵ add_rule({→w, W_{t+1}} → {←w, W_{t+2}, →ℓ^ε}) ;
    else
        M ⟵ add_rule({→w, W_i} → {←w, W_{i+1}, →ℓ^π}) ;
        i ⟵ i + 1 ;
    end
end
M ⟵ add_rule({→w, W_{t+2}} → {←w, W_{t+3}, →store}) ;
M ⟵ add_rule({W_{t+3}, ←w} → {→w}) ;
M ⟵ add_rule({→w, W_{t+3}} → {w, →collect}) ;
```

Algorithm 2: Generating the formula

B.4 Broadcasting the formula

In this subsection, the main algorithm is given by Algo. 3. Whereas all the other algorithms are in linear time in the size of the formula, Algo. 3 is in quadratic time because of its two imbricated loops. It makes the compilation of the signal machine be in quadratic time. We note Q_i the quantifier linking the variable x_i in the Q-SAT formula.

B.5 Evaluating the formula

We just propose here Tab.5 to give the idea of how rules of evaluation are defined: it respect the classical boolean operations. The reconstruction process follows the reverse

input : $\mathcal{M}$ signal machine
output: $\mathcal{M}$ added with the rules of propagation

`/* Meta-signals for aggregating the results in function of quantifiers */` ;
for *i=1 to n* **do**
  $\mathrm{R}^i_{\mathrm{Q}_i} \longleftarrow \mathcal{M}$.add.new_meta_signal_of_Speed(0) ;
  $\mathrm{L}^i_{\mathrm{Q}_i} \longleftarrow \mathcal{M}$.add.new_meta_signal_of_Speed(0) ;
end
`/* Boolean meta-signals for assignating the variables */` ;
foreach $\ell^\pi \in \mathcal{N}_\phi$ **do**
  if $\ell = x_i$ **then**
    $\overrightarrow{t^\pi} \longleftarrow \mathcal{M}$.add.new_meta_signal_of_Speed(1) ;
    $\overrightarrow{f^\pi} \longleftarrow \mathcal{M}$.add.new_meta_signal_of_Speed(1) ;
    $\overleftarrow{t^\pi} \longleftarrow \mathcal{M}$.add.new_meta_signal_of_Speed(-1) ;
    $\overleftarrow{f^\pi} \longleftarrow \mathcal{M}$.add.new_meta_signal_of_Speed(-1) ;
  end
end
`/* Rules of split and assignment of variables */` ;
for *i=1 to n* **do**
  foreach $\ell^\pi \in \mathcal{N}_\phi$ **do**
    if $\ell = x_i$ **then**
      $\mathcal{M} \longleftarrow$ add_rule($\{\overrightarrow{\ell^\pi}, \mathsf{x}_i\} \rightarrow \{\overleftarrow{f^\pi}, \mathsf{x}_i, \overrightarrow{t^\pi}\}$) ;
      $\mathcal{M} \longleftarrow$ add_rule($\{\overleftarrow{\ell^\pi}, \mathsf{x}_i\} \rightarrow \{\overleftarrow{f^\pi}, \mathsf{x}_i, \overrightarrow{t^\pi}\}$) ;
    else
      $\mathcal{M} \longleftarrow$ add_rule($\{\overrightarrow{\ell^\pi}, \mathsf{x}_i\} \rightarrow \{\overleftarrow{\ell^\pi}, \mathsf{x}_i, \overrightarrow{\ell^\pi}\}$);
      $\mathcal{M} \longleftarrow$ add_rule($\{\overleftarrow{\ell^\pi}, \mathsf{x}_i\} \rightarrow \{\overleftarrow{\ell^\pi}, \mathsf{x}_i, \overrightarrow{\ell^\pi}\}$);
    end
  end
  $\mathcal{M} \longleftarrow$ add_rule($\{\overrightarrow{\mathsf{collect}}, \mathsf{x}_i\} \rightarrow \{\overleftarrow{\mathsf{collect}}, \mathsf{x}_i, \overrightarrow{\mathsf{collect}}\}$) ;
  $\mathcal{M} \longleftarrow$ add_rule($\{\mathsf{x}_i, \overleftarrow{\mathsf{collect}}\} \rightarrow \{\overleftarrow{\mathsf{collect}}, \mathsf{x}_i, \overrightarrow{\mathsf{collect}}\}$) ;
  $\mathcal{M} \longleftarrow$ add_rule($\{\overrightarrow{\mathsf{store}}, \mathsf{x}_i\} \rightarrow \{\overleftarrow{\mathsf{store}}, \mathrm{L}^i_{\mathrm{Q}_i}, \overrightarrow{\mathsf{store}}\}$) ;
  $\mathcal{M} \longleftarrow$ add_rule($\{\mathsf{x}_i, \overleftarrow{\mathsf{store}}\} \rightarrow \{\overleftarrow{\mathsf{store}}, \mathrm{R}^i_{\mathrm{Q}_i}, \overrightarrow{\mathsf{store}}\}$) ;
end

Algorithm 3: Rules for the propagation

process involved in the generation of the formula: signals collide with respect to the lexical order of the set of nodes $\mathcal{N}_\phi$.

B.6 Collecting the results and answering Q-SAT

We give in Tab.6 the necessary collision rules for collecting the results of all evaluations. This is done with respect to the quantifiers of the Q-SAT formula, and so it depends on the formula. We use meta-signals $\mathrm{R}^i_{\mathrm{Q}_i}$ and $\mathrm{L}^i_{\mathrm{Q}_i}$ emitted during the propagation, where

$$
\begin{array}{lll}
\{\ \overrightarrow{\vee^{\mathsf{r}}},\ \overleftarrow{\mathsf{T}^{\mathsf{rl}}}\ \} \to \{\ \overrightarrow{t()^{\mathsf{r}}}\ \} & \{\ \overrightarrow{t()^{\mathsf{r}}},\ \overleftarrow{\mathsf{T}^{\mathsf{rr}}}\ \} \to \{\ \overrightarrow{\mathsf{t}^{\mathsf{r}}}\ \} & \{\ \overrightarrow{id^{\mathsf{r}}},\ \overleftarrow{\mathsf{T}^{\mathsf{rr}}}\ \} \to \{\ \overrightarrow{\mathsf{t}^{\mathsf{r}}}\ \} \\
\{\ \overrightarrow{\vee^{\mathsf{r}}},\ \overleftarrow{\mathsf{F}^{\mathsf{rl}}}\ \} \to \{\ \overrightarrow{id^{\mathsf{r}}}\ \} & \{\ \overrightarrow{t()^{\mathsf{r}}},\ \overleftarrow{\mathsf{F}^{\mathsf{rr}}}\ \} \to \{\ \overrightarrow{\mathsf{t}^{\mathsf{r}}}\ \} & \{\ \overrightarrow{id^{\mathsf{r}}},\ \overleftarrow{\mathsf{F}^{\mathsf{rr}}}\ \} \to \{\ \overrightarrow{\mathsf{f}^{\mathsf{r}}}\ \} \\
\{\ \overleftarrow{\vee^{\mathsf{r}}},\ \overrightarrow{\mathsf{T}^{\mathsf{rl}}}\ \} \to \{\ \overleftarrow{t()^{\mathsf{r}}}\ \} & \{\ \overleftarrow{t()^{\mathsf{r}}},\ \overrightarrow{\mathsf{T}^{\mathsf{rr}}}\ \} \to \{\ \overleftarrow{\mathsf{t}^{\mathsf{r}}}\ \} & \{\ \overleftarrow{id^{\mathsf{r}}},\ \overrightarrow{\mathsf{T}^{\mathsf{rr}}}\ \} \to \{\ \overleftarrow{\mathsf{t}^{\mathsf{r}}}\ \} \\
\{\ \overleftarrow{\vee^{\mathsf{r}}},\ \overrightarrow{\mathsf{F}^{\mathsf{rl}}}\ \} \to \{\ \overleftarrow{id^{\mathsf{r}}}\ \} & \{\ \overleftarrow{t()^{\mathsf{r}}},\ \overrightarrow{\mathsf{F}^{\mathsf{rr}}}\ \} \to \{\ \overleftarrow{\mathsf{t}^{\mathsf{r}}}\ \} & \{\ \overleftarrow{id^{\mathsf{r}}},\ \overrightarrow{\mathsf{F}^{\mathsf{rr}}}\ \} \to \{\ \overleftarrow{\mathsf{f}^{\mathsf{r}}}\ \}
\end{array}
$$

Table 5. Collision rules to evaluate the disjunction $\vee^r$.

Q_i is the quantification on x_i (see Algo. 3). Table 7 shows the explicit collecting rules for the variable x_3 of our running example. Since x_3 is linked by an universal quantifier, the collect for the stage corresponding to x_3 must perform a conjonction.

$$
\begin{array}{rl}
\{\ \overrightarrow{\mathsf{collect}},\ B\} \to \{\ \overleftarrow{B}\ \} & \{\ \overrightarrow{\mathsf{collect}}, \mathsf{x}_i\} \to \{\ \overleftarrow{\mathsf{collect}},\ \mathrm{L}^i_{\mathrm{Q}_i},\ \overrightarrow{\mathsf{collect}}\ \} \\
\{\ B,\ \overleftarrow{\mathsf{collect}}\ \} \to \{\ \overrightarrow{B}\ \} & \{\ \mathsf{x}_i,\ \overleftarrow{\mathsf{collect}}\ \} \to \{\ \overleftarrow{\mathsf{collect}},\ \mathrm{R}^i_{\mathrm{Q}_i},\ \overrightarrow{\mathsf{collect}}\ \} \\
\{\ \overrightarrow{B_1},\ \mathrm{R}_{\mathrm{Q}},\ \overleftarrow{B_2}\ \} \to \{\ \overrightarrow{B_3}\ \} & \text{for } B, B_1, B_2, B_3 \in \{\mathsf{T}, \mathsf{F}\},\ \mathrm{Q} \in \{\exists, \forall\} \text{ and} \\
\{\ \overrightarrow{B_1},\ \mathrm{L}_{\mathrm{Q}},\ \overleftarrow{B_2}\ \} \to \{\ \overleftarrow{B_3}\ \} & B_3 = B_1 \vee B_2 \text{ if } \mathrm{Q} = \exists,\ B_3 = B_1 \wedge B_2 \text{ if } \mathrm{Q} = \forall
\end{array}
$$

Table 6. Collision rules for the collect process.

$$
\begin{array}{ll}
\{\ \overrightarrow{\mathsf{T}},\ \mathrm{R}^3_\forall,\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overrightarrow{\mathsf{T}}\} & \{\ \overrightarrow{\mathsf{T}},\ \mathrm{L}^3_\forall,\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overleftarrow{\mathsf{T}}\} \\
\{\ \overrightarrow{\mathsf{T}},\ \mathrm{R}^3_\forall,\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overrightarrow{\mathsf{F}}\} & \{\ \overrightarrow{\mathsf{T}},\ \mathrm{L}^3_\forall,\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overleftarrow{\mathsf{F}}\} \\
\{\ \overrightarrow{\mathsf{F}},\ \mathrm{R}^3_\forall,\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overrightarrow{\mathsf{F}}\} & \{\ \overrightarrow{\mathsf{F}},\ \mathrm{L}^3_\forall,\ \overleftarrow{\mathsf{T}}\ \} \to \{\ \overleftarrow{\mathsf{F}}\} \\
\{\ \overrightarrow{\mathsf{F}},\ \mathrm{R}^3_\forall,\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overrightarrow{\mathsf{F}}\} & \{\ \overrightarrow{\mathsf{F}},\ \mathrm{L}^3_\forall,\ \overleftarrow{\mathsf{F}}\ \} \to \{\ \overleftarrow{\mathsf{F}}\} \\
\{\ \mathsf{x}_3,\ \overleftarrow{\mathsf{collect}}\ \} \to \{\ \overleftarrow{\mathsf{collect}},\ \mathrm{R}^3_\forall,\ \overrightarrow{\mathsf{collect}}\ \} & \{\ \overrightarrow{\mathsf{collect}}, \mathsf{x}_3\} \to \{\ \overleftarrow{\mathsf{collect}},\ \mathrm{L}^3_\forall,\ \overrightarrow{\mathsf{collect}}\ \}
\end{array}
$$

Table 7. Rules for the universal quantifier of the variable x_3.