A generic and modular signal machine solving satifiability problems[*]

Denys Duchier, Jérôme Durand-Lose, Maxime Senot

LIFO, University of Orléans, B.P. 6759, F-45067 Orléans Cedex 2, France
{denys.duchier,jerome.durand-lose,maxime.senot}@univ-orleans.fr

Abstract

Segments lines and their intersections in the Euclidean plane allow to realize non-trivial computations. Despite the simplicity of this abstract geometrical model of computation, it is possible to compute in the sense of Church-Turing, and even in an analog way. We present here a formalization of this idea into abstract devices: *signal machines*. To illustrate the power of these machines working on a continuous space-time, we present in this talk an efficient and geometrical solution to satifiability problems such as SAT or Q-SAT, by means of signals and their collisions. We also discuss complexities and propose a new measure for time complexity on signal machines: *the collision depth*, which is cubic for our proposed algorithm for Q-SAT.

Abstract Geometrical Computation. Signal machines, introduced in [Durand-Lose2003], take their origins in the world of cellular automata (see Fig.1). Indeed, signals and their interactions are very useful for studying problems and properties of cellular automata *e.g.* universality [Cook2004], synchronization [Mazoyer1996] or for implementing computations [Delorme and Mazoyer2002]. By abstracting the discrete nature of the cellular space-time to the continuity of the Euclidean plane, we can consider 1-dimensional and colored signals moving on the plane. Deterministic rules describe what happens when several signals meet, in function of their colors and speeds.

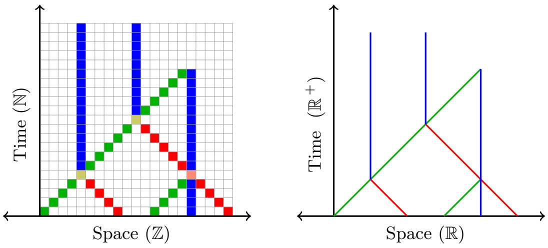


Figure 1: From cellular automata to signal machines.

Computations with colored segment lines in the Euclidean plane, their formalization into abstract devices—signal machines— and topological definitions of corresponding space-time diagrams constitute a larger class of models of computation, called *abstract geometrical computation* (*AGC*).

Other geometrical models of computation exist and allow to compute: Euclidean machines [Mycka and al.2006], colored universes [Jacopini and Sontacchi1990], geometric machines [Huckenbeck1989], piece-wise constant derivative systems [Bournez1997] . . .

Signal machines can simulate Turing machines, and are thus Turing-universal [Durand-Lose2005]. They are also capable of analog computation by using the continuity of space and time to simulate analog models such as BSS's one [Durand-Lose2008, Blum and al.1989] or computable analysis [Durand-Lose2009, Weihrauch2000].

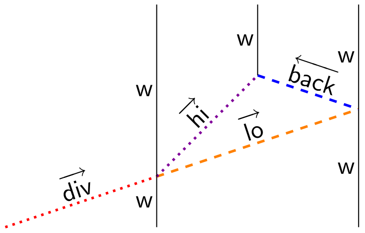
Signal machines. Each *signal* is an instance of a *meta-signal*, defining its type and its speed. When a set of signals collide, they are replaced by a new set of signals according to a matching *collision rule*. A rule has the form: $\{\sigma_1, \ldots, \sigma_n\} \to \{\sigma'_1, \ldots, \sigma'_p\}$ where all σ_i are meta-signals of distinct speeds as well as σ'_j. A *signal machine* is defined by a finite set of meta-signals and a set of collision rules. A signal machine is runned started from an initial configuration, *i.e.* a finite number of signals placed on the real line. Its evolution is representated geometrically as a *space-time diagram*, where space is always represented horizontally, and time vertically, growing upwards.

The geometrical algorithm displayed in Fig. 2 computes the middle: the new w is located exactly half way between the initial two w.

Computing in the fractal cloud. In this talk, to illustrate some abilities of signal machines, we present a geometrical algorithm solving Q-SAT—the problem of quantified boolean formula satisfiablity—which also provides a structure for solving other variants of boolean satifiablity.

[*] Details of this work can be found at http://arxiv.org/abs/1105.3454 ([Duchier and al.2011a]).

Meta-Signals	**Speed**
w	0
div	3
hi	1
lo	3
back	-3



Collision rules

{ w, div } → { w, hi, lo }
{ lo, w } → { back, w }
{ hi, back } → { w }

Initial configuration

div @ $s < 0$
w @ 0
w @ $x > 0$

Figure 2: Geometrical algorithm for computing the middle.

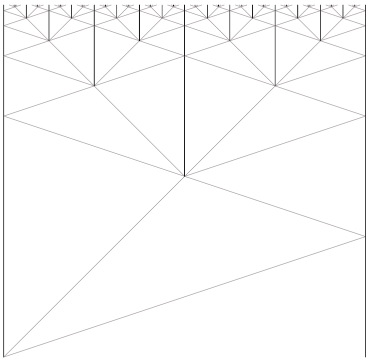


Figure 3: The fractal tree.

This construction is based on the previous algorithm computing the middle, which is used recursively to build the infinite binary tree of Fig. 3, the *fractal cloud*. This tree is then interpreted as a binary search tree, where the number of boolean variables determines the number of needed levels, and all possible boolean assignments can be tested in a massive parallel way. A computation in the fractal cloud is based on the *Map/Reduce* principle and follows three steps: distributing the computation along the tree, evaluating each case at the top level, and collecting all the results to yield the final answer.

Solving Q-SAT with a single machine. A Q-SAT formula is coded by signals and is propagated along the tree. Variables are represented by a set of signals so that at the i^{th} level, the variable x_i splits into two signals: a true signal going right and a false signal.

At the final stage, in each box (corresponding to each possible assignment of variables), there is no more variable but only true and false signals and signals coding boolean connectives. The unquantified formula is then evaluated in each case, and the results are aggregated with respect to the quantifiers of the input formula. The final answer—either the quantified formula is true or false—is given by the last signal going left at the top of the whole construction, which can be seen in Fig. 4.

We already provided geometrical algorithms in [Duchier and al.2010] and in [Duchier and al.2011b] to solve respectively SAT and Q-SAT, but in both cases, the machines were dependent on the input formula: a signal machine was generated in polynomial time for each boolean formula.

We present here a *single* signal machine solving Q-SAT for any instance coded in the initial configuration. We also provide a structure (the fractal cloud) and a *modular* approach (the tree, the propagation, the evaluation... can be programmed independently) permitting to solve easily other satisfiablity problems such as #SAT or MAX-SAT.

As all these constructions, bounded by the infinite binary tree, are made in constant width and time independently of the size of the formula, space and time are no longer appropriate complexity measures. We define a new measure for time complexity, better suited to the massive parallelism of signal machines: the *collision depth*. It is defined as the maximal number of consecutive collisions when we follow an ascending path through the whole diagram. Whereas the instance-specific constructions were in quadratic collision depth, the generic one is in cubic collision depth. This gives us an idea of the cost of genericity.

References

[Blum and al.1989] Blum, L., Shub, M., and Smale, S. (1989). On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the AMS*, 21(1):1–46.

[Bournez1997] Bournez, O. (1997). Some bounds on the computational power of piecewise constant derivative systems. In *ICALP '97*, number 1256 in LNCS, pages 143–153.

[Cook2004] Cook, M. (2004). Universality in elementary cellular automata. *Complex Systems*, 15(1):1–40.

[Delorme and Mazoyer2002] Delorme, M. and Mazoyer, J. (2002). Signals on cellular automata. In Adamatzky, A., editor, *Collision-based Computing*, pages 231–275. Springer-Verlag.

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

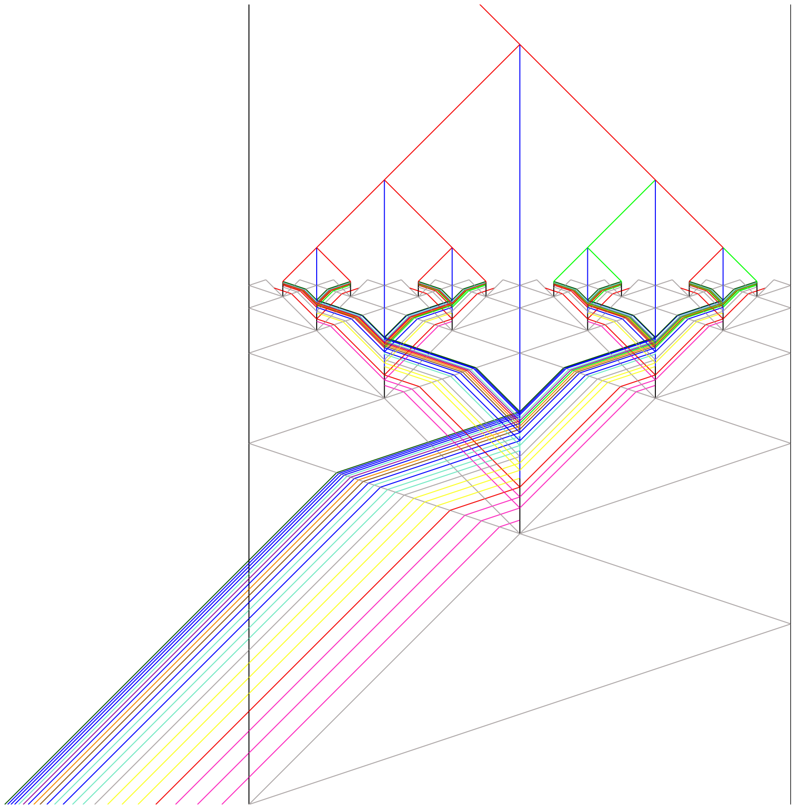


Figure 4: The whole diagram.

[Duchier and al.2011a] Duchier, D., Durand-Lose, J., and Senot, M. (2011). Computing in the fractal cloud: modular generic solvers for SAT and Q-SAT variants. *Arxiv preprint arXiv:1105.3454*. Available at `http://arxiv.org/abs/1105.3454`.

[Duchier and al.2011b] Duchier, D., Durand-Lose, J., and Senot, M. (2011). Solving Q-SAT in bounded space and time by geometrical computation. In *7th Int. Conf. Computability in Europe (CiE '11)*. To appear in the local booklet (abstracts and extended abstracts of unpublished papers).

[Durand-Lose2003] Durand-Lose, J. (2003). *Calculer géométriquement sur le plan – machines à signaux*. Habilitation à Diriger des Recherches, École Doctorale STIC, Université de Nice-Sophia Antipolis. In French.

[Durand-Lose2005] Durand-Lose, J. (2005). Abstract geometrical computation: Turing computing ability and undecidability. In Cooper, B., Löwe, B., and Torenvliet, L., editors, *New Computational Paradigms, 1st Conf. Computability in Europe (CiE '05)*, number 3526 in LNCS, pages 106–116. Springer.

[Durand-Lose2008] Durand-Lose, J. (2008). Abstract geometrical computation with accumulations: Beyond the Blum, Shub and Smale model. In Beckmann, A., Dimitracopoulos, C., and Löwe, B., editors, *4th Conf. Computability in Europe (CiE '08) (extended abstracts of unpublished papers)*, pages 107–116. University of Athens.

[Durand-Lose2009] Durand-Lose, J. (2009). Abstract geometrical computation and computable analysis. In Costa, J. and Dershowitz, N., editors, *8th Int. Conf. on Unconventional Computation 2009 (UC '09)*, number 5715 in LNCS, pages 158–167. Springer.

[Huckenbeck1989] Huckenbeck, U. (1989). Euclidian geometry in terms of automata theory. *Theoret. Comp. Sci.*, 68(1):71–87.

[Jacopini and Sontacchi1990] Jacopini, G. and Sontacchi, G. (1990). Reversible parallel computation: an evolving space-model. *Theoret. Comp. Sci.*, 73(1):1–46.

[Mazoyer1996] Mazoyer, J. (1996). On optimal solutions to the firing squad synchronization problem. *Theoret. Comput. Sci.*, 168(2):367–404.

[Mycka and al.2006] Mycka, J., Coelho, F., and Costa, J. F. (2006). The Euclid abstract machine: Trisection of the angle and the halting problem. In Calude, C. S., Dinneen, M. J., Paun, G., Rozenberg, G., and Stepney, S., editors, *5th Int. Conf. on Unconventional Computation(UC 06)*, volume 4135 of *LNCS*, pages 195–206. Springer.

[Weihrauch2000] Weihrauch, K (2000). Computable Analysis: an Introduction. Springer Verlag.