Reversible conservative rational abstract geometrical computation is Turing-universal

Jérôme Durand-Lose

Reversible conservative rational abstract geometrical computation is Turing-universal
(extended abstract)

Jérôme Durand-Lose*

Laboratoire d'Informatique Fondamentale d'Orléans, Université d'Orléans,
B.P. 6759, F-45067 ORLÉANS Cedex 2.

Abstract. In *Abstract geometrical computation for black hole computation (MCU '04, LNCS 3354)*, the author provides a setting based on rational numbers, abstract geometrical computation, with super-Turing capability. In the present paper, we prove the Turing computing capability of reversible conservative abstract geometrical computation. Reversibility allows backtracking as well as saving energy; it corresponds here to the local reversibility of collisions. Conservativeness corresponds to the preservation of another energy measure ensuring that the number of signals remains bounded. We first consider 2-counter automata enhanced with a stack to keep track of the computation. Then we built a simulation by reversible conservative rational signal machines.

Key-words. Abstract geometrical computation, Conservativeness, Rational numbers, Reversibility, Turing-computability, 2-counter automata.

1 Introduction

Reversible computing is a very important issue because it allows on the one hand to backtrack a phenomenon to its source and on the other hand to save energy. Let us note that quantum computing relies on reversible operations. There are general studies on reversible computation [LTV98] as well as many model dependent results: on Turing machines [Lec63,Ben73,Ben88] and 2-counter machines [Mor96] (we do not use these), on logic gates [FT82,Tof80], and last but not least, on reversible Cellular automata on both finite configurations [Mor92,Mor95,Dub95] and infinite ones [DL95,DL97,DL02,Kar96,Tof77] as well as decidability results [Čul87,Kar90,Kar94] and a survey [TM90].

Abstract geometrical computation (AGC) [DL05a,DL05b] comes from the common use in the literature on *cellular automata* (CA) of Euclidean lines to model discrete lines in space-time diagrams of CA (*i.e.* colorings of $\mathbb{Z} \times \mathbb{N}$ with states as on the left of Fig. 1) to access dynamics or to design. The main characteristics of CA, as well as abstract geometrical computation, are: *parallelism, synchrony, uniformity* and *locality* of updating. Discrete lines are often observed and idealized as on Fig. 1. They can be the keys to understanding the dynamics

* `http://www.univ-orleans.fr/lifo/Members/Jerome.Durand-Lose`, `Jerome.Durand-Lose@univ-orleans.fr`

like in [Ila01, pp. 87–94] or [BNR91,JSS02]. They can also be the tool to design CA for precise purposes like Turing machine simulation [LN90]. These discrete line systems have also been studied on their own [MT99,DM02].

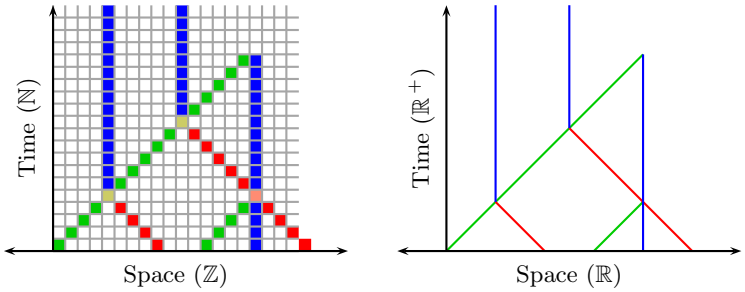


Fig. 1. Space-time diagram of a cellular automaton and its signal machine counterpart.

Abstract geometrical computation considers Euclidean lines. The support of space and time is $\mathbb{R}$. Computations are produced by *signal machines* which are defined by finite sets of *meta-signals* and of *collision rules*. Signals are atomic information that correspond to meta-signals and move at constant speed thus generating Euclidean line segments on space-time diagrams. Collision rules are pairs *(incoming meta-signals, outgoing meta-signals)* that define a mapping over sets of meta-signals. They define what happens when signals meet. A configuration is a mapping from $\mathbb{R}$ to meta-signals, collision rules and two special values: void (*i.e.* nothing there) and accumulations (amounting for black holes). The time scale being $\mathbb{R}^+$, there is no such thing as a "next configuration". The following configurations are defined by the uniform movement of signals. In the configurations following a collision, incoming signals are replaced by outgoing signals according to a collision rule.

Zeno like acceleration and accumulation can be brought out as on Fig. 2 of Sect. 2. Accumulations can lead to an unbounded burst of signals producing infinitely many signals in finite time (as in the right of Fig. 2). To avoid this, a *conservativeness* condition is imposed: a positive energy is defined for every meta-signal, the sum of the energies must be preserved by each rule. Thus no energy creation is possible; the number of signals is bounded.

To our knowledge, AGC is the only computing model that is a dynamical system with continuous time and space but finitely many local values. The closest model we know of is the Mondrian automata of Jacopini and Sontacchi [JS90] which is also reversible. Their space-time diagrams are mappings from $\mathbb{R}^n$ to a finite set of colors representing bounded finite polyhedra. Another close model is the piecewise-constant derivative system [AM95,Bou99]: $\mathbb{R}^n$ is partitioned into finitely many polygonal regions, trajectories are defined by a constant derivative on each region and form sequences of (Euclidean) line segments.

In this paper, space and time are restricted to rational numbers. This is possible since all the operations used preserve rationality. All quantifiers and intervals are over $\mathbb{Q}$, not $\mathbb{R}$.

The Turing computing capability of conservative signal machines is proved in [DL05a,DL06] by simulating any 2-counter automaton since Turing machines and 2-counter automata compute exactly the same functions. To build a reversible version of the simulation, we have to cope with the inherent irreversibility of counter automaton (the result of Morita [Mor96] is not used here because it does not fit well with our approach): branching (there is no way to guess the previous instruction) and subtracting (0 comes from both 0 and 1).

To cope with this we add stacks to store information for reversibility. A stack over an alphabet $\{1, 2, \ldots, l\}$ is encoded by a rational number σ such that: $\frac{1}{l+2}$ encodes the empty stack, otherwise $\frac{1}{l+1}<\sigma<1$. After pushing a value v on a stack σ the new stack is $\frac{v+\sigma}{l+1}$. The top of the stack is $\lfloor(l+1)\sigma\rfloor$ and $(l+1)\sigma-\lfloor(l+1)\sigma\rfloor$ encodes the rest of the stack. This more or less corresponds to a base $l+1$ decimal number manipulation as can be found in *e.g.* [Bou99].

This simple memory scheme can be implemented inside a reversible conservative rational signal machine. We then finish the simulation by adapting the construction from [DL05a,DL06] by adding the storing of current line number before passing to the next and of the previous value of a counter whenever it reaches 0.

The definition of signal machines can be found in Sect. 2. Section 3 deals with 2-counter automata and their enhancement with stacks. Section 4 shows how the stacks are implemented. In Sect. 5, the different pieces are gathered in order to achieve the simulation. Section 6 gives a short conclusion.

2 Abstract geometrical computations

Abstract geometrical computations are defined by the following machines:

Definition 1 A *rational signal machine* is defined by (M, S, R) where M (*meta-signals*) is a finite set, S (*speeds*) a mapping from M to $\mathbb{Q}$ and R (*collision rules*) a partial mapping from the subsets of M of cardinality at least 2 into the subsets of M (speeds must differ in both domain and range).

Each instance of a meta-signal is a *signal*. The mapping S assigns rational *speeds* to meta-signals. They correspond the slopes of the segments in space-time diagrams. The *collision rules*, denoted $\rho^- \to \rho^+$, define what happens when two or more signals meet.

The *extended value set*, V, is the union of M and R plus two symbols: one for void, $\oslash$, and one for an accumulation (or black hole) ❋. A *configuration*, c, is a total mapping from $\mathbb{Q}$ to V such that the set $\{\, x \in \mathbb{Q} \mid c(x) \neq \oslash \,\}$ is finite.

A signal corresponding to a meta-signal μ at a position x, *i.e.* $c(x) = \mu$, is moving uniformly with constant speed $S(\mu)$. A signal must start (resp. end) in the initial (resp. final) configuration or in a collision. These correspond to condition 2 in Def. 2. At a $\rho^- \to \rho^+$ collision signals corresponding to the meta-signals in ρ^- (resp. ρ^+) must end (resp. start); no other signal should be present (condition 3). A black hole corresponds to an accumulation of collisions and disappears without a trace (condition 4).

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

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

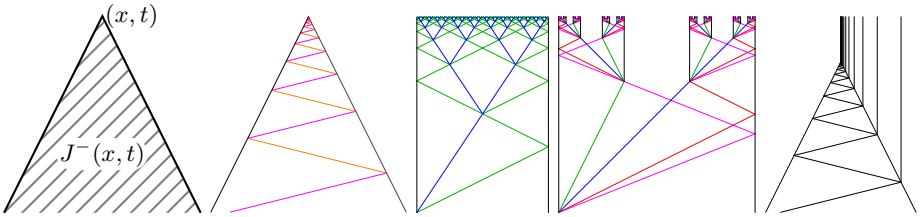


Fig. 2. Light-cone, a simple accumulation and three unwanted phenomena.

Definition 2 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 $\mathbb{Q}\times[0,T]$ to V) such that, $\forall(x,t)\in\mathbb{Q}\times[0,T]$:

1. $\forall t{\in}[0,T]$, $\{\, x\in\mathbb{Q} \mid c_t(x)\neq\oslash\,\}$ is finite,
2. if $c_t(x){=}\mu$ then $\exists t_i,t_f{\in}[0,T]$ with $t_i{<}t{<}t_f$ or $0{=}t_i{=}t{<}t_f$ or $t_i{<}t{=}t_f{=}T$ s.t.:
   - $\forall t'\in(t_i,t_f)$, $c_{t'}(x+S(\mu)(t'-t))=\mu$,
   - $t_i=0$ or $c_{t_i}(x_i)\in R$ and $\mu\in(c_{t_i}(x_i))^+$ where $x_i=x+S(\mu)(t_i-t)$,
   - $t_f=T$ or $c_{t_f}(x_f)\in R$ and $\mu\in(c_{t_f}(x_f))^-$ where $x_f=x+S(\mu)(t_f-t)$;
3. if $c_t(x){=}\rho^-{\to}\rho^+\in R$ then $\exists\varepsilon$, $0{<}\varepsilon$, $\forall t'{\in}[t{-}\varepsilon,t{+}\varepsilon]\cap[0,T]$, $\forall x'{\in}[x-\varepsilon,x+\varepsilon]$,
   - $c_{t'}(x')\in\rho^-{\cup}\rho^+\cup\{\oslash\}$,
   - $\forall\mu{\in}M$, $c_{t'}(x'){=}\mu \Rightarrow \bigvee \begin{cases}\mu\in\rho^- \text{ and } t'<t \text{ and } x'=x+S(\mu)(t'-t))\,,\\ \mu\in\rho^+ \text{ and } t<t' \text{ and } x'=x+S(\mu)(t'-t))\,;\end{cases}$
4. if $c_t(x)=$ ❋ then
   - $\exists\varepsilon>0$, $\forall(x',t')\notin J^-(x,t)$, ($|x{-}x'|{<}\varepsilon$ and $|t{-}t'|{<}\varepsilon$) $\Rightarrow c_{t'}(x)=\oslash$,
   - $\forall\varepsilon>0$, $\{\ (x',t')\in J^-(x,t) \mid t{-}\varepsilon{<}t'{<}t\wedge c_{t'}(x')\in R\ \}$ is infinite.

In the illustrations of space-time diagrams, time increases upwards.

2.1 Conservativeness

The three space-time diagrams on the right of Fig. 2 provide examples uncompatible with Def. 2 at the time of accumulation. In each case, the number of signals is bursting to infinity and black holes are not isolated. To prevent this, the following restriction is imposed.

Definition 3 A signal machine is *conservative* iff there exists an energy from meta-signals to positive integers, $E : M \to \mathbb{N}^*$, such that the total energy of the system, *i.e.* the sum of the energies of all present signals, is constant.

One can check easily that the total energy is constant iff for each rule the sum of the energies of incoming meta-signals and the sum of outgoing ones are equal. It follows automatically that given a conservative signal machine and an initial configuration, the number of signals in any following configuration, as well as the number of accumulations, is bounded (by the total energy divided by the least atomic energy). A simple sub-case of conservativeness is when all the meta-signals have the same energy and the numbers of in and out meta-signals are always equal.

2.2 Reversibility

A dynamical systems is said to be *reversible* when from any configuration it is possible to generate all the previous configurations. Moreover, the inverse dynamical system should be of the same nature.

Concerning signal machines, the "inversion" of an isolated signal is the same signal with opposite speed. Regarding collision, one has to guess its position and the in-coming signals from the out-going ones. Collisions resulting in nothing are impossible to guess going backward, and they cannot be conservative. Collisions resulting in only one signal are also impossible to predict. Reversibility holds if and only if the (partial mapping) M is one-to-one and always yields 2 or more out-going meta-signals. The inverse signal machine is the same with the rules reversed.

Definition 4 A signal machine is *reversible* if and only if R is one-to-one and maps only on sets of cardinality at least 2.

Let us point out this is true as long as there is no accumulation! The way an accumulation disappears is like a (second order) collision resulting in nothing. Moreover if its location could be guessed, there are infinitely many way to scale it since there is no absolute scale for space nor time.

3 2-counter automaton with stack

A *2-counter automaton* is a finite automaton coupled with two counters, A and B. The possible actions on any counter are *add/subtract* 1 and *branch if non-zero*. Such an automaton can be described with a six-operations assembly language with branching labels as on the left part of Fig. 6 (see [Min67] for more on 2-counter automata). The configuration of a 2-counter automaton is defined by (n, a, b) (the line number and the values of the counters).

Two-counter automaton are intrinsically irreversible: subtracting 1 yields 0 for both values 0 and 1 and before a labeled instruction the instruction can be the one on the previous line or a branching to this label.

To achieve reversibility two stacks are added: one, Σ_i, records the instruction number (*i.e.* it records the values of the instruction counter) and another one, Σ_z, record the previous value (0 or 1) of any counter that holds zero after a subtraction. We write $x.\Sigma$ to indicate pushing x on S or that x is the top of the stack. The dynamics is described on Fig. 3 for instructions on A. The ones for

B are similar. Discarding the stacks, one gets the usual dynamics. The inverse dynamics is automatic (as long as the sequence was generated legally, otherwise things like undoing adding 1 from a zero counter might happen).

Instruction at line n	**Associated action**
`A ++`	$(n, \quad a, b, \Sigma_i, \Sigma_z) \vdash (n+1, a+1, b, n.\Sigma_i, \quad \Sigma_z)$
`A --`	$(n, a+2, b, \Sigma_i, \Sigma_z) \vdash (n+1, a+1, b, n.\Sigma_i, \quad \Sigma_z)$
`A --`	$(n, \quad 1, b, \Sigma_i, \Sigma_z) \vdash (n+1, \quad 0, b, n.\Sigma_i, 1.\Sigma_z)$
`A --`	$(n, \quad 0, b, \Sigma_i, \Sigma_z) \vdash (n+1, \quad 0, b, n.\Sigma_i, 0.\Sigma_z)$
`IF A !=0` m	$(n, \quad 0, b, \Sigma_i, \Sigma_z) \vdash (n+1, \quad 0, b, n.\Sigma_i, \quad \Sigma_z)$
`IF A !=0` m	$(n, a+1, b, \Sigma_i, \Sigma_z) \vdash (\quad m, a+1, b, n.\Sigma_i, \quad \Sigma_z)$

Fig. 3. Dynamics of a 2-counter automata with memory stacks.

4 Stack implementation

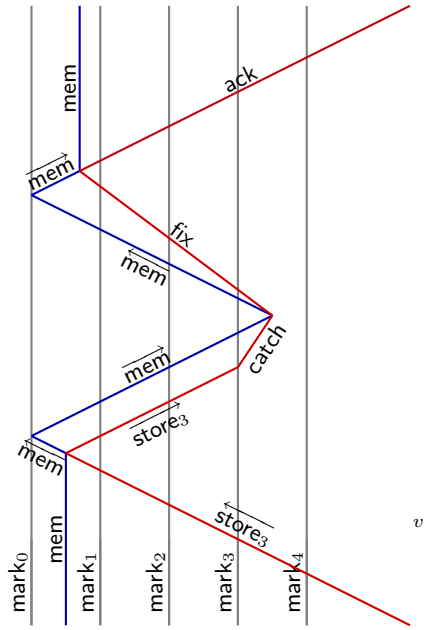
Since we are dealing with rational numbers, it is very easy to implement an unbounded stack of natural numbers 1 to l in the following way: $\frac{1}{l+2}$ encodes the empty stack. Let σ be a rational number encoding of a stack ($0<\sigma<1$). After pushing a value v on top of a stack σ, the new stack is $\frac{v+\sigma}{l+1}$. This ensures that, as soon as the stack is not empty, $\frac{1}{l+1}<\sigma<1$ and distinct from $\frac{i}{l+1}$, $i \in \{1, 2, \ldots, l\}$. The top of the stack $\lfloor (l+1)\sigma \rfloor$ and rest of the stack is $(l+1)\sigma - \lfloor (l+1)\sigma \rfloor$.

To implement this in a signal machine, a scale is defined (because the space is continuous and scaleless) and then the push operation is implemented. We are not interested in the pop and test of emptiness because our 2-counter simulation only pushes values. The pop corresponds to the inverse of push and is thus implicitly built.

The rational σ is encoded with a zero-speed signal mem. The scale is defined with zero-speed signals mark_0, mark_1,..., mark_l. They are regularly positioned, never move and defined positions 0, 1, ..., l. Thus the normal position of mem is between mark_0 and mark_1. To push v, mem is translated by v (lower part of Fig. 4). Then this position is scaled by $\frac{1}{l+1}$ (upper part of Fig. 4) to get $\frac{\sigma+v}{l+1}$. The process starts at the arrival of $\overleftarrow{\mathsf{store}_v}$.

Translating is very easy: $\overrightarrow{\mathsf{mem}}$ and $\overrightarrow{\mathsf{store}_v}$ are parallel. Their distance encodes σ. Their movement stops when the first one, $\overrightarrow{\mathsf{store}_v}$, reaches mark_v. The signal catch is then issued to stop $\overrightarrow{\mathsf{mem}}$ as in the middle of Fig. 4. This collision is distance v away from the original position of mem. This is ensured by the definition of speeds (we leave to the reader to verify this linear equation system based on the speeds given in Fig. 4). When this point is reached, a scaling remains to be done. Scaling by $\frac{1}{l+1}$, to go from $\sigma+v$ to $\frac{\sigma+v}{l+1}$, fix has to travel $(\sigma+v)-\frac{\sigma+v}{l+1} = (\sigma+v)\frac{l}{l+1}$ units, and $\overleftarrow{\mathsf{mem}}$ and $\overrightarrow{\mathsf{mem}}$ have to travel $(\sigma+v)+\frac{\sigma+v}{l+1} = (\sigma+v)\frac{l+2}{l+1}$ units. Thus if the speeds of $\overleftarrow{\mathsf{mem}}$ and $\overrightarrow{\mathsf{mem}}$ have the same absolute value then the speed of fix must be $\frac{l}{l+2}$ times the one of $\overleftarrow{\mathsf{mem}}$ (notice in Fig. 4 that $-2 = -3\frac{4}{4+2}$).

If the stack is empty, then backward collision between $\overleftarrow{\mathsf{mem}}$ and fix happens between mark_0 and mark_1. So that there is a (backward) collision between $\overrightarrow{\mathsf{mem}}$



Meta-signal	Speed
mark_i	0
$\overleftarrow{\mathsf{mem}}$	−3
mem	0
$\overrightarrow{\mathsf{mem}}$	3
$\overleftarrow{\mathsf{store}_i}$	−3
$\overrightarrow{\mathsf{store}_i}$	3
catch	1
fix	−2
ack	3

Rules

$$\{\mathsf{mem}, \overleftarrow{\mathsf{store}_v}\} \to \{\overleftarrow{\mathsf{mem}}, \overrightarrow{\mathsf{store}_v}\}$$
$$\{\mathsf{mark}_0, \overleftarrow{\mathsf{mem}}\} \to \{\mathsf{mark}_0, \overrightarrow{\mathsf{mem}}\}$$
$$\{\overrightarrow{\mathsf{store}_v}, \mathsf{mark}_v\} \to \{\mathsf{mark}_v, \mathsf{catch}\}$$
$$\{\overrightarrow{\mathsf{mem}}, \mathsf{catch}\} \to \{\overleftarrow{\mathsf{mem}}, \mathsf{fix}\}$$
$$\{\overrightarrow{\mathsf{mem}}, \mathsf{fix}\} \to \{\mathsf{mem}, \mathsf{ack}\}$$
$$\{\mathsf{mark}_i, \overleftarrow{\mathsf{store}_v}\} \to \{\overleftarrow{\mathsf{store}_v}, \mathsf{mark}_i\}$$
$$v < i,\ \{\overrightarrow{\mathsf{store}_v}, \mathsf{mark}_i\} \to \{\mathsf{mark}_i, \overrightarrow{\mathsf{store}_v}\}$$
$$\{\overrightarrow{\mathsf{mem}}, \mathsf{mark}_i\} \to \{\mathsf{mark}_i, \overrightarrow{\mathsf{mem}}\}$$
$$\{\mathsf{mark}_i, \overleftarrow{\mathsf{mem}}\} \to \{\overleftarrow{\mathsf{mem}}, \mathsf{mark}_i\}$$
$$\{\mathsf{mark}_i, \mathsf{fix}\} \to \{\mathsf{fix}, \mathsf{mark}_i\}$$
$$\{\mathsf{ack}, \mathsf{mark}_i\} \to \{\mathsf{mark}_i, \mathsf{ack}\}$$

Fig. 4. Implementation the stack for $l = 4$ and push(3).

(regenerated from mark_0 and $\overleftarrow{\mathsf{mem}}$) and catch. There no such a rule, it can be defined to have, for example, mem fixed and catch exiting on the left.

It is easy to check that the rules are invertible.

Let us note that, with slight modifications, $\overleftarrow{\mathsf{store}_v}$ and ack could come from/be sent left or right. It is also possible to carry extra information (like a next line number) through the storing process. This is done by having a special set of $\overleftarrow{\mathsf{store}_i}$, $\overrightarrow{\mathsf{store}_i}$, catch, fix, and ack for each possible piece of information.

5 Reversible computation

The idea is to use the construction provided in [DL05a] to simulate any 2-counter automaton with a conserving signal machine (which cannot be detailed here). Figure 5 shows how the counters are encoded using two fixed signals zero and one as a scale. A signal amounting for the current line zigzags between these signals. Figure 6 presents the code of a simple 2-counter automaton and some simulations. When a simulation stops, a signal stop appears and bounces between zero and one. In the reversible version, it exits the configuration on the right.

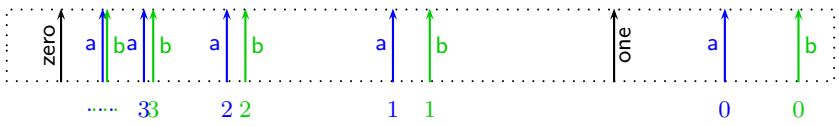


Fig. 5. Encoding positions of counters.

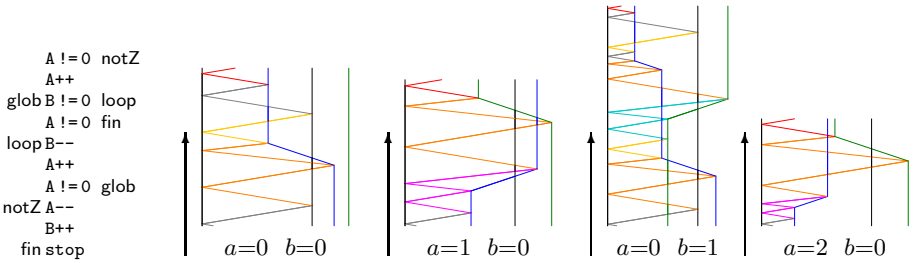


Fig. 6. A 2-counter automaton and its simulations for three different initial values.

Figure 7 show how everything is interconnected. Before going to the next configuration, the number of the just carried out instruction is stored on the left while recording the next instruction number. Each time there is a subtract 1 generating a 0, the previous value of the counter is stored on the right (cases $(1, 0)$ and $(0, 1)$) whatever counter is concerned.

6 Conclusion

As far as there is no accumulation, reversible conservative rational abstract geometrical computation has exactly the same computing capability as Turing machines (because rational numbers can be implemented exactly).

Two-counter automaton are exponentially slower than Turing machines. This is not important since we are only interested in computational issues. Nevertheless, it is easy to simulate reversibly (but without conservativeness) Turing machine in such a way that the number of collisions is proportional to the number of TM iterations. It would be interesting to do so to study the complexity.

It is possible to apply the iterated shrinking construction of [DL05a,DL06] which preserves reversibility so that the black hole models can now be embedded in a reversible setting. We do not do it here for lack of room on one side and on the other side this would provoke an accumulation which is clearly not a reversible phenomena.

References

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

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

[Ben73] C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Dev.*, 6:525–532, 1973.

[Ben88] C. H. Bennett. Notes on the history of reversible computation. *IBM J. Res. Dev.*, 32(1):16–23, 1988.

[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.

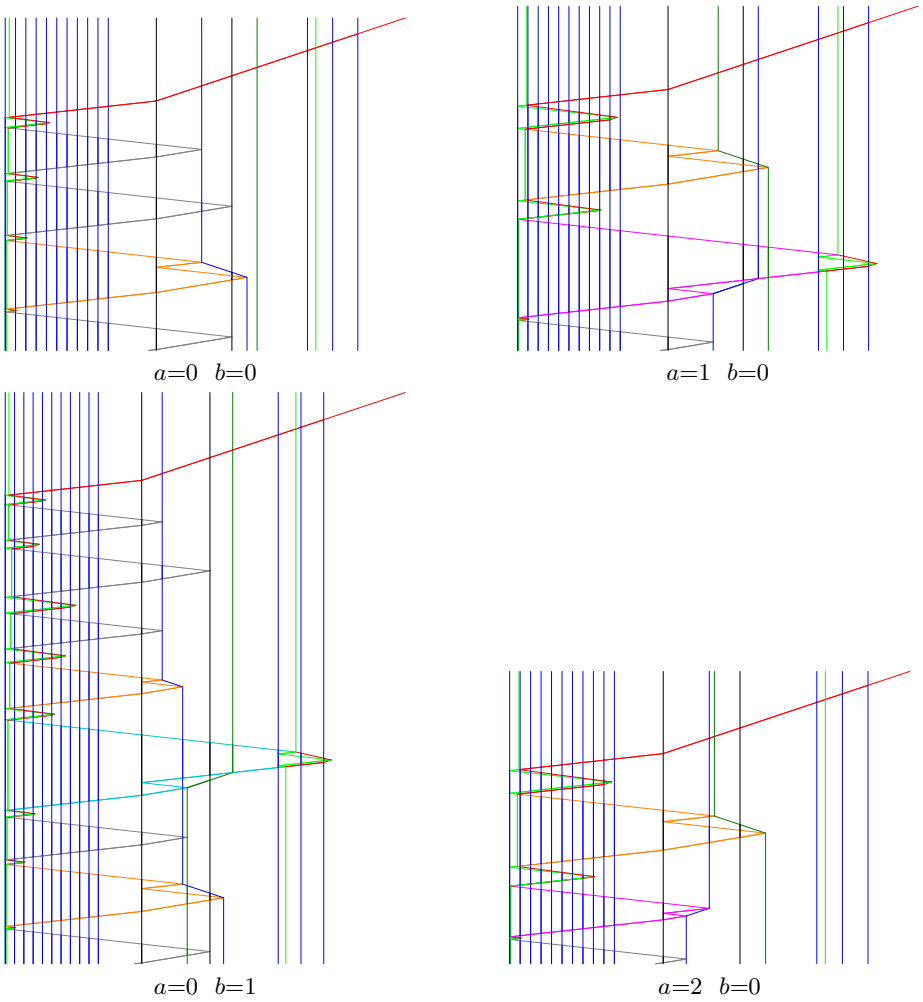


Fig. 7. The reversible simulations for same automaton and values.

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

[Čul87] K. Čulik II. On invertible cellular automata. *Complex Systems*, 1:1035–1044, 1987.

[DL95] J. Durand-Lose. Reversible cellular automaton able to simulate any other reversible one using partitioning automata. In *LATIN '95*, number 911 in LNCS, pp. 230–244. Springer, 1995.

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

[DL02] J. Durand-Lose. Computing inside the billiard ball model. In A. Adamatzky, ed., *Collision-based computing*, pp. 135–160. Springer, 2002.

[DL05a] J. Durand-Lose. Abstract geometrical computation for black hole computation (extended abstract). In M. Margenstern, ed., *Machines, Computations, and Universality (MCU '04)*, number 3354 in LNCS, pp. 176–187. Springer, 2005.

[DL05b] J. Durand-Lose. Abstract geometrical computation: Turing-computing ability and undecidability. In B. S. Cooper, B. Löwe, and L. Torenvliet, eds, *New Computational Paradigms, 1st Conference on Computability in Europe (CiE '05)*, number 3526 in LNCS, pp. 106–116. Springer, 2005.

[DL06] J. Durand-Lose. Abstract geometrical computation 1: embedding black hole computations with rational numbers. *Fundamenta Informaticae*, 74(4):491–510, 2006.

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

[Dub95] J.-C. Dubacq. How to simulate Turing machines by invertible 1d cellular automata. *International Journal of Foundations of Computer Science*, 6(4):395–402, 1995.

[FT82] E. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21, 3/4:219–253, 1982.

[Ila01] A. Ilachinski. *Cellular automata –a discrete universe*. World Scientific, 2001.

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

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

[Kar90] J. Kari. Reversibility of 2D cellular automata is undecidable. *Phys. D*, 45:379–385, 1990.

[Kar94] J. Kari. Reversibility and surjectivity problems of cellular automata. *J. Comput. System Sci.*, 48(1):149–182, 1994.

[Kar96] J. Kari. Representation of reversible cellular automata with block permutations. *Math. System Theory*, 29:47–61, 1996.

[Lec63] Y. Lecerf. Machines de Turing réversibles. Récursive insolubilité en $n \in \mathbb{N}$ de l'équation $u = \theta^n u$, où θ est un isomorphisme de codes. *Comptes rendus des séances de l'académie des sciences*, 257:2597–2600, 1963.

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

[LTV98] M. Li, J. Tromp, and P. Vitanyi. Reversible simulation of irreversible computation. *Physica D*, 120:168–176, 1998.

[Min67] M. Minsky. *Finite and infinite machines*. Prentice Hall, 1967.

[Mor92] K. Morita. Computation-universality of one-dimensional one-way reversible cellular automata. *Inform. Process. Lett.*, 42:325–329, 1992.

[Mor95] K. Morita. Reversible simulation of one-dimensional irreversible cellular automata. *Theoret. Comp. Sci.*, 148:157–163, 1995.

[Mor96] K. Morita. Universality of a reversible two-counter machine. *Theoret. Comp. Sci.*, 168(2):303–320, 1996.

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

[TM90] T. Toffoli and N. Margolus. Invertible cellular automata: a review. *Phys. D*, 45:229–253, 1990.

[Tof77] T. Toffoli. Computation and construction universality of reversible cellular automata. *J. Comput. System Sci.*, 15:213–231, 1977.

[Tof80] T. Toffoli. Reversible computing. In *ICALP*, volume 85 of *LNCS*, pp. 632–644. Springer, 1980.