

# Abstract geometrical computation 4: small Turing universal signal machines

Jérôme Durand-Lose

*LIFO, Université d'Orléans, B.P. 6759, F-45067 ORLÉANS Cedex 2, France.*

---

## Abstract

This article provides several very small signal machines able to perform any computation—in the classical understanding—generated from Turing machines, cellular automata and cyclic tag systems. A halting universal signal machine with 13 meta-signals and 21 collision rules is presented (resp. 15 and 24 for a robust version). If infinitely many signals are allowed to be present in the initial configuration, 5 meta-signals and 7 collision rules are enough to achieve non-halting weak universality (resp. 6 and 9 for a robust version).

*Key words:* Abstract geometrical computation; cyclic tag systems; Turing universality; signal machines; small universal machines.

---

## 1 Introduction

Computation and universality have been introduced in the 1930s. In the last five decades, it has been unveiled how common they are. The question moved from proving the universality of dynamical systems to the complexity of universal machines. Not only is this an intellectual challenge, but also an important issue to identify up to what size machines could escape un-decidability results based on universality or, otherwise, assert that it is too narrow to be worth considering.

There have already been a lot of investigations on small Turing machines [Rogozhin, 1982, 1996, Margenstern, 1995, Kudlek, 1996, Baiocchi, 2001], register machines [Korec, 1996], and cellular automata [Ollinger, 2002, Cook, 2004].

---

*Email address:* [Jerome.Durand-Lose@univ-orleans.fr](mailto:Jerome.Durand-Lose@univ-orleans.fr) (Jérôme Durand-Lose).  
*URL:* <http://www.univ-orleans.fr/lifo/Members/Jerome.Durand-Lose> (Jérôme Durand-Lose).

A machine is said to be *universal* if it can be operated in such a way that: from an initial configuration that encodes any computable function and its argument, the machine will end its computation in a configuration that encodes the value of the function (if any). The end of the computation does not necessarily imply the halting of the machine. By composition, it is enough to simulate a universal machine to be universal.

As smaller and smaller machines were exhibited in very limited systems, various *qualities* of universality arose:

- *polynomial-time universality*: the simulation of any TM  $M$  is done in a time that is polynomial in the running time of  $M$ ,
- *weak universality*: the computation should be started on an ultimately periodic (on each side) infinite configuration —for example, the whole tape of a Turing machine is filled, or for cellular automata which naturally operate on infinite configurations; and
- *semi-universality*: the computation should to be started on an infinite configuration that may or may not be ultimately periodic. The type of initial configurations that are allowed should be stated clearly to ensure the function that encodes the initial configuration is not itself universal.

In the present article, minimal universal machines in the context of *Abstract Geometrical Computation* (AGC) are investigated. Since AGC is a graphical model, it allows us to understand the way the information is moved around and interacts as shown by the various illustrations. AGC has been introduced as a continuous counterpart of cellular automata; this move was inspired by the way dynamics of CA are often designed or analysed in Euclidean space. In AGC, signals are moving with constant speed in Euclidean space. When they meet, they are replaced by new signals; interacting in a collision-based computing way [Adamatzky, 2002, Adamatzky and Durand-Lose, 2012]. A *signal machine* (SM) defines existing kinds of signals, *meta-signals*, and their interactions, *collision rules*.

Since a collision is an exact meeting in a continuous space, if little imprecision is allowed, the collision of three or more signals has a probability of zero. It thus makes sense to distinguish signal machines where only collisions of two signals are considered. These machines are called *robust*.

The straightforward measure of the size of a SM is the number of meta-signals. If there are  $m$  meta-signals defined, then there are at most  $2^m - m - 1$  possible collision rules (at least two signals are needed in a collision and they must be different). Since parallel signals cannot interact, the number of rules could be much lower, but nevertheless exceeding by far the number of meta-signals. In many constructions, only a small portion of potential collision rules are defined, the rest being either undefined or *blank* (i.e. signals just cross each

other). So that the number of specially defined rules is also a good appendage to the size of a SM.

In Durand-Lose [2005b,a], universality in AGC is proven by reduction from 2-counter automata. The simulation time (the longest chain of collisions linked by signals is used to measure the time complexity) is linear, but 2-counter automata simulate TM with a doubly exponential time overhead. Not only is this not polynomial universality, but also the size of such SM is not especially small. Here smaller and polynomial universal machines are provided.

In the present article, universal SMs are generated from Turing machines, cellular automata and cyclic tag systems. In each case, special care is taken in order to save signals and collision rules.

For Turing machines, the linear-time simulation of Turing machines in Durand-Lose [2009b] is improved to fit into a bounded space. Bounds are provided according to known small universal Turing machines [Neary and Woods, 2006b, 2009a, Woods and Neary, 2006, 2007, 2009b,a]

Cellular automata (CA) are massively parallel devices composed of cells regularly displayed and interconnected. Cells are identical finite automata that are updated synchronously. The generated dynamics are very rich [Sarkar, 2000, Kari, 2005]. It should be noted that the signals are a key notion in CA [Delorme and Mazoyer, 2002, Mazoyer and Terrier, 1999]. The main difference from AGC is that CA operate in discrete time and space, whereas these are continuous for signal machines. Tight bounds exist for universal CA [Ollinger, 2002, Cook, 2004, Richard and Ollinger, 2008]. They provide small weakly universal SMs.

**Theorem 1** *There is a non-halting weakly universal signal machine with 5 meta-signals and 7 non-blank rules. There is a non-halting weakly universal robust signal machine with 6 meta-signals and 9 non-blank rules.*

A Cyclic tag system (CTS) is defined by a list of binary word called *appendants*. The input to a CTS is a binary word called the *data word*. At every iteration, the first bit of the data word is discarded and if it is 1, the first appendant of the list is added to the end of the word; the list of appendants is then rotated. CTSs are polynomial-time universal [Neary and Woods, 2006a] and have been used to provide what is currently the most efficient simulation time overheads for the smallest weakly universal cellular automata [Cook, 2004]. Economic simulations of CTSs provide the following bounds.

**Theorem 2** *There is a universal halting signal machine with 13 meta-signals and 21 non-blank rules. There is a universal halting robust signal machine with 15 meta-signals and 24 non-blank rules.*

Signal machines are presented in Section 2. Each following section deals with a different universal model and presents ways to simulate them with SMs: Turing machines in Section 3, cellular automata in Section 4, and cyclic tag systems in Section 5.

## 2 Definitions

### 2.1 Signal machines and space-time diagrams

In *Abstract geometrical computation*, dimensionless objects move on the real axis. When they collide, they are replaced according to given collision rules. This is defined by the following machines.

**Definition 3** A *signal machine* is defined by  $(M, S, R)$  where  $M$  (*meta-signals*) is a finite set,  $S$  (*speeds*) a mapping from  $M$  to  $\mathbb{R}$ , and  $R$  (*collision rules*) a function from the subsets of  $M$  of cardinality at least two into subsets of  $M$  (all these sets are composed of meta-signals of distinct speeds).

Each instance of a meta-signal is a *signal*. The mapping  $S$  assigns *speeds* to signals. They correspond to the inverse slopes of the line segments in space-time diagrams. A *collision rule*,  $\rho^- \rightarrow \rho^+$ , defines what emerges ( $\rho^+$ ) from the collision of two or more signals ( $\rho^-$ ). Since  $R$  is a function, signal machines are deterministic. The *extended value set*,  $V$ , is the union of  $M$  and  $R$  plus one symbol for void,  $\emptyset$ . A (*finite*) *configuration*,  $c$ , is a mapping from  $\mathbb{R}$  to  $V$  such that the set  $\{x \in \mathbb{R} \mid c(x) \neq \emptyset\}$  is finite. An *infinite configuration* is a similar mapping such that the previous set has no accumulation point.

A signal corresponding to a meta-signal  $\mu$  at a position  $x$ , i.e.  $c(x) = \mu$ , moves uniformly with constant speed  $S(\mu)$ . A signal must start (resp. end) in the initial (resp. final) configuration or in a collision. This corresponds to condition 1 in Def. 4. At a  $\rho^- \rightarrow \rho^+$  collision, signals corresponding to the meta-signals in  $\rho^-$  (resp.  $\rho^+$ ) must end (resp. start) and no other signal should be present (condition 2).

**Definition 4** The *space-time diagram* issued from an initial configuration  $c_0$  and lasting for time  $T$  is a mapping  $c$  from  $[0, T]$  to configurations (i.e. from  $\mathbb{R} \times [0, T]$  to  $V$ ) such that  $\forall (x, t) \in \mathbb{R} \times [0, T]$ :

- (1) if  $c_t(x) = \mu$  then  $\exists t_i, t_f \in [0, T]$  with  $t_i < t < t_f$  or  $0 = t_i \leq t < t_f$  or  $t_i < t \leq t_f = T$  s.t.:
  - $\forall t' \in (t_i, t_f)$ ,  $c_{t'}(x + S(\mu)(t' - t)) = \mu$ ,
  - $t_i = 0$  or  $(c_{t_i}(x + S(\mu)(t_i - t)) = \rho^- \rightarrow \rho^+$  and  $\mu \in \rho^+$ ),

- $t_f=T$  or  $(c_{t_f}(x + S(\mu)(t_f - t)) = \rho^- \rightarrow \rho^+$  and  $\mu \in \rho^-$ );
- (2) if  $c_t(x)=\rho^- \rightarrow \rho^+$  then  $\exists \varepsilon, 0 < \varepsilon, \forall t' \in [t - \varepsilon, t + \varepsilon] \cap [0, T], \forall x' \in [x - \varepsilon, x + \varepsilon],$ 
  - $(x', t') \neq (x, t) \Rightarrow c_{t'}(x') \in \rho^- \cup \rho^+ \cup \{\emptyset\},$
  - $\forall \mu \in M, c_{t'}(x') = \mu \Leftrightarrow$  or  $\begin{cases} \mu \in \rho^- \text{ and } t' < t \text{ and } x' = x + S(\mu)(t' - t), \\ \mu \in \rho^+ \text{ and } t < t' \text{ and } x' = x + S(\mu)(t' - t). \end{cases}$

Any signal should be traced to the bottom of the diagram (there is no infinite backward sequence of signals).

On space-time diagrams, time is increasing upward. The traces of signals are line segments whose directions are defined by  $(S(\cdot), 1)$  (1 is temporal). Collisions correspond to the extremities of these segments. This definition can easily be extended to  $T = \infty$  and to infinite configurations.

Although speeds may be any real and thus encode information, in the following, only a few rational values are used. Similarly, the distance between signals may be any real but only rational positions are used. Then, because of the way they are computed, all successive collisions happen at rational locations.

## 2.2 Computation, universality and complexity measures

Usually, computing means that when the device is started on an initial configuration representing the input, it “halts” with the output represented in the final configuration. The problem in AGC —as well as for cellular automata— is that the device does not halt. To cope with this, two approaches could be considered:

- an observer checks that the “end of the computation” is reached according to some property over the configuration (for example, the presence of a given meta-signal) or
- the system reaches a stable state, i.e. no more collision is possible.

The latter is preferred since the ‘halting’, i.e. stabilizing, is then a part of the system. Nevertheless, when simulating never halting systems, like cellular automata, the discussion arises again and the end of computation criterion has to be “imported” too.

Like any computing system, a SM is (*Turing*) *universal* if there exist representations through which any Turing machine (or any equivalent model of computation) and an input can be given to the SM, and the output (if any) represents the output (if any) of the TM on the input. (As soon as one output is undefined, so must be the other.)

To consider *polynomial-time universality*, the time complexity of a computation should be defined. A space-time diagram can be deformed by changing speeds and positions in the initial configuration so that only the locations of generated collisions and signals are changed [Durand-Lose, 2003, Chap. 7]. Space and time are continuous; by rescaling a finite computation, the duration can be made as short as we want, and moreover: any infinite computation starting from a finite configuration can be automatically folded into a finite portion of the space-time diagram [Durand-Lose, 2009b]. So that a correct notion of time complexity does not lie in elapsed continuous time.

Collisions can be considered as discrete steps related by signals: a collision is *causally before* another if a signal generated by the first one ends in the second one. When taking collisions as nodes and signals (except from initial and final configurations) as edges, a direct acyclic graph (DAG) is generated.

**Definition 5** The *time complexity* of a SM computation is the longest length of a chain in the collision-causality DAG formed by collisions and signals. A SM is *polynomial-time universal* if the slowdown in simulating a Turing machine is bounded by a polynomial in the size of the running time of the simulated computation.

If the universality construction relies on ultimately periodic (on each side) infinite configurations, then one speaks of *weak universality*.

### 2.3 Generating an infinite periodic signal pattern

Generating a periodic infinite sequence of signals inside a space-time diagram is useful to provide, starting from a finite number of signals, the infinite data for weak universality. At the cost of extra meta-signals and collision rules, it allows to fall back into regular universality as soon as there is periodicity.

The construction works as follows: a signal bounces back and forth between two “walls”, each time it bounces on the one side, the next signal in the sequence is emitted. The signals used for the wall record the location in the pattern and appear periodically  $\text{wall}_1, \text{wall}_2 \dots \text{wall}_1, \text{wall}_2 \dots$ . The construction is simple and straightforward. It is only presented on an example: an infinite sequence of period 3,  $(\mu_1\mu_2\mu_3)^\omega$ , where each  $\mu_i$  is already defined. The added meta-signals and collision rules as well as the resulting space-time diagram are displayed in Fig. 1.

To save one bouncing meta-signal, the  $\mu_i$  signals are directly generated on the bottom wall. The number of added meta-signals is one plus the period (the output signals are not counted). The number of collision rules is twice the period.

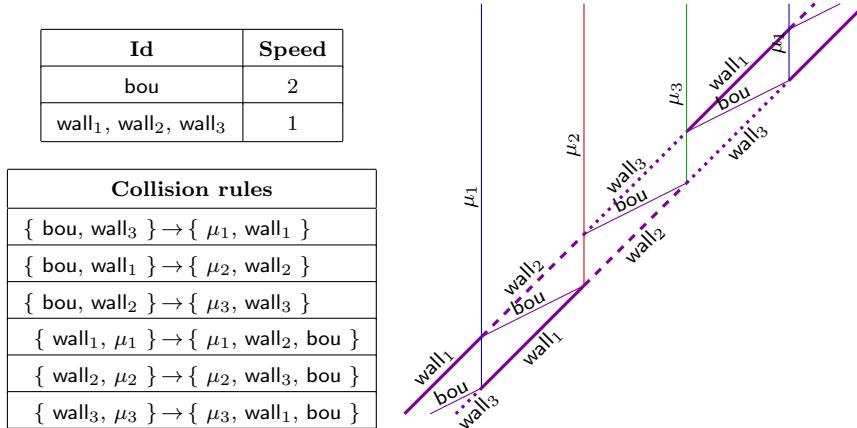


Fig. 1. Generating a periodic pattern.

If the output signals are not parallel or to be set at unequal distances, following the scheme exemplified in Fig. 2, this can be done by adding one plus twice the period of meta-signals and twice the period of collision rules.

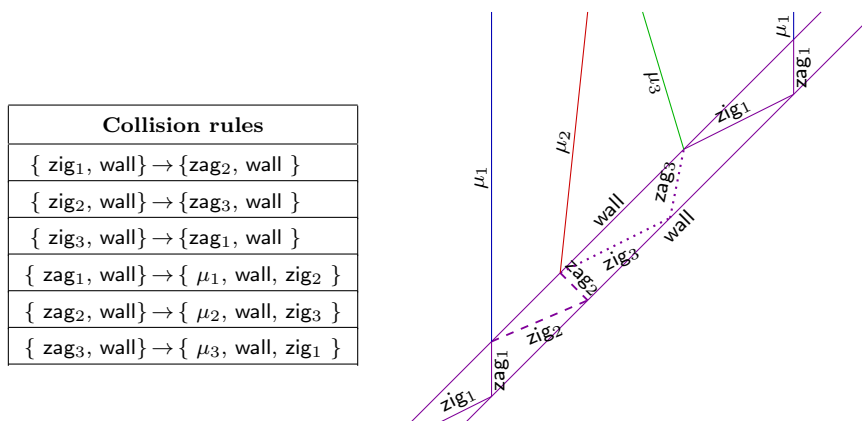


Fig. 2. Generating an unevenly spaced periodic pattern.

Altogether, this is useful to turn weak universality into standard universality (over a finite configuration), but, unless the period is very short, the modified SM is much bigger.

### 3 Turing machines

Only notations and constructions are presented since the reader is expected to have some knowledge of Turing machines. This section provides small polynomial-time universal SMs. It also illustrates AGC.

A *Turing machine* (TM) is defined by  $(Q, q_i, \Gamma, \#, \delta)$  where  $Q$  is a finite set of *states*,  $q_i$  is the *initial state*,  $\Gamma$  is a finite set of *symbols*,  $\#$  (*blank*) is a

distinguished symbol, and  $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{\leftarrow, \rightarrow\}$  is the *transition function*.

The construction is exemplified in the TM given in Fig. 3 and its operation in Fig. 4(a).

$Q = \{q_i, q_1, q_2, q_f\}$	$\delta$	$\hat{\phantom{a}}$	$a$	$b$	$\#$
$\Gamma = \{\hat{\phantom{a}}, a, b, \#\}$	$q_i$	$q_i, \hat{\phantom{a}}, \rightarrow$	$q_1, a, \rightarrow$	$q_1, a, \leftarrow$	-
=	$q_1$	-	$q_1, b, \rightarrow$	-	$q_2, a, \rightarrow$
“-” means undefined	$q_2$	-	-	-	$q_f, b, \leftarrow$
	$q_f$	$q_f, \hat{\phantom{a}}, \leftarrow$	$q_f, a, \leftarrow$	$q_f, b, \leftarrow$	-

Fig. 3. Definition of the TM.

The simulation goes as follows: there are finitely many parallel signals (encoding the content of the tape) in which zigzags a signal mimicking the movement of the head and encoding the state.

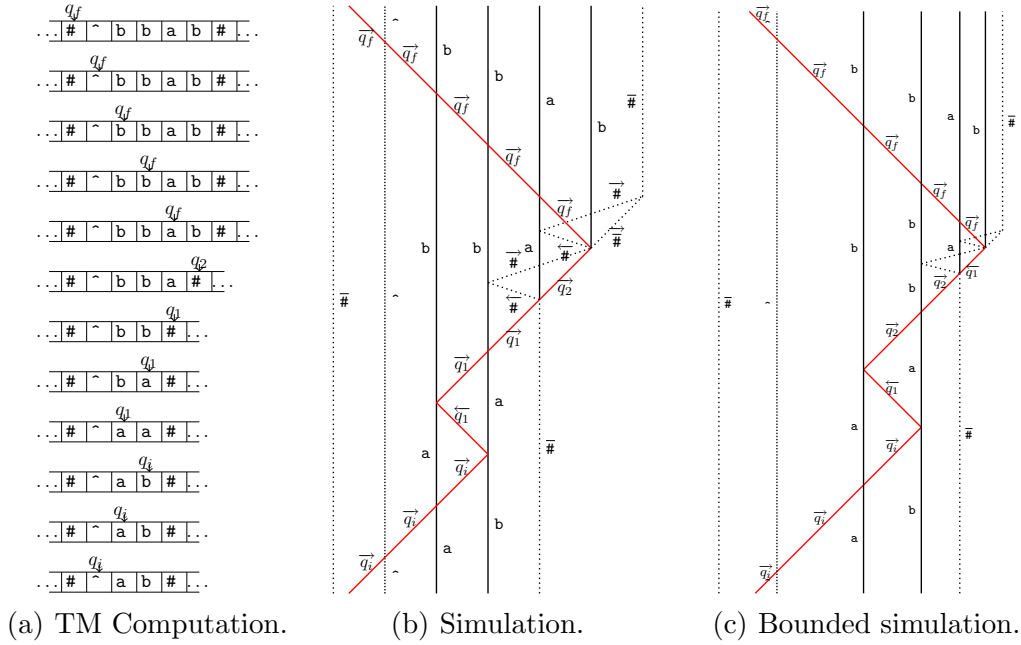


Fig. 4. Example of a TM computation and its simulation by a SM.

The simulating SM is defined by the following meta-signals:

- one *symbol meta-signal* (of speed 0) for each value in  $\Gamma$ , to encode the cells of the tape;
- $\overrightarrow{q}$  (of speed 1) and  $\overleftarrow{q}$  (of speed  $-1$ ) *head meta-signals* for each state  $q$ , to encode the state and the location and movement of the head;
- $\overline{\#}$  (of null speed),  $\overleftarrow{\#}$  (of speed  $-3$ ),  $\overrightarrow{\#}$  (of speed 3),  $\overleftarrow{\#}$  (of speed  $-1$ ) and  $\overrightarrow{\#}$  (of speed 1) which are used to denote the extremities of the tape and to manage the enlargement of the tape on both sides.



The number of meta-signals in the simulating signal machine is  $2|Q| + |\Gamma| + 5$  ( $\bar{\#}$  is distinct from  $\#$ ). For weak universality, bounding and enlarging signals are unneeded, so that this number drops to  $2|Q| + |\Gamma|$ .

A SM configuration encodes a TM configuration if it corresponds to the same sequence of symbols (up to trailing  $\#$ 's on both ends) surrounded by  $\bar{\#}$ 's together with an extra signal encoding the state. The latter signal should be moving to meet the signal corresponding to the position of the head.

The collision rules ensure that the evolution of the SM corresponds to the computation of the TM. When the head encounters a symbol signal, it performs the update and moves to its next position on the tape. When it meets the special signal,  $\bar{\#}$ , the configuration is automatically enlarged on the corresponding side.

The scheme to generate the collision rules is given in Fig. 5. For each input of the TM transition function, two collision rules are generated accounting for the cases where the head would come from the left or from the right. The extremity rules ensure a correct enlargement of the configuration in the case where a head signal would meet  $\bar{\#}$ , the signal marking the last cell on both sides. (Only the right case is presented, the left one being symmetric.) In such a case, two things have to be done: simulate the TM transition (as if it were a  $\#$  signal) and enlarge the configuration. If the head goes left (first case), the latter means generating a new  $\bar{\#}$  signal on the right (right column). If the head goes right (second case, left column), then the head signals must meet something on the right; what is met is a  $\bar{\#}$  signal as presented on the second row. Single rules ensure that  $\bar{\#}$  bounces on the first signal on the left as  $\bar{\#}$ . This  $\bar{\#}$  signal meets  $\bar{\#}$  and regenerates  $\bar{\#}$  (first case); otherwise (second case) it participates in a collision from the second row (enlarging again). This is illustrated in Fig. 4.

The number of collision rules is:

- 2 for each entry of the transition table of the TM,
- plus 4 for each transition on the blank symbol,
- plus 4 (for enlarging the table).

Considering the known limit curves for universality in Neary and Woods [2009a,b] and Woods and Neary [2009a], this leads to the following values: 19 meta-signals for polynomial-time universality for Rogozhin's Rogozhin [1996] 4-state 6-symbol universal TM. The number of collision rules is bounded by 62. The exact numbers have not been computed since a construction with fewer meta-signals is possible.

With the 3-state 3-symbol polynomial-time weakly universal TM from Neary

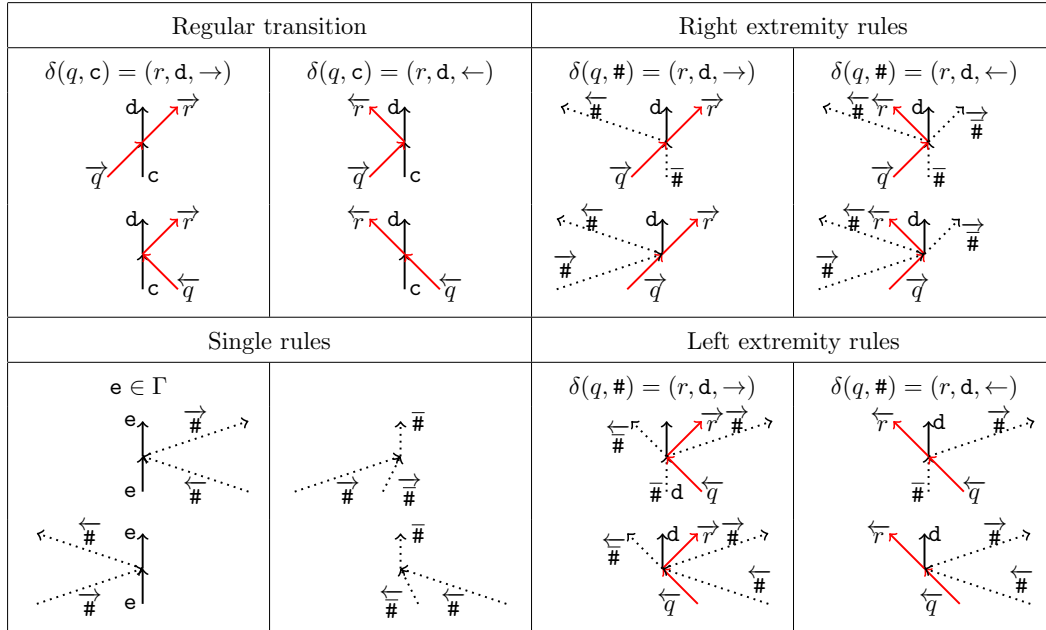


Fig. 5. Collision rules generated.

and Woods [2009a], we generate a polynomial-time weakly universal SM with only 9 meta-signals.

Using Smith [2007], the 2-state 3-symbol TM translates into a semi-universal SM with 7 meta-signals (the additional five meta-signals are not needed) and 6 collision rules. Please note that the universality of Smith's construction is still a controversial issue.

The computation is robust, the positions do not have to be exact as long as the order of the signals is preserved.

With a slight increase of the speeds of  $\overleftarrow{\#}$  and  $\overrightarrow{\#}$ , the tape can be enlarged not by one unit but by a geometrically decreasing length. This is displayed in Fig. 4(c). This provides a way to compute in a spatially bounded portion of space. The drawback is that if the head moves to the left forever, then in finite duration, an accumulation is generated: infinitely many signals and collisions in a bounded part of the space-time diagram.

#### 4 Cellular automata

Cellular automata (CA) operate over infinite arrays of cells. Each cell can be in finitely many states. (The infiniteness of the array is the only way to ensure unbounded memory.) A CA changes the states of all the cells simultaneously. Each cell is updated according to a common local function and the states of

the two surrounding cells. This is a parallel, synchronous, local and uniform process.

**Definition 6** A cellular automaton is defined by  $(Q, f)$  where:  $Q$  is a finite set of states,  $f : Q^3 \rightarrow Q$ , is the local function. The global function,  $\mathcal{G} : Q^{\mathbb{Z}} \rightarrow Q^{\mathbb{Z}}$ , is defined by:  $\forall i \in \mathbb{Z}, \mathcal{G}(c)_i = f(c_{i-1}, c_i, c_{i+1})$ .

Only CA of dimension 1 and radius 1 are considered here. Higher dimensions can be covered similarly by higher dimension signal machines. Radius 1 means that a cell only communicates with its two closest neighbours (one on each side). Broader radii could have been considered, but more signals are used to convey information at greater distances (for example, for radius 2, 5 meta-signals per state instead of 3).

Cellular automata are inherently non-halting. So that halting is provided by an outside observer according to some predefined condition like: reaching a stable/periodic configuration, the appearance of a state/pattern in the configuration or on some designated cell. Detecting any of these would require too many meta-signals.

The simulation works by locating the cells at equal distances. Each time the local function is used, three signals are emitted, one for the cell and one for each of the two closest cells. For each state,  $s$ , there are three meta-signals corresponding to these three signals:  $s_L$ ,  $s$  and  $s_R$  of speeds  $-1$ ,  $0$  and  $1$  respectively. A transition is performed when a cell received simultaneously the values from its two neighbours. The local function is encoded in the collision rules:

$$\text{if } f(s, t, u) = v \text{ then } \{s_R, t, u_L\} \rightarrow \{v_L, v, v_R\}.$$

The rule 110 (elementary) CA is presented in Fig. 6 as well as a generated collision rule.

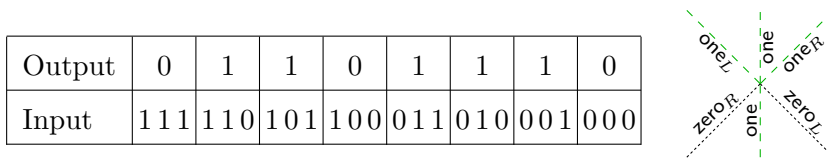


Fig. 6. Local transition function and the collision rule implementing  $f(0, 1, 0) = 1$ .

CA work on infinite configurations. With rule 110 —proved weakly universal in Cook [2004]— a 6-meta-signal 8-collision-rule weakly universal SM is generated. Moreover, one signal, say  $\text{one}_R$ , can be easily removed. This “ghost” signal can be identified by the presence of only two signals of speeds  $0$  and  $-1$ , instead of three. The other collisions of two signals are blank (with signals of speeds  $1$  and  $-1$  thus differentiating them). Noticing that the transition  $f(0, 0, 0) = 0$  generates a blank rule, this leads to a 5-meta-signal 7-collision-rule weakly universal SM.

To consider standard universality, infinitely many cells should be encoded with finitely many signals. The configurations used are quasi-periodic so that the periodic pattern generation of Subsect. 2.3 can be used. When a cell is generated, it corresponds to a future iteration, so that the periodicity to be considered is not the spatial one but the one following a space-time line which might have a very different period. Since periodic configurations have (ultimately) periodic orbits, following a rational space-time discrete line, a periodic observation is found. The construction is only presented on an example in Fig. 7: evolution of rule 110 CA on the configuration 11. The periodic space-time discrete lines are typed in boldface:  $\omega(10)$  on the left and  $(011)^\omega$  on the right.

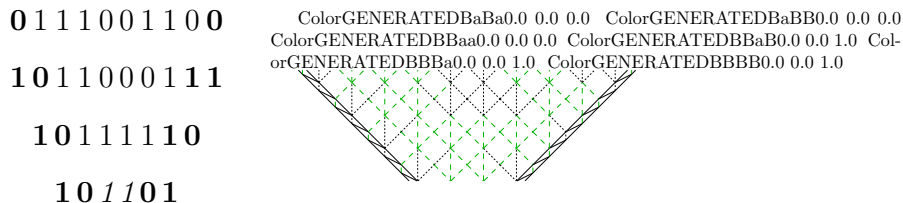


Fig. 7. Evolution and simulation of rule 110 on 11 framed by  $\omega(10)$  and  $(011)^\omega$ .

With the references in Ollinger [2008] and Durand-Lose [2009a], it is hard to go below, say, 30 meta-signals for standard universality. Ollinger [2002], Cook [2004] and Richard and Ollinger [2008] provide universal CA with very few states but they use periodic extensions encoding a boolean function or a cyclic tag system, which leads to many meta-signals.

#### 4.1 Robust weak universality

Previous construction is not robust since it works with collisions of three signals. If one signal is slightly moved, then the exact collision of three signals is transformed into a two-signal collision plus a signal on the side.

The CA simulation can be turned robust by adding a few rules and signals: leftward meta-signals are replaced by faster ones, and then each central still signal receives the information from the left (a 2-collision) and then from the right (another 2-collision). This is depicted on the example of Fig. 8.

The robustness is at the price of extra meta-signals to encode the information brought by the first two signals. Each kind of 2-collision is identified by the speed of incoming signals:  $-1$  and  $1$  for blank collision (information exchange between cells),  $-1$  and  $0$  for the first part of the transition and  $0$  and  $1$  for the second part. This means, that null-speed signals may be freely reused. Only three new meta-signals (instead of four) are needed since  $z \mapsto f(1, 0, z)$  and  $z \mapsto f(0, 0, z)$  are the same function. This cost one extra meta-signal.

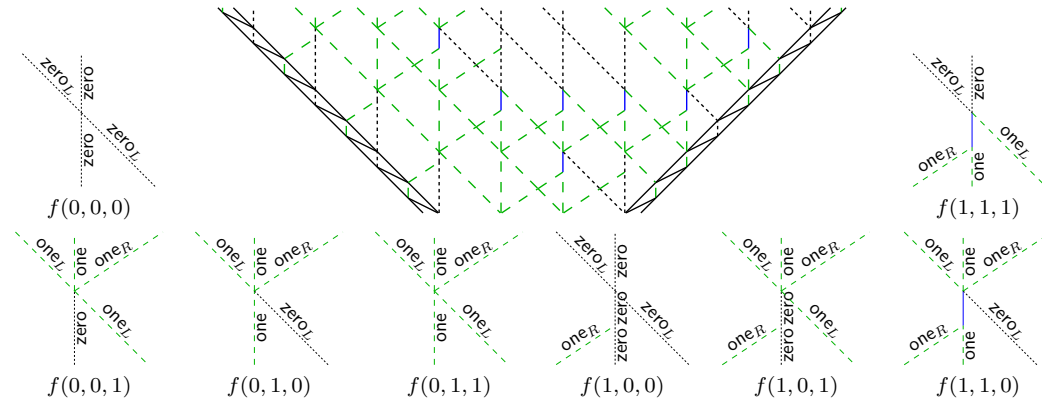


Fig. 8. Robust simulation of rule 110 with a ghost signal.

The ghost signal trick can be used again as depicted in Fig. 8, leading to 6 meta-signals and 9 rules: 4 (for the left and central bit) plus 5 (3 cases by 2 values for the last bit minus one blank rule for 0,0) 2-meta-signal rules.

## 5 Cyclic tag systems

A Cyclic tag system (CTS) is defined by a list of binary word called *appendants*. The input to a CTS is a binary word called the *data word*. The system is updated in the following way: the first bit of the data word is removed. If it is 1, then the first appendant is appended at the end of the data word (otherwise nothing is done). Then, the list is rotated circularly. The systems stabilize by reaching an empty data word or by entering a cycle. It can also be decided that when a special appendant is triggered then the systems halts and the data word is the output. Not only are CTSs universal [Cook, 2004] but they also simulate TM in polynomial time [Neary and Woods, 2006a].

The simulation follows the structure of CTSs: parallel signals are used to encode the data word and the circular list. At every iteration, depending on the first signal, a copy of the first appendant is appended (for 1) or not (for 0). Signals encoding the data word are configured in such a way that delivered copies automatically enlarge the data word on the right.

The signals used in the simulation are defined in Figure 9. The following naming convention is used: meta-signals with no subscript have speed 0 and the ones with subscript *LL*, *R* and *RR* have speed  $-2$ ,  $1$  and  $2$  respectively.

As illustrated in Figure 10, the initial configuration is composed, left to right, of: **last**,  $\text{go}_{LL}$  that starts the dynamics, then **one**'s and **zero**'s to encode the data word, then **first** to indicate the beginning of the cyclic list, then **one**'s and **zero**'s to encode each appendant, and **sep** to separate one appendant from the next, and finally **last**. The iteration starts when the  $\text{go}_{LL}$  signal erases

Speed	Meta-Signals
-2	go <sub>LL</sub>
0	zero, one, first, sep, last
1	zero <sub>R</sub> , one <sub>R</sub> , false <sub>R</sub> , true <sub>R</sub>
2	zero <sub>RR</sub> , one <sub>RR</sub> , go <sub>RR</sub>

Fig. 9. List of all the meta-signals.

last to give go<sub>RR</sub>. The latter signal erases the first bit, restores last and sends the corresponding signal, zero<sub>R</sub> or one<sub>R</sub> to the cyclic list. Once the rotation is initiated and the possible addition of signals is done, a go<sub>LL</sub> signal is sent back through the data word to the last on the left.

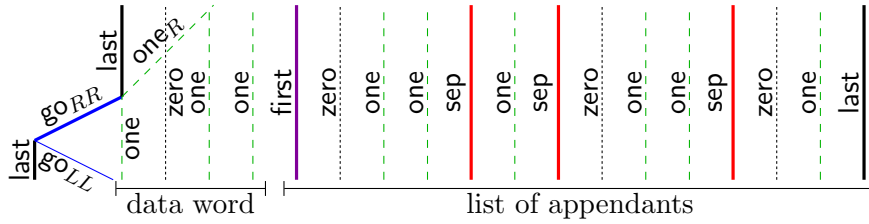


Fig. 10. Initial configuration and first collisions for 1011 and list [011, 1, 011, 01].

The emitted one<sub>R</sub> or zero<sub>R</sub> crosses the one's and zero's encoding the data word without modifying anything until first is reached, then the rotation starts. (For technical reasons linked to the end of the rotation, first remains and is later removed by go<sub>LL</sub>.) If the data word is empty, then go<sub>RR</sub> meets first. Reaching the empty word is a halting condition so that, in such a case, go<sub>RR</sub> is just discarded.

Figure 11 lists all the collision rules. All non-blank rules are collision of only two signals so that the rules can be presented in a two-dimensional array.

	zero	one	first	sep	last	go <sub>LL</sub>
go <sub>RR</sub>	last, zero <sub>R</sub>	last, one <sub>R</sub>	first			
one <sub>RR</sub>	zero, zero <sub>R</sub> , one <sub>RR</sub>	one, one <sub>R</sub> , one <sub>RR</sub>		go <sub>LL</sub> , last, false <sub>R</sub>		
zero <sub>RR</sub>	zero <sub>R</sub> , zero <sub>RR</sub>	one <sub>R</sub> , zero <sub>RR</sub>		go <sub>LL</sub> , last, false <sub>R</sub>		
true <sub>R</sub>	—	—			first, false <sub>R</sub>	—
false <sub>R</sub>	—	—	sep, go <sub>RR</sub>	—	first, one <sub>R</sub>	go <sub>LL</sub> , true <sub>R</sub>
one <sub>R</sub>	—	—	first, true <sub>R</sub> , one <sub>RR</sub>	—	—	—
zero <sub>R</sub>	—	—	first, false <sub>R</sub> , zero <sub>RR</sub>	—	—	—
go <sub>LL</sub>	—	—	go <sub>LL</sub>		go <sub>RR</sub>	

	zero <sub>R</sub>	one <sub>R</sub>	false <sub>R</sub>	true <sub>R</sub>
go <sub>RR</sub>		last	zero, go <sub>RR</sub>	one, go <sub>RR</sub>
one <sub>RR</sub>		—	—	—
zero <sub>RR</sub>		—	—	—

“—” means blank  
“ ” means unused

Other blank rules
zero <sub>R</sub> , zero, go <sub>LL</sub>
zero <sub>R</sub> , one, go <sub>LL</sub>
one <sub>R</sub> , zero, go <sub>LL</sub>
one <sub>R</sub> , one, go <sub>LL</sub>

Fig. 11. List of all the collision rules.

The rotation is handled in three steps: signals for the first appendant are set on movement then they freely move to the right end of the list, and finally after reaching last, they are positioned. The first step is presented in Figure 12. A signal of speed 2 ( $\text{zero}_{RR}$  or  $\text{one}_{RR}$ ) is emitted to generate the speed 1 versions of zero's ( $\text{zero}_R$ ) and one's ( $\text{one}_R$ ) encoding the first appendant.  $\text{one}_{RR}$  leaves a copy of the first appendant (which appears at the end of the data word) while  $\text{zero}_{RR}$  does not. This  $\text{one}_{RR}$  or  $\text{zero}_{RR}$  stops at the first  $\text{sep}$  which is replaced by last and then by first. (Basically, this is the case because both  $\text{zero}_R$  and  $\text{one}_R$  react with first.)

Moving  $\text{zero}_R$ 's and  $\text{one}_R$ 's are surrounded by two signals with the same speed,  $\text{false}_R$ , to drive the end of the rotation. The bottom right  $\text{false}_R$  signal is generated when  $\text{zero}_{RR}$  (or  $\text{one}_{RR}$ ) reaches  $\text{sep}$ . The top left  $\text{false}_R$  is generated when the translation starts (i.e. at the collision on first). As can be seen in Fig. 12, this signal has identity  $\text{false}_R$  (only for  $\text{zero}_R$ ), then  $\text{true}_R$  (after  $\text{go}_{LL}$ ) and finally  $\text{false}_R$  (on last). The signal to start the next iteration,  $\text{go}_{LL}$ , is emitted when meeting the first  $\text{sep}$  (it is not a problem since the next iteration cannot catch up with the rotation). This  $\text{go}_{LL}$  signal is used to turn  $\text{false}_R$  to  $\text{true}_R$  (and to provide a simple halting scheme, as explained later).

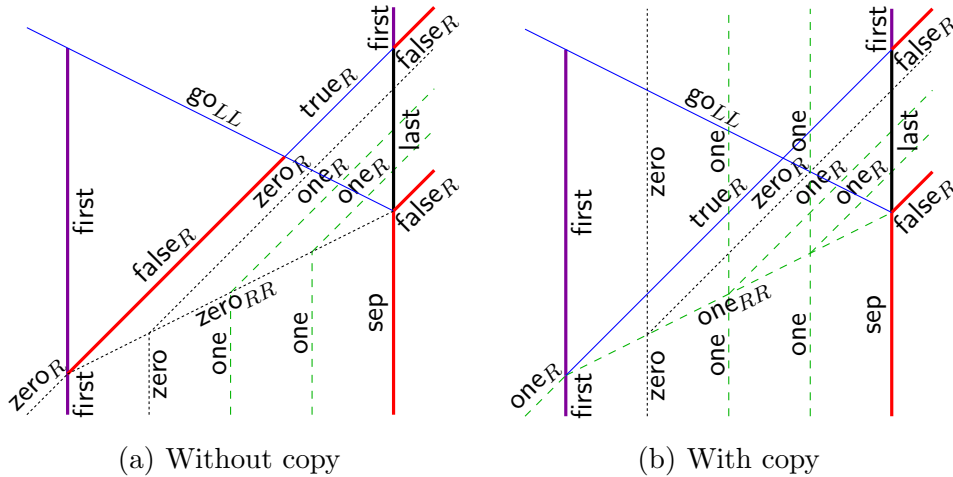


Fig. 12. Starting the rotation of appendants for the current simulated time step and providing the signal  $\text{go}_{LL}$  that starts the simulation of the next time step.

Step 2 of the rotation, the translation, is straightforward: the parallel  $\text{false}_R$ ,  $\text{zero}_R$  and  $\text{one}_R$  cross  $\text{sep}$ ,  $\text{zero}$  and  $\text{one}$  until last is eventually met.

The last step of the rotation, as presented in Figure 13, is almost symmetric to the first part. To save meta-signals,  $\text{false}_R$  has been used to mark both the beginning and the end of the appendant. But they have different meanings, so the first one changes last to first and the second to  $\text{sep}$  and  $\text{go}_{RR}$ . The problem is that first interacts with  $\text{zero}_R$  and  $\text{one}_R$ :  $\text{zero}_R$  (resp.  $\text{one}_R$ ) is changed to  $\text{false}_R$  (resp.  $\text{true}_R$ ) and  $\text{one}_{RR}$  (resp.  $\text{zero}_{RR}$ ). This generates the lattice in the

triangle (the signals do not interact inside it). The bits of the appendant are now encoded with  $\text{false}_R$ 's and  $\text{true}_R$ 's. It is the role of  $\text{go}_{RR}$  to turn them back into zero's and one's and to turn the final  $\text{one}_R$  to  $\text{last}$ . Signals  $\text{go}_{RR}$ ,  $\text{zero}_{RR}$  and  $\text{one}_{RR}$  are parallel, all the translating signals are parallel; so that the appendant is recreated with exactly the same distances.

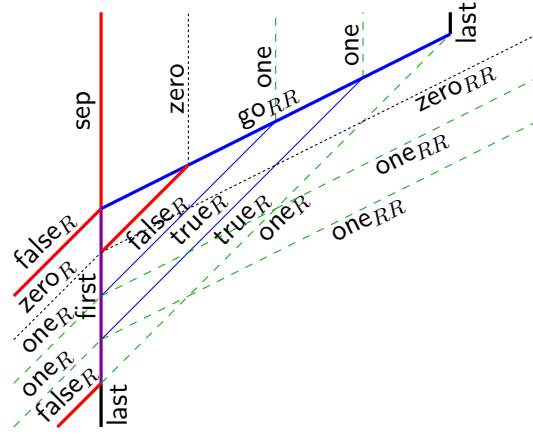


Fig. 13. Ending the rotation.

Figure 14(a) shows one full iteration of a CTS including a whole rotation. Figure 14(b) shows halting by reaching an empty word. Figure 14(c) shows the effect of the halting appendant as explained below. Figure 14(d) shows one entire simulation with a halting appendant and cleaning added: blank rules have been modified in order to destroy the garbage signals escaping on the right that nevertheless would never interact with the rest of the configuration, nor provoke any collision since they are parallel. The modified rules to destroy the signals on the right are:

$$\begin{aligned} \{ \text{zero}_{RR}, \text{one}_R \} &\rightarrow \{ \text{one}_R \} \\ \{ \text{one}_{RR}, \text{one}_R \} &\rightarrow \{ \text{one}_R \} \end{aligned}$$

Theses rules are only used in the collisions in Fig. 13.

### 5.1 Taking halting into account

CTSs stabilize by reaching an empty data word or by cycling. The first case is already implanted as displayed in Fig. 14(b) with the rule

$$\{ \text{go}_{RR}, \text{first} \} \rightarrow \{ \text{first} \} .$$

Cycling is also implemented but detecting it requires more meta-signals.



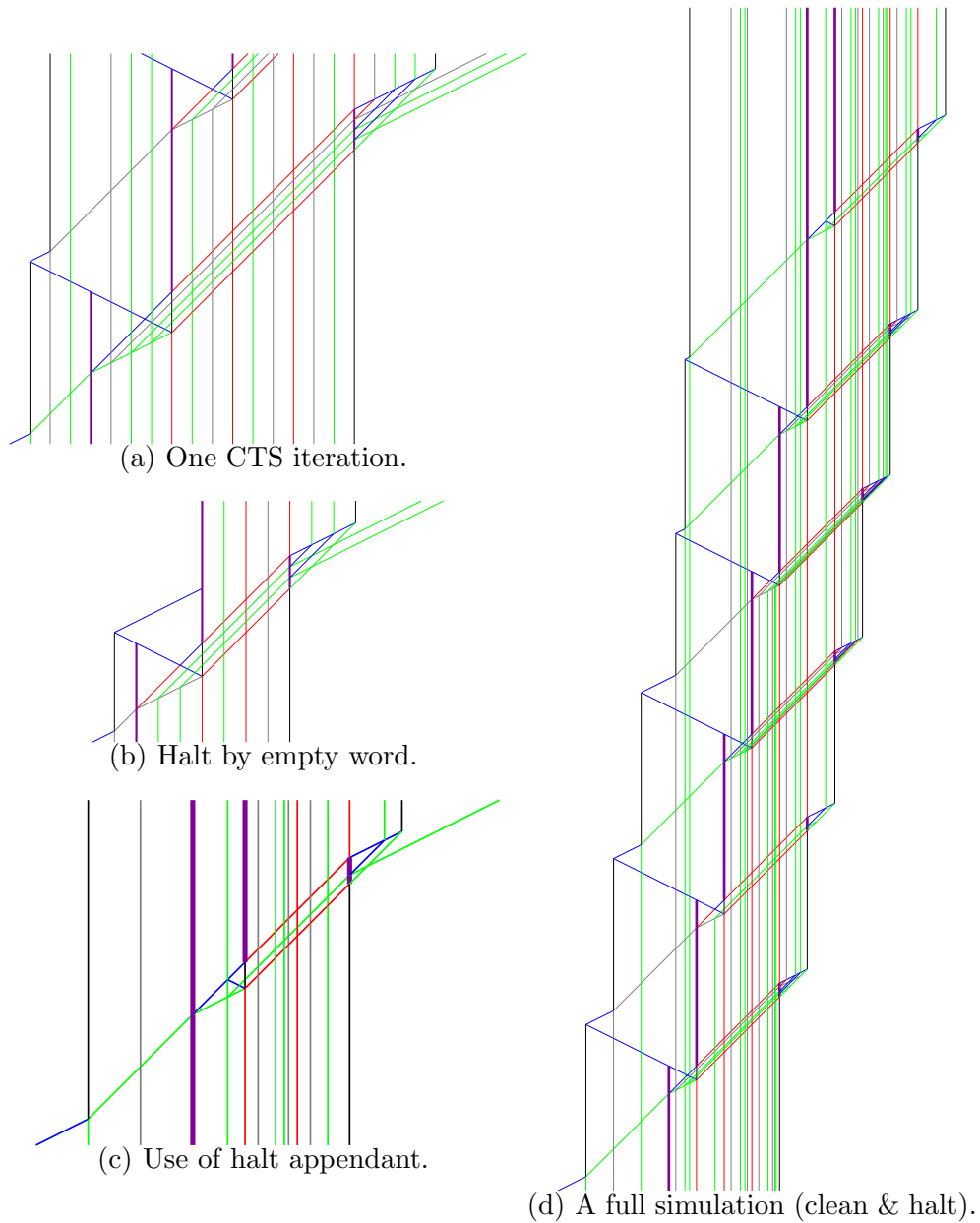


Fig. 14. Simulation of a cyclic tag system.

CTSs can halt by triggering a special appendant. This is implemented by adding two meta-signals for this appendant,  $\text{halt}$  and  $\text{halt}_R$  (in rotation), with speeds 0 and 1, and four collision rules for rotating:

$$\{ \text{zero}_{RR}, \text{halt} \} \rightarrow \{ \text{halt}_R, \text{zero}_{RR} \}$$

$$\{ \text{one}_{RR}, \text{halt} \} \rightarrow \{ \text{halt}, \text{halt}_R, \text{one}_{RR} \}$$

$$\{ \text{go}_{RR}, \text{halt}_R \} \rightarrow \{ \text{halt}, \text{go}_{RR} \}$$

and for halting:

$$\{ \text{go}_{LL}, \text{halt} \} \rightarrow \{ \} .$$

The generated machine is robust.

Altogether, this sums up to 15 meta-signals and 24 non-blank collision rules.

### 5.1.1 *Non-robust version*

Instead of introducing two meta-signals, it is possible to take advantage of one 2-signal collision and turn it into a 3-signal collision in the case the halt appendant is triggered (thus the machine is not robust anymore). The collision between `trueR` and `goLL` in Fig. 12(b) is in the zone crossed by the copy to be appended to the data word. The position of this collision is totally determined by the position of the surrounding `first` and `sep`: it appends at exactly 2/3 of the distance.

If a `one` signal is positioned exactly at this position, if a copy is left, it enters the collision. The extra rule:

$$\{ \text{true}_R, \text{one}, \text{go}_{LL} \} \rightarrow \{ \text{true}_R \}$$

destroys both `one` and `goLL`, so that no new cycles is started. Then, the rotation is finished and no more collision is possible. The result of the computation is the sequence on the left of `first`.

The rotating process ensures that the distances between the signals remain constant, so that a `one` at 2/3 remains there (and a `one` not at 2/3 cannot get there). It should also be ensured that, in the initial position, there is no `one` at 2/3 not standing for halt, which is straightforward to achieve: for example, by using locations 1/2, 3/4, 7/8... between `sep`.

This construction brings forth the need for an exact collision of three signals which is the only non-robust collision rule. Altogether, this sums up to 13 meta-signals and 21 non-blank collision rules.

### *Acknowledgement*

We are very grateful to Matthew Cook for proposing the ghost signal trick which improved the bounds by one in the weak universal cases.

We are also very grateful to Turlough Neary, Anthony Seda, Damien Woods and the anonymous referees for, among other things, clarifying the various notions of universality.

## References

- Andrew Adamatzky, editor. *Collision based computing*. Springer, 2002.
- Andrew Adamatzky and Jérôme Durand-Lose. Collision computing. In David Corne, editor, *Handbook of Natural Computing, Section VII: Broader Perspective — Alternative Models of Computation*, chapter 58, pages 1949–1978. Springer, 2012. doi: 10.1007/978-3-540-92910-9\_58.
- Claudio Baiocchi. Three small universal Turing machines. In Maurice Margenstern and Yurii Rogozhin, editors, *Machines, Computations, and Universality*, number 2055 in LNCS, pages 1–10. Springer, 2001.
- Matthew Cook. Universality in elementary cellular automata. *Complex Systems*, 15:1–40, 2004.
- Marianne Delorme and Jacques Mazoyer. Signals on cellular automata. In Andrew Adamatzky, editor, *Collision-based computing*, pages 234–275. Springer, 2002.
- Jérôme Durand-Lose. *Calculer géométriquement sur le plan – machines à signaux*. Habilitation à Diriger des Recherches, École Doctorale STIC, Université de Nice-Sophia Antipolis, 2003. URL [HDR/index.html](http://HDR/index.html). In French.
- Jérôme Durand-Lose. Abstract geometrical computation: Turing computing ability and undecidability. In Barry S. Cooper, Benedikt Löwe, and Leen Torenvliet, editors, *New Computational Paradigms, 1st Conf. Computability in Europe (CiE '05)*, number 3526 in LNCS, pages 106–116. Springer, 2005a. doi: 10.1007/11494645\_14.
- Jérôme Durand-Lose. Abstract geometrical computation for black hole computation (extended abstract). In M. Margenstern, editor, *Machines, Computations, and Universality (MCU '04)*, number 3354 in LNCS, pages 176–187. Springer, 2005b. doi: 978-3-540-25261-0.
- Jérôme Durand-Lose. Cellular automata, Universality of. In Robert A. Meyers and Andrew Adamatzky, editors, *Encyclopedia of Complexity and System Science*, pages 901–913. Springer, 2009a. doi: 10.1007/978-0-387-30440-3\_59. URL <http://www.springer.com/east/home/physics?SGWID=5-10100-22-173779107-0>.
- Jérôme Durand-Lose. Abstract geometrical computation 3: Black holes for classical and analog computing. *Nat Comput*, 8(3):455–472, 2009b. doi: 10.1007/s11047-009-9117-0.
- Jarkko Kari. Theory of cellular automata: a survey. *Theoret Comp Sci*, 334:3–33, 2005.
- Ivan Korec. Small universal register machines. *Theoret Comp Sci*, 168(2):267–301, 1996.
- Manfred Kudlek. Small deterministic turing machines. *Theoret Comp Sci*, 168(2): 241–255, 1996.
- Maurice Margenstern. Non-erasing Turing machines: A new frontier between a decidable halting problem and universality. In R. Baeza-Yates, E. Goles, and P. Poblete, editors, *LATIN'95*, number 911 in LNCS, pages 386–397. Springer, 1995.
- Jacques Mazoyer and Véronique Terrier. Signals in one-dimensional cellular automata. *Theoret Comp Sci*, 217(1):53–80, 1999. doi: 10.1016/S0304-3975(98)00150-9.
- Turlough Neary and Damien Woods. P-completeness of cellular automaton rule

110. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *International Colloquium on Automata Languages and Programming (ICALP '06)*, number 4051(1) in LNCS, pages 132–143. Springer, 2006a.
- Turlough Neary and Damien Woods. Small fast universal Turing machines. *Theoret Comp Sci*, 362(1–3):171–195, 2006b.
- Turlough Neary and Damien Woods. Small weakly universal Turing machines. In Mirosław Kutylowski, Witold Charatonik, and Maciej Gebala, editors, *Fundamentals of Computation Theory, 17th International Symposium, FCT 2009, Wrocław, Poland, September 2–4, 2009. Proceedings*, volume 5699 of *Lecture Notes in Computer Science*, pages 262–273. Springer, 2009a. doi: 10.1007/978-3-642-03409-1\_24.
- Turlough Neary and Damien Woods. Four fast universal Turing machines. *Fund Inf*, 91(1):123–144, 2009b. doi: 10.3233/FI-2009-0036.
- Nicolas Ollinger. The quest for small universal cellular automata. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Bueno Morales, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *International Colloquium on Automata Languages and Programming (ICALP '02)*, number 2380 in LNCS, pages 318–329. Springer, 2002.
- Nicolas Ollinger. Universalities in cellular automata: a (short) survey. In Bruno Durand, editor, *Journées Automates cellulaires (JAC '08)*, pages 102–118, 2008.
- Gaétan Richard and Nicolas Ollinger. A particular universal cellular automaton. In Turlough Neary, Damien Woods, Anthony K. Seda, and Niall Murphy, editors, *The Complexity of Simple Programs*. National University of Ireland, Cork, 2008.
- Yu. V. Rogozhin. Seven universal Turing machines. In *Systems and theoretical programming*, number 63 in Mat. Issled., pages 76–90. Academia Nauk Moldavskoi SSR, 1982. in Russian.
- Yurii V. Rogozhin. Small universal Turing machines. *Theoret Comp Sci*, 168(2): 215–240, 1996. doi: 10.1016/S0304-3975(96)00077-1.
- Palash Sarkar. A brief history of cellular automata. *ACM Computing Surveys*, 32(1):80–107, 2000.
- Alex Smith. Universality of Wolfram’s 2, 3 Turing machine. *Complex Systems*, 2007. to appear.
- Damien Woods and Turlough Neary. On the time complexity of 2-tag systems and small universal Turing machines. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS '06), Berkeley, CA, USA*, pages 439–448. IEEE Computer Society, 2006.
- Damien Woods and Turlough Neary. Small semi-weakly universal Turing machines. In J. Durand-Lose and M. Margenstern, editors, *Machines, Computations and Universality (MCU '07)*, number 4664 in LNCS, pages 303–315. Springer, 2007.
- Damien Woods and Turlough Neary. Small semi-weakly universal Turing machines. *Fund Inf*, 91(1):179–195, 2009a. doi: 10.3233/FI-2009-0039.
- Damien Woods and Turlough Neary. The complexity of small universal Turing machines: A survey. *Theoret Comp Sci*, 410(4–5):443–450, 2009b. doi: 10.1016/j.tcs.2008.09.051.