Black hole computation: implementations with signal machines

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

1. Introduction

No position is taken on the theoretical and practical feasibility of using any potentially existing particular black hole for hyper-computing. The reader is referred to other contributions in this issue as well as Etesi and Németi (2002); Németi and Dávid (2006); Németi and Andréka (2006) for clear introductions and surveys on this topic. All the work presented here in done the context of so-called "Malament-Hogarth" space-times and not slowly rotating Kerr black hole. The main differences are that, one, the observer remains out of the black hole and second, nested black holes are considered.

A black hole computation works in the following way. At some location[⋆] one observer starts a computing device which he sends on a different world-line such that: at some point, after a finite time on his own world-line the observer has the whole future of the device infinite world-line in its causal past and the device can send some limited information to the observer. The device has an infinite time for computing ahead of it and can send a single piece of information to the observer. Not only can the observer *perceive* this piece of information, but after a finite duration which he is aware of, the observer knows for certain that if he has not received anything then the machine never sends anything during its whole, possibly infinite, computation. So that, for example, if the piece of information is sent only

[⋆] Throughout the article *location* is to be understood as a position in space and time and *position* as only spatial.

when the computation stops, at some point, the observer knows for certain whether the computation ever stops.

From the computer scientist point of view, this allows to have some piece of information on a whole, potentially infinite, computation. This way the Halting problem or the consistency of many first order logic theory (e.g. Set theory or Arithmetic theory) can be decided. This clearly falls out of the classical recursion/computability theory since it allows to decide semi-decidable subsets (Σ_1^0 in the arithmetic hierarchy).

Malament-Hogarth space-times allow to have nested black holes and so-called *arithmetical sentence deciding (SAD) space-times* (Hogarth, 1994, 2004). With each infinite level of nested black-holes, the computing power climbs up a level of the arithmetic hierarchy (and even the hyper-arithmetic hierarchy in second order number theory (Welch, 2006)). The arithmetic hierarchy is formed by recursive predicates preceded by alternating quantifiers on natural numbers. The level in the hierarchy is defined by the first quantifier and the number of alternations.

In computer science, considering and continuing a computation after its potential infinite span exist in various form. Infinite time Turing machines (Copeland, 2002; Hamkins and Lewis, 2000; Hamkins, 2002, 2007) consider ordinal times (and values); the main difference is that the limit of the tape is available, whilst with black holes only a finite and bounded piece of information is available. Infinite computations also provide limits, for example computable analysis (Weihrauch, 2000) (type-2 Turing machines) manipulates infinite inputs and generates infinite outputs, each one representing a real number. In analog computations, limit operators (as limits of sequences of real numbers) are also considered (Chadzelek and Hotz, 1999; Kawamura, 2005; Mycka, 2003b,a, 2006; Mycka and Costa, 2007; Bournez and Hainry, 2007).

The setting in which the black hole effect is simulated here is not the solution to any relativity equations. It is rather something that is constructed inside a Newtonian space-time and provide the desired effect whereas black holes involves non-Euclidean geometry (like the Schwarzschild one). Any initially spatially bounded computation can be embedded in a shrinking structure resulting in the same computation happening in a spatially and temporally bounded room, even if it was not the case before. This structure provide the black hole, outside of it, some information may be received to gain some insight on what happen inside.

Abstract geometrical computation (AGC) deals with dimensionless signals with rectilinear and uniform movement in a (finite dimension) Euclidean space. It is both a continuous and an analog model. The way it is analog is quite unusual because there are finitely many values but continuum many variables. Time is evolving equally everywhere. What brings forth the accelerating power is that space and time are continuous so that Zeno effect can happen and indeed, everything heavily relies on it. Throughout the article, only dimension 1 spaces are considered, thus space-time diagrams are 2 dimensional.

Signals are defined as instances of meta-signals which are of finite number. They move with constant speeds uniquely defined by their meta-signals. When two or more signals meet, a collision happen and they are replaced other signals according to collision rules. A *signal machine* defines the available meta-signals, their speeds and the collisions rules.

A configuration at a given time is a mapping from $\mathbb{R}$ to a finite set (containing the meta-signals) defining signals and their positions. Signals are supposed to be away one from another. The void positions (i.e. position with nothing) are supposed to form an open set. The rest of the configuration contains only singularities. They are many ways to continue a space-time diagram after an isolated singularity, the ones used here are: let it disappear or turn it into signals.

This model originates in discrete signals in Cellular Automata (Durand-Lose, 2008b). This explains the choice of finitely many meta-signals and constant speeds. There are other models of computation dealing with Euclidean spaces.

Huckenbeck (1989, 1991) developed a model based on finite automata able to draw lines and circles and to compute intersections. Not only are the primitives different, but also it is sequential and depends on an external operator to perform the construction whereas in signals in AGC operate on their own.

Jacopini and Sontacchi (1990) developed an approach where a computation results in a polyhedron. They only encompass finite polyhedron (i.e. bounded with finitely many vertices) and allow surfaces and volumes of any dimensions while AGC has only lines.

Another model worth mentioning is the Piecewise Constant Derivative (PCD) system (Bournez, 1999b). There is only one signal, but its speed changes each time it enters a different polygonal region. Not only is AGC parallel but also there is no such things as boundaries. Nevertheless, PCD are able to hyper-compute and climb up the hyper-arithmetic hierarchy (Bournez, 1999a) while the other two cannot.

In AGC, since signals dwell in a continuous space ($\mathbb{R}$) and time ($\mathbb{R}^+$) and there is no absolute scale, it is possible to rescale a configuration. Rescaling a configuration rescales the rest of the space-time. An automatic procedure to freeze the computation, scale it down and unfreeze it, is provided. When this construction is made to restart it-self forever, a singularity is reached. Any AGC-computation starting with only finitely many signals can be embedded in this structure so that the corresponding computation is cut in countably many bounded pieces geometrically shrunk and inserted. This brings a general scheme to transform any AGC-computation into one that can do the same computation but in a piece of the space-time diagram bounded in both space and time. This is the desired black hole effect. Another shrinking scheme is presented in Durand-Lose (2006a), but it works only for spatially bounded AGC-computations while the one presented here does not impose this condition. The construction is detailed since AGC is not so well-known and it is the cornerstone of the results in the article.

Simulating a Turing machine (TM) is easy using one signal to encode each cell of the tape and an extra signal for the location of head and the state of the finite automaton. Any Turing machine simulation can be embedded in the shrinking structure making it possible to decide semi-recursive problems in finite time.

As long as speeds and initial positions are rational and there is no singularity, the location of each collision is rational (as the intersection of lines with rational coefficients). This can be computed in classical discrete computability (and have been implemented to generate figures). The model is also perfectly defined with irrational speeds and positions and can thus be also considered as an analog model. It has recently been connected to the Blum, Shub and Smale model of computation (which can perform algebraic operations as well as test the sign on real numbers with exact precision) (Durand-Lose, 2007, 2008a). In the present paper, it is only recalled how real numbers are encoded and how a singularity is used to provide the multiplication. Our shrinking scheme can be used in this analog context to answer analog decision problems.

The way a signal machine is transformed to have shrinking capability can be iterated so that singularities/accumulations of any order can be generated. This allows to decide a formula formed by a total (resp. BSS) recursive predicate preceded by a finite alternation of quantifiers on $\mathbb{N}$ (i.e. to climb the corresponding arithmetical hierarchies). For the analog case, this remains a quantification over a countable set[⋆⋆].

[⋆⋆]This is why we do not talk about an analytic hierarchy.

In Section 2, the model and the shrinking structure are presented. In Section 3, the way discrete computation can be done and shrunk is presented. In Section 4, analog computation is considered. Section 5 explains how to have nested black holes so as to climb the arithmetic hierarchies. Section 6 gathers some concluding remarks.

2. Model and Mechanics

The underlying time is $\mathbb{R}^+$; there is no such thing as a next configuration. The underlying space is $\mathbb{R}$. A configuration is a mapping from $\mathbb{R}$ to a finite set (yet to be defined). A space-time diagram is a function from $\mathbb{R} \times \mathbb{R}^+$ to the same finite set.

What lies on $\mathbb{R}$ are signals, collisions between signals or singularities (created by accumulations). Signals are moving with constant speed depending only on their nature. When they meet, an instantaneous collision happens and the signals are replaced by others according to some collision rules. A singularity happens when and where infinitely many collisions, signals and/or singularities accumulate. Ways to continue a space-time diagram beyond an isolated singularity are proposed at the end of this section.

2.1. *Signal machines and space-time diagrams*

Definition 1 A *signal machine (SM)* is defined by (M, S, R) where M (*meta-signals*) is a finite set, S (*speeds*) a function from M to $\mathbb{R}$, and R (*collision rules*) a partial function from the subsets of M of cardinality at least two into subsets of M (all these sets are composed of signals of distinct speeds).

Each signal is an instance of a meta-signal. Its speed is constant and only depends on its meta-signal (given by S). When two or more signals meet, R indicates the signals to replace them. Meeting signals must have distinct speeds otherwise they are parallel and never meet. Signals are not allowed to be superposed, so that all signals emitted by a collision must also have distinct speeds.

Definition 2 The *extended value set*, V, is the union of M and R plus two special values: one for void, $\oslash$, and one for singularity ❊. An *(valid) configuration* is a total function from $\mathbb{R}$ to V such that all the accumulation points of its *support* (the set of non void location, $supp(c) = \{ x \in \mathbb{R} \mid c(x) \neq \oslash \}$) have the value ❊. A configuration is *finite* if its support is finite. It is *simple* if it is finite and ❊ is not reached. It is *rational* if it is simple and its support is included in $\mathbb{Q}$. A SM is *rational* if all the speeds are rational numbers and only rational configurations are used.

As long as there is no singularity, a finite configuration is only followed by finite ones.

To be rational is *robust*. Signals at rational positions with rational speeds can only meet at rational location. All collisions happen at rational dates and at these dates the positions of signals are all rational. Since rational numbers can be encoded and manipulated exactly with any computer, rational SM can be handled inside classical computability theory.

Two results limit the interest and extend of rational SM. First, predicting whether a singularity ever happens is Σ_2^0-complete (in the arithmetical hierarchy, which means not even semi-decidable) (Durand-Lose, 2006b) and second, a singularity can happen at an irrational position (Durand-Lose, 2008a). On the other hand, as long as Turing-computability is involved, rational SM are enough (as shown in Sect. 3). But if analog computations are to be considered, then it is not the case anymore as in Sect. 4.

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

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

Definition 3 The *space-time diagram* issued from an initial configuration c_0 and lasting for T, is a function c from $[0,T]$ to configurations (i.e. a function from $\mathbb{R} \times [0,T]$ to V) such that, $\forall (x,t) \in \mathbb{R} \times [0,T]$:

(i) $\forall t{\in}[0,T]$, $c_t(.)$ is valid, and $c_0(.) = c_0$;

(ii) if $c_t(x){=}\mu \in M$ 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.:

  (a) $\forall t' \in (t_i, t_f)$, $c_{t'}(x + S(\mu)(t'-t)) = \mu$,

  (b) $(\, t_i{=}0$ and $c_0(x_i) = \mu\,)$ or $(\, c_{t_i}(x_i) = \rho^-{\to}\rho^+{\in}R$ and $\mu \in \rho^+\,)$ where $x_i{=}x + S(\mu)(t_i - t)$,

  (c) $(\, t_f{=}T$ and $c_{t_f}(T) = \mu\,)$ or $(\, c_{t_f}(x_f) = \rho^-{\to}\rho^+{\in}R$ and $\mu \in \rho^-\,)$ or $c_{t_f}(x_f){=}❋$ where $x_f{=}x + S(\mu)(t_f - t)$;

(iii) if $c_t(x){=}\rho^-{\to}\rho^+ \in R$ then $\exists \varepsilon$, $0{<}\varepsilon$, $\forall t'{\in}[t{-}\varepsilon, t{+}\varepsilon] \cap [0,T]$, $\forall x'{\in}[x - \varepsilon, x + \varepsilon]$,

  (a) $(x',t') \neq (x,t) \Rightarrow c_{t'}(x') \in \rho^-{\cup}\rho^+ \cup \{\oslash\}$,

  (b) $\forall \mu{\in}M, c_{t'}(x'){=}\mu \Leftrightarrow \vee \begin{cases} \mu \in \rho^- \text{ and } t' < t \text{ and } x' = x + S(\mu)(t'-t) \\ \mu \in \rho^+ \text{ and } t < t' \text{ and } x' = x + S(\mu)(t'-t) \end{cases}$;

  and

(iv) if $c_t(x) = ❋$ then $\forall\varepsilon{>}0$, there are infinitely many collisions in $I^{-}(x,t) \cap \mathbb{R} \times [t{-}\varepsilon, t)$ or infinitely many signals in $[x-\varepsilon, x+\varepsilon] \times [t{-}\varepsilon, t)$.

The definition naturally extends to the case $T = +\infty$. A space-time diagram is *rational* if it is correspond to the one of a rational SM. As long as no singularity is reached, the evolution is deterministic; the space-time diagram only depends on c_0 and the SM.

2.1.1. *Encompassing singularities.*
When a singularity is reached isolated (i.e. there is nothing round it), there are various ways to *continue* the space-time diagram beyond it.

Definition 4 (Schemes to handle isolated singularities) A singularity at (x, t) is *isolated* if there is nothing but void around it in the configuration (i.e. $\exists\varepsilon, \forall x' \in [x-\varepsilon, x+\varepsilon], x' \neq x \Rightarrow c_t(x') = \oslash$). There are various schemes to continue a space-time diagram at an isolated singularity.
(i) *Wall.* It remains there forever;
(ii) *Vanish.* It disappears as if it where on another dimension;
(iii) *Simple trace.* It disappears but sends a μ_s signal; or
(iv) *Conditional traces.* It disappears but sends signals according to which signals join in (i.e. signals that are not interrupted by any collision –not an infinite succession of signals).

In the two last schemes, μ_s or the singularity rules have to be added to the definition of the SM. In such a case, one talks about an *extended SM (ESM)*. Next sections consider ESM and the schemes used are indicated.

The first scheme is not considered here; although it makes sense since singularities generally do not just disappear, in dimension one, it unfortunately produces an unbreakable frontier definitively splitting the configuration in two. The second one is considered in the section on discrete computation while the third is considered in the section on analog computation. The reason in the analog case is that its position is an important piece of information. The last case is also used to consider the case of a singularity happening exactly on a higher level structure (as the one presented below) with nested structures in Sect. 5.

The last two schemes impose a modification of the rules of space-time diagrams. One is to allow a signal to start from a singularity to add to b "or $(\,c_{t_i}(x_i){=}❋ \wedge \mu\ =\ \mu_s\,)$" and to iv something amounting for signals emitted like the rules for collisions (iii) which is not given since it depends on the scheme and for the last scheme of the singularity rules.

In case of non-isolated singularities, the fist two schemes remain possible while the two last ones would make it necessary to define signals with a dimension (which might be not an integer since the accumulation set can be a Cantor set).

Accumulations of second (or more) order can be considered, as long as all the singularities are isolated. In the last case, more distinction could be made according to the level of singularity. As long as finitely many levels are considered or distinguished, the description remains finite.

2.2. *Shrinking and space and time bounding*

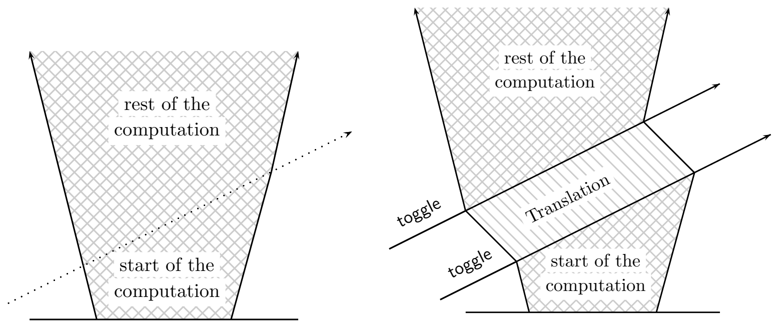
All the constructions work in the following way: starting from a SM, new meta-signals and rules are added in order to generate another SM that works identically with original meta-signals but the new ones provide an extra layer with some kind of meta instructions. For a computation to be altered, extra signals have to be added to the initial configuration.

2.2.1. *Freezing and unfreezing.*

A new meta-signal, toggle, with a speed strictly greater than any present speed, say s_0 is added. If one instance of toggle is set on the left, and is re-generated by every collision, then a succession of toggle (signals) crosses the entire computation. It is the freezing signal. Freezing is done in the following way: each time it meet some signal μ, it replaces it by some frozen counterpart ${}^{\mathsf{F}}\mu$. All these new frozen meta-signals have the same speed, say f_0 (which should be less that s_0 to ensure that they are above the freezing line on the space-time diagram). It might happen that the freezing signal passes exactly on a collision, say ρ. Then the collision is frozen too, i.e. it is replaced by a frozen signal amounting for it, ${}^{\mathsf{F}}\rho$. The signals resulting from the collision are generated when the collision is unfrozen.

The unfreezing scheme is the same, in reverse: signals and collisions replace their frozen counterparts at the passing of another toggle signal.

This scheme is illustrated on Fig. 1. In all the figures, time is elapsing upwards. On the left, one computation is showed unaltered but the intended trace of toggle is indicated with dotted lines. On the right, the computation is frozen, the frozen signals (the ${}^{\mathsf{F}}.$) are left to move for some time and then they are unfrozen by a second toggle. Signal are shifted on one side, they could be shifted on the other side by symmetry, or shifted anywhere (as long as it is in the future) by moving the toggling signals and changing the inner slope.



(a) Regular evolution. (b) Frozen, translated and unfrozen.

Fig. 1. Freezing principle.

The algorithm to modify a SM is given on Fig. 2 where object oriented notations are used. Each time new identities are created, renaming is used if necessary. There is nothing particular about it and every modification works on the same pattern.

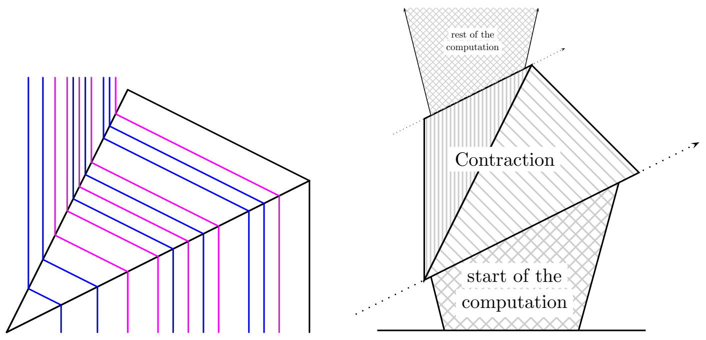
```
Input:
    $M$: signal machine
    $\beta$: real number { speed of the toggle }
    $\theta$: real number { speed of frozen signals }
Assert: ( $S_{max} < \beta$) $\wedge$ ( $\theta < \beta$)
Do:
    { Create the toggle }
 1: toggle $\leftarrow$ $M$.add_new_meta-signal_of_speed( $\beta$ )
    { For the meta-signals }
 2: for each $\mu$ original meta-signal from $M$ do
 3:     ${}^{\mathsf{F}}\mu \leftarrow M$.add_new_meta-signal_of_speed( $\theta$ )
 4:     $M$.add_rule( { toggle, $\mu$ } $\rightarrow$ { toggle, ${}^{\mathsf{F}}\mu$ } )
 5:     $M$.add_rule( { toggle, ${}^{\mathsf{F}}\mu$ } $\rightarrow$ { toggle, $\mu$ } )
 6: end for
    { For the rules }
 7: for each $\rho = \rho^{-} \rightarrow \rho^{+}$ original rule from $M$ do
 8:     ${}^{\mathsf{F}}\rho \leftarrow M$.add_new_meta-signal_of_speed( $\theta$ )
 9:     $M$.add_rule( { toggle } $\cup\ \rho^{-} \rightarrow$ { toggle, ${}^{\mathsf{F}}\rho$ } )
10:     $M$.add_rule( { toggle, ${}^{\mathsf{F}}\rho$ } $\rightarrow$ { toggle } $\cup\ \rho^{+}$ )
11: end for
Output: toggle: meta-signal { the freezing/unfreezing one }
Side effect: New signals and rules added to $M$
```

Fig. 2. Algorithm to add the freezing capability.

2.2.2. *Scaling parallel signals and any computation.*

When signals are parallel, they remain so and do not interact, their structure is quite loose. So that a group of parallel signals can be manipulated quite easily as long as they remain parallel they remain *synchronized.* They just have to be unfrozen with the same slope.

Using a Thales based construction, or prismatic if thought of as light beams, it is easy to scale parallel signals as depicted on Fig. 3(a). The idea is to change the direction twice to be scaled and recover the original direction. The speed of the signals crossing the triangle is carefully selected in order to insure the wanted ratio.



(a) Scaling parallel signals. (b) Freeze, Scale and unfreeze.

Fig. 3. Scaling.

This construction can be set inside the freezing/unfreezing construction. This leads to the scheme of Fig. 3(b). The specially added signal for the structure are only a part of the ones in the next subsection. The algorithms to modify SM are so plain that they are not presented anymore.

2.2.3. *Iterating forever.*

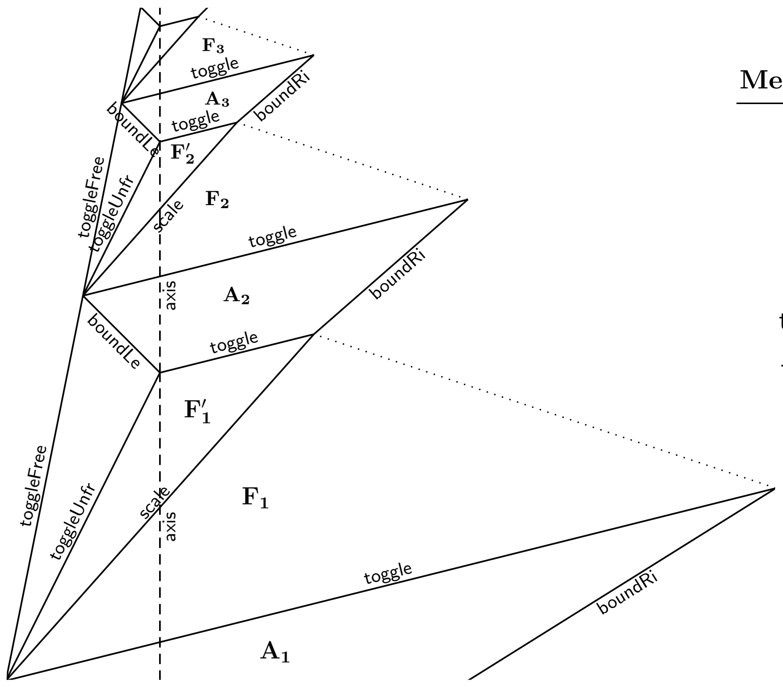
The idea is to iterate *ad infinitum* above construction. More signals have to be added in order to restart the freeze, scale down and unfreeze process on and on. The structure and some basic properties are presented before the embedding of computations.

The structure is presented on Fig. 4. The collision rules are not given because they can be read directly from the figure; for example

$$\{\mathsf{toggleUnfr}, \mathsf{axis}\} \to \{\mathsf{boundRi}, \mathsf{axis}, \mathsf{toggle}\}\ .$$

The dashed axis signals are part of the construction, while the dotted lines do not correspond to any signal and are only there to delimit some regions. Signal axis is dashed because it does not delimit any region.

The toggleUnfr and toggleFree signals are used to start the freezing and unfreezing toggle. The scale signals are used both to change the direction of parallel signals (and achieve scaling down the embedded configuration) and also to start boundRi. The signals boundLe and boundRi are used to bound the potential room used by the embedded computation when active. The



Meta-signal	Speed
boundLe	$-\nu_0$
boundRi	ν_0
scale	$\frac{8}{9}\nu_0$
toggle	$4\nu_0$
toggleUnfr	$\frac{1}{2}\nu_0$
toggleFree	$\frac{1}{5}\nu_0$
axis	0

Fig. 4. iterated scaling.

initial position of axis is at one third of the distance between the four on the left and boundRi.

The construction works because the speed parameter ν_0 used to compute the various speeds is equal to the maximum absolute value of the speeds in the original SM. The speeds given on Fig. 4 are computed such that:

– at each iteration, the structure is scaled by half; and
– the length of the unfreezing toggle signal is one fourth of the preceding freezing one.

The structure is twice as small and twice as fast each time but the initial computation is scaled by one fourth. Relatively to the structure the computation is two times faster each time. This is wanted because not only should the structure collapse in finite time, but meanwhile the embedded computation should have infinite time ahead of it. This is to be understood considering the regions.

The $\mathbf{A_i}$ regions (on Fig. 4) are the ones where the embedded computation is active. The first one is triangular while all the other ones are trapezoidal. The other two regions are triangular. The $\mathbf{F_i}$ regions are the ones where the embedded computation is frozen and signals follow the direction of the dotted line. The $\mathbf{F'_i}$ regions are the ones where the embedded computation is also frozen but signals follow the direction of toggleUnfr. The frontiers between $\mathbf{A_i}$ and $\mathbf{F_i}$ (as well as $\mathbf{F'_i}$ and $\mathbf{A_{i+1}}$) are toggle signals thus the correct freeze (and unfreeze). The frontiers between $\mathbf{F_i}$ and $\mathbf{F'_i}$ are scale signals which correspond to a change of direction of frozen signals.

On the frozen regions $\mathbf{F_i}$, all frozen signals have to be parallel to the dotted line in order that the lower toggle is "mapped bijectively" onto scale. So that their speed is $-\frac{8}{5}\nu_0$. On $\mathbf{F'_i}$ all frozen signals have to be parallel to the scaleLo signal in order that scale is "mapped bijectively" onto the upper toggle. So that their speed is $\frac{1}{2}\nu_0$ (toggleUnfr).

The embedded configuration is scaled by one fourth but the piece is only one half in size of the previous one. Each time the duration, relatively to the original computation, is halved by 2 (for the structure size) but multiplied by 4 for the scale. Altogether, the ratio is 2. So that each time, the elapsing time for the original computation is doubled. This ensures that the original computation is entirely embedded, i.e. has infinite time ahead of it.

Figure 5 shows a simple space-time diagram that is growing on both side on the left and its embedded version on the right. The active part is where the lattice is visible, otherwise it is frozen, shift and scaled.

The location of the singularity can be computed easily as the intersection of two lines (but also as a geometrical sum) and is rational as long as it is stated with rational positions.

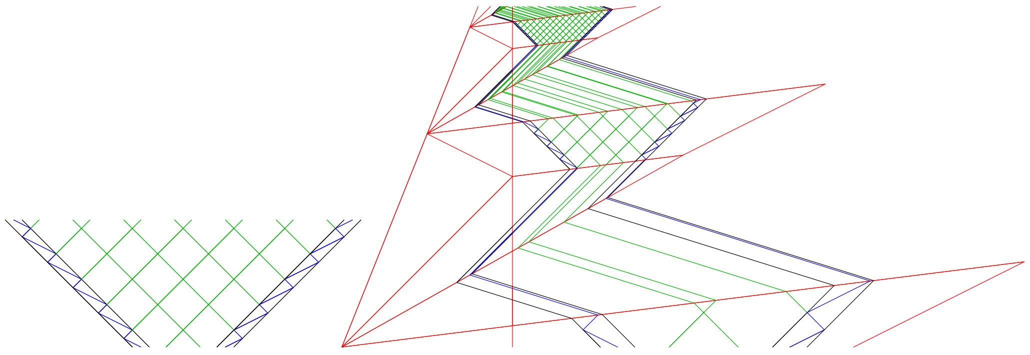

(a) Unaltered (b) Shrunk

Fig. 5. Iterated scaling example.

The shrinking structure serves as a black hole. The configuration embedded inside is trapped. In the following sections, small modifications of already modified SM show how to let some information leave.

3. Discrete computations

Definition 5 A *Turing machine* (TM) is defined by $(Q, q_i, \Gamma, \texttt{\^{}}, \texttt{\#}, \delta)$ where Q is a finite set of *states*, q_i is the *initial state*, Γ is a finite set of *symbols*, $\texttt{\^{}}$, *tape head*, and #, *blank*, are two distinguished symbols, and $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ is the *transition function.*

A *TM-configuration* is defined by (q, w, i) such that q is a state, w is a finite sequence of symbols –or word over Γ– and i is a natural number. The automaton is in state q, w (completed by #'s) is written on the tape and the head is over the ith cell of the tape. The TM is *self-delimiting* when there is only one ^ on the tape which is written on the first cell, and there is nothing but # right of the first #. If it is not the case, the TM is in an illegal configuration.

The *next configuration* is defined by the transition function as follows. If $\delta(q, w_i) = (r, \mathtt{a}, \rightarrow)$ then the next configuration is defined by $(r, w', i+1)$ where w' is obtained from w by replacing the ith symbol by a. If $\delta(q, w_i) = (r, \mathtt{a}, \leftarrow)$ then, if $i = 0$ then the TM stops otherwise the next configuration is defined by $(r, w', i-1)$.

The TM *computes* in the following way: some input (without ^ and #) is written on the tape preceded by ^ and followed by potentially infinitely many #. The result of the computation, if any, is what is written on the tape (^ and all # are discarded) when the TM stops.

The TM halts normally when the head tries to leave the tape. For example, the machine defined on Fig. 6(b), computes on Fig. 6(a) with the entry ab. The output is bbab. If the computation does not halt or enters an illegal configuration, the output is undefined.

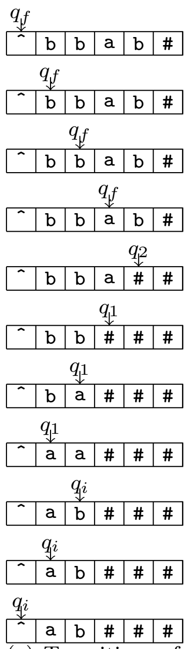
When a TM is used for a decision (yes/no output), the states are partitioned into accepting and refusing one. The answer is then given by the state in which it halts.

3.1. *Turing-machine simulation*

The simulation goes as follows: there are finitely many parallel signals encoding the content of the tape in-between which zigzags a signal mimicking the movement of the head and encoding the state. This is depicted with an example on Fig. 6(d).

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

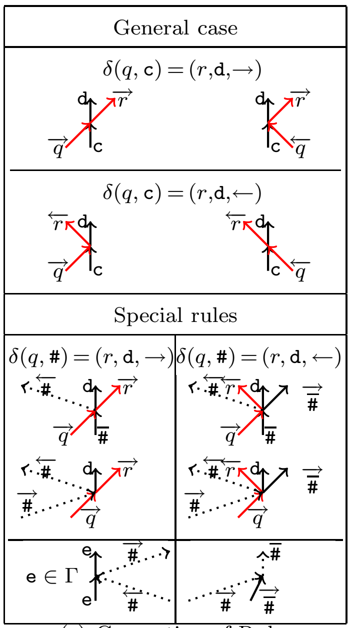
– one *symbol signals* for each value in Γ, with null speed, to encode the cells of the tape;



(a) Transitions of the TM.

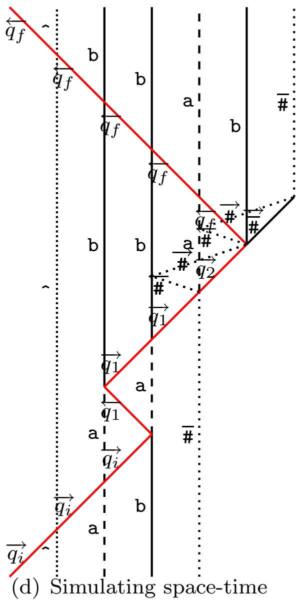
δ	ˆ	a	b	#
q_i	q_i,ˆ,$\longrightarrow$	q_i,a,$\longrightarrow$	q_1,a,$\longleftarrow$	-
q_1	-	q_1,b,$\longrightarrow$	-	q_2,a,$\longrightarrow$
q_2	-	-	-	q_f,b,$\longleftarrow$
q_f	q_f,ˆ,$\longleftarrow$	q_f,a,$\longleftarrow$	q_f,b,$\longleftarrow$	-

(b) Transition table of the TM.



General case
$\delta(q,\texttt{c}) = (r,\texttt{d},\longrightarrow)$
$\delta(q,\texttt{c}) = (r,\texttt{d},\longleftarrow)$
Special rules
$\delta(q,\texttt{\#}) = (r,\texttt{d},\longrightarrow)$
$\delta(q,\texttt{\#}) = (r,\texttt{d},\longleftarrow)$
$\texttt{e} \in \Gamma$

(c) Generation of Rules.



(d) Simulating space-time diagram.

Fig. 6. Example of a TM computation and its simulation by a SM.

– $\overrightarrow{q}$ (of speed 1) and $\overleftarrow{q}$ (of speed -1) *head signals* for each state q, to encode the state and the location and movement of the head;
– $\overline{\texttt{\#}}$ (of null speed), $\overleftarrow{\texttt{\#}}$ (of speed -3), $\overrightarrow{\texttt{\#}}$ (of speed 3), and $\overrightarrow{\overrightarrow{\texttt{\#}}}$ (of speed 1) which are used to denote the end of the tape and manage the enlargement of the tape.

The initial SM-configuration for a TM-computation on the entry $w = w_1 w_2 \ldots w_k$ is generated by putting one $\overrightarrow{q_i}$-signal at position -1, one ˆ-signal at position 0, one w_i-signal at position i ($1 \leq i \leq k$), and one $\overline{\texttt{\#}}$-signal at position $k+1$,

A SM-configuration encodes a TM configuration if it corresponds to the same sequence of symbols (up to extra #'s at the end) closed by a $\overline{\texttt{\#}}$ with an extra signal encoding the state and moving to meet the signal corresponding to the position of the head.

The collisions 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 move to its next position on the tape. When it meets the right special signal, the configuration is automatically enlarge.

The generated collision rules are given on Fig. 6(c). When a rule is not defined, the signal just cross unaffected. From top to bottom, they correspond to the following cases. For each TM-transition, two collision rules are generated, they correspond to the cases where the head would come from the left or from the right. The special rules are made in order to ensure a correct enlargement of the configuration in the case a head signal meet $\overline{\#}$, the signal marking the last cell. In such a case, two things have to be done: normally do the TM-transition (as if it were a # signal) and enlarge the configuration. The latter means generate one $\overline{\#}$ one position on the right if the head goes left (left side rules). If the head goes right (right side rules), then it should be taken care that the head meets something on the right (lower row rules). This is illustrated in the middle of Fig. 6(d). In each first case, a $\overleftarrow{\#}$ is sent on the left. It bounces on the symbol signal on the left (bottom rule of Fig. 6(c)) and is replaced by $\overrightarrow{\#}$, cross whatever signal present to get where would be the head if it would have gone right. If indeed the head went right, it is met and the TM-transition is done and the configuration enlargement starts again. If it is not the case (left rules of Fig. 6(c)), then $\overrightarrow{\overline{\#}}$ is sent in order to meet $\overrightarrow{\#}$ and place the new $\overline{\#}$. Signals $\overleftarrow{\#}$ and $\overrightarrow{\#}$ are three time faster in order to ensure that the meeting happens exactly where the next symbol signal should have been.

At the end of the TM-computation simulation, the signal encoding the state goes out on the left. The process is robust, the positions does not have to be exact as long as the order of the signals is preserved. It can be implemented with a rational SM.

3.2. *Infinite Turing computation in bounded time*

The construction described in Sect. 2 rightfully apply to the above construction. It is considered that the singularity vanishes leaving no signal (scheme ii of Def. 4). The infinite acceleration is granted. But the information leaving the black hole is missing. The ESM has to undergo some modifications. The first one is to generate the escaping signal, the second to retrieve it.

The result of a deciding TM is given by the new state whenever it performs a transition on ˆ and sends the head on the left. These transitions are clearly identified in the transition table of the TM. So there are also clearly identified in the rules of the simulating ESM.

Once the shrinking mechanics have been added, the rules can be changed so that to generate new signals amounting for accept and refuse. These signals are unaffected by other signals including the structure ones. Their speed is -1 so that they exit on the left.

The last thing is to add two signals orizonLe and orizonRi of speed ($\frac{1}{5}\nu_0$ and $\frac{-16}{15}\nu_0$) on the left and right of the shrinking structure. Their speeds ensure that they remain at constant distance from the structure. The way these signals are intended to work is displayed on Fig. 7

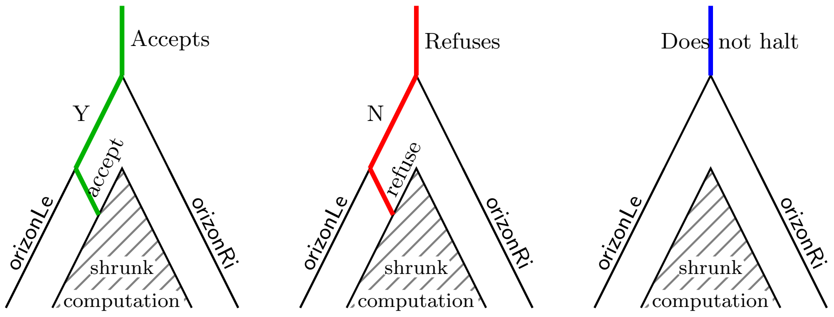


Fig. 7. Encapsulating the shrinking structure to retrieve any leaving signal.

The desired effect is achieved. Recursively enumerable problems (Σ^0_1) can be decided and computation can carry on after the answer is known.

4. Analog computations

In this section, analog computations in the understanding of the Blum, Shub and Smale model (Blum et al., 1989, 1998) (BSS for short) are considered. After briefly presenting the BSS model on $\mathbb{R}$ and its linear restriction, results on its simulation in AGC are recalled before the shrinking construction is applied.

4.1. *BSS Model*

BSS machines (BM) operate like TM on unbounded arrays/tapes. Each cell of the array hold a real number and all the computations are made with exact precision. The BM has a head allowing to access a finite portion of the tape called a *window*. It has finitely many states and to each one corresponds an instruction among the following:

(i) compute a polynomial function of the window and place the result in it;

(ii) test for the sign of a given cell in the window and branch accordingly;
(iii) shift the window one cell on the left or on the right; and
(iv) halt.

The window is shifted by one cell so that consecutive windows overlaps. This is used to carry information around, since the BM has no real value storage on its own.

A BSS machine is *linear* if instruction i is replaced by “compute a linear function [...]”. Thus multiplication is allowed only if it is by a constant.

Like for TM, the input (resp. output) is the content of the array when the BM is started (resp. halts). Comparing to TM, trying to leave the array on the left is an error but halting states are provided. If BM are used for decision, then there are distinct halting states for acceptance and refusal.

The simulation is not presented since details are not relevant to the following. Only the encoding of real values and two main results are given.

A real number is encoded as the distance between two signals. But as it has already been guessed by the reader, time and distance are very relative concepts. The distance between two scale signals is thus used as a scale. The same scale is used for all the real numbers so that each one can be encoded by a just pair base and value (or just nul for 0 since superposition of signals is not allowed). This is illustrated for various values on Fig. 8.

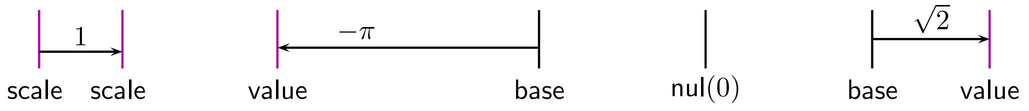


Fig. 8. Encoding: scale and values $-\pi$, 0 and $\sqrt{2}$.

Theorem 6 ((Durand-Lose, 2007)) *AGC with finitely many signals and no singularity is equivalent to the linear BSS model.*

Simulations have been established in both directions. In the constructions for SM simulation, the encoding pairs are manipulated so that they are clearly identified and everything is scaled down whenever they might mess up.

Theorem 7 ((Durand-Lose, 2008a)) *With the Simple trace scheme (iii of Def. 4) for singularities, AGC is able to carry out any multiplication achieving the simulation of the whole BSS.*

In this case AGC is strictly more powerful than classical BSS since square root can be computed.

The meaning of the shrinking structure on a BM simulation is considered before entering into technical issues.

4.2.1. *Interpretation.*

Assuming that everything works well as expected, the ESM can be made so that if the computation stops then some information leaves the singularity. But what kind of information? Usually it is considered that only finitely many values can be used***, not countably many, not continuum many! Mechanisms as in Subsect. 3.2 can be used to send a few bits, but it is not clear how to send four signals (to encode a real number) ensuring that they are all at the same scale (i.e. are emitted from the same active region).

With the same construction as previously, using a universal BM, the BSS halting problem, can be decided.

Instance $\textsc{Halt}_{\mathrm{BSS}}$
- $\overrightarrow{X}$, $\overrightarrow{Y}$: vectors of real numbers

Question
Does the BM represented by $\overrightarrow{X}$ stop on entry $\overrightarrow{Y}$?

Let us consider another example. BSS machines can simulate any TM and any recursive function. There is a BM with one entry that tries all the rational numbers sequentially and stops if it is equal to the entry. Since halting can be decided, the characteristic function of $\mathbb{Q}$ is computable (which is not usually the case).

The general form of decision problems that can be solved is of the form $\exists n \in \mathbb{N}, \phi(n, \overrightarrow{X})$ where ϕ is decision BM that stops for all entries. This definition is the same as the one for classical recursion except that total recursive predicate is replaced by total BSS predicate. The quantification is on $\mathbb{N}$ (that can encode $\mathbb{Q}$ or many other countable sets) but not on $\mathbb{R}$. This does not correspond to the first and second level of an analytical hierarchy but an arithmetical one. It corresponds to BCSS-Σ_1 in Ziegler (2007). Please note that this is only a lower bound on the computing capability.

If one wants the analog output if the computation stops and just the information that is does not stop otherwise, it uses a singularity to know whether the computation stops and if it is the case, then it starts the computation normally to get the result in finite time.

***For example think about the so-called blue-shift effect and how it might be countered (Németi and Dávid, 2006, Sub. 5.3.1).

If one would like to have the limit of say first cell of the array if any, not only does the shrinking might turn the scale and the distance between encoding signals to zero but moreover, if the BM-computation uses bigger and bigger real numbers, its accumulated re-scaling also turns them to zero. So that one would have to find how to generate the value in a way that does not lead to zero divided by zero and neither prevents the shrinking.

This has been achieved to get internal multiplication out of linear BSS: three of the four real numbers encoding signals are constant outside of the singularity and the singularity happens at the correct position for the last signal. A general scheme still have to be found.

4.2.2. *Technical issues.*

In each active region, the configuration only undergoes a regular scaling. Up to scaling, the active regions perfectly assemble. The computation is exact and the real values are preserved even though the encoding can be split between various active regions (retrieving the value is not obvious).

The difference from the discrete case is that singularities are already used for multiplication. These are handled with the simple trace scheme (iii of Def. 4).

With the structure, both type of singularity should be distinguished particularly because the singularity for multiplication could happen exactly on toggle so that with the simple trace scheme, the structure would be damaged. In this case, one extra frozen meta-signal must exist to encode the singularity as well as a toggle for the structure. The case iv –which is more general– is used. Since the speed of toggle is greater than the any of the other signal present for the multiplication, it gets straight into the singularity without getting involved in any collision and thus distinguishes between the singularities.

If there are infinitely many multiplications, then the singularity is of second order. This is not a problem for the structure nor the definition of space-time diagrams. Rule iv of Def. 3 only asks for infinitely many collisions or signals which is also ensured by the accumulating multiplications.

5. Nested singularities

Previous Section ends with a second order singularity. It is possible to built higher order singularities.

Section 2.2 explains how to shrink any space-time diagram that has no singularity. It is quite easy to set signals to automatically start a singularity (in the spirit of what is done in Durand-Lose (2006b)) and according to the result to start another. There can be finitely or countably many isolated singularities (there is a rational location in each open).

The interesting part is when singularities are nested inside another one. In Sect. 4.2.2, it is explained how, with the conditional traces scheme (iv of Def. 4), to handle sub-singularities. The second order structure works because it is built on top of the previous one (after renaming the meta-signals to avoid any conflict). So that the outer one can handle the inner ones while the latter proceed on their own.

It is thus natural to reiterate the construction. This can be done safely a finite number of times (more would yield an infinite number of signals). Inside any structure, it is only possible to start a lesser level structure so that the first one is the topmost and that the number of nested levels inside is bounded by construction of the ESM.

For discrete computations, in the setting presented in Subsect. 3.2, singularities are only used to decide and leave no trace (ii of Def. 4). But since a singularity could happen on a toggle of a higher level structure, for the same reason as before, the conditional traces scheme has to be used. For analog computations, this scheme is already used. It is thus the natural scheme to use in both cases.

In the discrete case, like for the SAD computers of Hogarth (1994, 2004), each order brings the capability to decide an extra level of the arithmetical hierarchy by deciding an extra alternating quantifier (on $\mathbb{N}$). As shown, first order decides Σ^0_1. If nth order singularity allow to decided Σ^0_n, then it can be used as oracles inside the top level of a $n+1$th order singularity.

In the analog case, each level of singularity identically allows to decide an extra alternation of quantifiers (on $\mathbb{N}$) and to climb the BSS arithmetical hierarchy. This is not a quantification on $\mathbb{R}$, this is not an algebraic hierarchy.

6. Conclusion

Abstract geometrical computation provides a setting where Black hole-like computation can be implemented: a finite portion is infinitely accelerated while a computation can continue outside and get a finite insight on what happens inside. For SAD computers, the (nested) black holes have to be found while in AGC the computation construct them (signals are the "fabric" of space and time) but the maximum level of nesting is limited by the construction of the ESM.

In our constructions, there are two levels of space-time: one absolute of the SM and one due to the embedding inside a shrinking structure; singularities are created by the fabric of signals. The following have been proved.

Theorem 8 *With proper handling of accumulation, for any level of the arithmetic hierarchy, a rational ESM can be built to decide it and for any level of the BSS arithmetic hierarchy, an ESM can be built to decide it.*

These are just decisions, but they can be used as sub-computations in any computation.

In the discrete case, the ESM and space-time-diagrams remain rational (irrational coordinate can only be generated as singularity location, but, by construction, the singularities built happen only at rational position).

A general construction that would allow all orders with the same ESM is still lacking. One important achievement would be to provide a system for transfinite order singularity together with the possibility to nest inside recursive computations. Considering spaces with more dimensions, could help getting hyper-arithmetic (Bournez, 1999a,b). But our scheme is only in one dimension, signals with dimension 1 or more could be used to shrunk in such spaces.

Comparing to infinite time Turing machine (Hamkins and Lewis, 2000; Hamkins, 2002, 2007), the content of the tape is lost since it all accumulates in one point. So a way to preserve the resulting tape and a limit mechanism still have to be found to relate to infinite time Turing machines and recursive analysis (Weihrauch, 2000). The same problem arise with the analog counterpart; providing limits would link to the hierarchy of Chadzelek and Hotz (1999).

References

Lenore Blum, Michael Shub, and Steve Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc.*, 21(1):1–46, 1989.

Lenore Blum, Felipe Cucker, Michael Shub, and Steve Smale. *Complexity and real computation.* Springer, New York, 1998.

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

Olivier Bournez. Some bounds on the computational power of piecewise constant derivative systems. *Theory Comput. Syst.*, 32(1):35–67, 1999b.

Olivier Bournez and Emmanuel Hainry. On the computational capabilities of several models. In Jérôme Durand-Lose and Maurice Margenstern, editors, *Machines, Computations, and Universality, MCU '07*, volume 4664 of *LNCS*, pages 12–23. Springer, 2007.

Thomas Chadzelek and Günter Hotz. Analytic machines. *Theoret. Comp. Sci.*, 219(1-2):151–167, 1999.

B. Jack Copeland. Hypercomputation. *Minds & Machines*, 12(4):461–502, 2002.

Jérôme Durand-Lose. Abstract geometrical computation 1: embedding black hole computations with rational numbers. *Fund. Inf.*, 74(4):491–510, 2006a.

Jérôme Durand-Lose. Forcasting black holes in abstract geometrical computation is highly unpredictable. In J.-Y. Cai, S. B. Cooper, and A. Li, editors, *Theory and Appliacations of Models of Computations (TAMC '06)*, number 3959 in LNCS, pages 644–653. Springer, 2006b.

Jérôme Durand-Lose. Abstract geometrical computation and the linear Blum, Shub and Smale model. In S.B. Cooper, B. Löwe, and A. Sorbi, editors, *Computation and Logic in the Real World, 3rd Conf. Computability in Europe (CiE '07)*, number 4497 in LNCS, pages 238–247. Springer, 2007.

Jérôme Durand-Lose. Abstract geometrical computation with accumulations: beyond the Blum, Shub and Smale model. In Arnold Beckmann, Costas Dimitracopoulos, and Benedikt Löwe, editors, *Logic adn Theory of Algorithms, CiE 2008 (abstracts and extended abstracts of unpublished papers)*, pages 107–116. University of Athens, 2008a.

Jérôme Durand-Lose. The signal point of view: from cellular automata to signal machines. In Bruno Durand, editor, *Journées Automates cellulaires (JAC '08)*, pages 238–249, 2008b. URL http://www.lif.univ-mrs.fr/jac/.

Gábor Etesi and Istvan Németi. Non-Turing computations via Malament-Hogarth space-times. *Int. J. Theor. Phys.*, 41(2):341–370, 2002. gr-qc/0104023.

Joel David Hamkins. Infinite time Turing machines: supertask computation. *Minds & Machines*, 12(4):521–539, 2002. arXiv:math.LO/0212047.

Joel David Hamkins. A survey of infinite time Turing machines. In J. Durand-Lose and M. Margenstern, editors, *Machines, Computations and Universality (MCA '07)*, number 4664 in LNCS, pages 62–71. Springer, 2007.

Joel David Hamkins and Andy Lewis. Infinite time Turing machines. *J. Symb. Log.*, 65(2):567–604, 2000. arXiv:math.LO/9808093.

Mark L. Hogarth. Deciding arithmetic using SAD computers. *Brit. J. Philos. Sci.*, 55:681–691, 2004.

Mark L. Hogarth. Non-Turing computers and non-Turing computability. In *Biennial Meeting of the Philosophy of Science Association*, pages 126–138, 1994.

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

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

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

Akitoshi Kawamura. Type-2 computability and Moore's recursive functions. *Electr. Notes Theor. Comput. Sci.*, 120:83–95, 2005.

Jerzy Mycka. Infinite limits and $\mathbb{R}$-recursive functions. *Acta Cybern.*, 16 (1):83–91, 2003a.

Jerzy Mycka. μ-recursion and infinite limits. *Theoret. Comp. Sci.*, 302(1-3): 123–133, 2003b.

Jerzy Mycka. Analog computation beyond the Turing limit. *Appl. Math. Comput.*, 178(1):103–117, 2006.

Jerzy Mycka and José Félix Costa. A new conceptual framework for analog computation. *Theoret. Comp. Sci.*, 374(1-3):277–290, 2007.

István Németi and Hajnal Andréka. Can general relativistic computers break the Turing barrier? In Arnold Beckmann, Ulrich Berger, Benedikt Löwe, and John V. Tucker, editors, *Logical Approaches to Computational Barriers, 2nd Conf. on Computability in Europe, CiE '06*, volume 3988 of *LNCS*, pages 398–412. Springer, 2006.

István Németi and Gyula Dávid. Relativistic computers and the Turing barrier. *Appl. Math. Comput.*, 178(1):118–142, 2006.

Klaus Weihrauch. *Introduction to computable analysis.* Texts in Theoretical computer science. Springer, Berlin, 2000.

Philip D. Welch. The extentent of computation in malament-hogarth space-time. *Brit. J. Philos. Sci.*, 2006. to appear.

Martin Ziegler. (Short) Survey of real hypercomputation. In S. Barry Cooper, Benedikt Löwe, and Andrea Sorbi, editors, *Computation and Logic in the Real World, 3rd Conf. Computability in Europe, CiE '07*, volume 4497 of *LNCS*, pages 809–824. Springer, 2007.