Definability by Horn Formulas and Linear Time on Cellular Automata*

Nicolas Bacquey[1], Etienne Grandjean[2], and Frédéric Olive[3]

1 **INRIA Lille, Université de Lille, CRIStAL, Villeneuve d'Ascq, France**
`nicolas.bacquey@inria.fr`
2 **Normandie Université, ENSICAEN, CNRS, GREYC, Caen, France**
`etienne.grandjean@unicaen.fr`
3 **Aix Marseille Université, CNRS, LIF UMR 7279, Marseille, France**
`frederic.olive@lif.univ-mrs.fr`

Abstract

We establish an *exact* logical characterization of linear time complexity of cellular automata of dimension d, for any fixed d: a set of pictures of dimension d belongs to this complexity class *iff* it is definable in existential second-order logic restricted to *monotonic* Horn formulas with built-in successor function and $d+1$ first-order variables. This logical characterization is optimal modulo an open problem in parallel complexity. Furthermore, its proof provides a systematic method for transforming an inductive formula defining some problem into a cellular automaton that computes it in linear time.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.3 Complexity Measures and Classes, F.4.1 Mathematical Logic, F.4.3 Formal Languages

Keywords and phrases Picture languages, linear time, cellular automata of any dimension, local induction, descriptive complexity, second-order logic, Horn formulas, logic programming

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.99

1 Introduction

Descriptive complexity provides machine-independent views of complexity classes. Typically, Fagin's Theorem [5] characterizes **NP** as the class of problems definable in existential second-order logic (ESO). Similarly, Immerman-Vardi's Theorem [15] and Grädel's Theorem [8, 9] characterize the class **P** by first-order logic plus least fixed-point, and second-order logic restricted to Horn formulas, respectively. The link between *computational* and *descriptive* complexity can be made as *tight* as possible, i.e. linear time or time $O(n^d)$, for a fixed integer d, can be *exactly* characterized [20, 14, 17, 11]. Two of the present authors have proved in [12, 13] that a problem is recognized in linear time on a non-deterministic cellular automaton of dimension d iff it is definable in ESO logic with built-in successor and $d+1$ first-order variables. Is there such a natural characterization in logic for the more interesting deterministic case? This question motivates the present paper.

A number of algorithmic problems (linear context-free language recognizability, product of integers, product of matrices, sorting...) are computable in linear time on cellular automata of appropriate dimension. For each such problem, the literature describes the algorithm of the cellular automaton in an informal way [2, 16]. In parallel computational models,

* This work was partially supported by ANR AGGREG.

algorithms are often difficult to design. However, the problems they solve can often be simply defined inductively. For instance, the product of two integers in binary notation is inductively defined by the classical Horner rule.

The first contribution of this paper is the observation that those inductive processes are naturally formalized by Horn formulas [9]. As our second and main contribution, we notice that for every concrete problem defined by a Horn formula with $d+1$ first-order variables ($d \geq 1$), this inductive computation by Horn rules has a precise *geometrical* characterization: It can be modeled as the displacement of a d-dimensional hyperplane H along some fixed line D in a space of dimension $d+1$. Provided we interpret the line D as a temporal axis, the parallel displacement of H with respect to D coincides with a computation of a d-dimensional cellular automaton. The converse is obvious: a d-dimensional cellular automaton computation can be regarded as the parallel displacement of a d-dimensional hyperplane – its set of cells – along the time axis.

In the next section, a logic is designed which captures these inductive behaviors (see Def. 14). Roughly speaking, it is obtained from the logic ESO-HORN taylored by Grädel to characterize $\mathbf{P}$, by restricting both the number of first-order variables and the arity of second-order predicate symbols. Besides, it includes an additional restriction – the 'monotonicity condition' – that reflects the geometrical consideration above-mentioned. We denote this logic by mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$.

Now we can quote the main result of the paper (Thm. 15): a set L of d-pictures can be decided in linear time by a deterministic cellular automaton – written $L \in \mathbf{DLIN}_{\text{ca}}^d$ – if, and only if, it can be expressed in mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$. For short:

$$\mathbf{DLIN}_{\text{ca}}^d = \text{mon-ESO-HORN}^d(\forall^{d+1}, \text{arity}^{\,d+1}). \tag{1}$$

A noticeable interest of this result is the constructive method of its proof. In order to design a cellular automaton that computes a problem in linear time, one has to define inductively the problem with a monotonic Horn formula. The normalized form of the formula is automatically obtained: this *is* the program of the cellular automaton[1].

The paper is structured as follows: The next section collects the preliminary definitions and gives a precise statement of our main result. In Sec. 3, we establish the left-to-right inclusion of the identity displayed in (1). The rest of the paper is devoted to the converse inclusion, whose proof is far more involved. In Sec. 4 we build a monotonic Horn formula expressing the language of palindromes (a "toy" example) and deduce from it a cellular automaton that recognizes this language in linear time. Sec. 5 generalizes this construction to any problem defined in mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$, thus completing the proof of (1). In Sec. 6, we conclude by arguing for the optimality of our result.

2 Preliminaries

2.1 Cellular automata, picture languages, linear time

We use the standard terminology of cellular automata as presented in [10].

▶ **Definition 1.** A *cellular automaton of dimension d* (d-CA or CA, for short) is a quadruple $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$, where $d \in \mathbb{N}$ is the *dimension* of the automaton, $\mathcal{Q}$ is a finite set whose elements are called *states* , $\mathcal{N}$ is a finite subset of $\mathbb{Z}^d$ called the *neighborhood* of the automaton, and $\delta : \mathcal{Q}^{\mathcal{N}} \to \mathcal{Q}$ is the *local transition function* of the automaton.

[1] For lack of space, this paper gives only one example of this method on a "toy" problem.

A d-CA acts on a grid of dimension d:

▶ **Definition 2.** A *d-dimensional configuration* $\mathcal{C}$ over the set of states $\mathcal{Q}$ is a mapping from $\mathbb{Z}^d$ to $\mathcal{Q}$. The elements of $\mathbb{Z}^d$ will be referred to as *cells* .

▶ **Definition 3.** Given a cellular automaton $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$, a configuration $\mathcal{C}$ and a cell $c \in \mathbb{Z}^d$, we call *neighborhood of c in $\mathcal{C}$* the mapping $\mathcal{N}_{\mathcal{C}}(c) : \mathcal{N} \to \mathcal{Q}$ defined by $\mathcal{N}_{\mathcal{C}}(c)(v) = \mathcal{C}(c + v)$.

From the local transition function δ of $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$, we can define the *global transition function* of the automaton $\Delta : \mathcal{Q}^{\mathbb{Z}^d} \to \mathcal{Q}^{\mathbb{Z}^d}$ obtained by applying the local rule on each cell, that means $\Delta(\mathcal{C})(c) = \delta(\mathcal{N}_{\mathcal{C}}(c)) = \delta((\mathcal{C}(c + v))_{v\in\mathcal{N}})$, for each cell c.

One identifies the CA $\mathcal{A}$ with its global rule: $\mathcal{A}(\mathcal{C}) = \Delta(\mathcal{C})$ is the image of a configuration $\mathcal{C}$ by the action of $\mathcal{A}$. Moreover, $\mathcal{A}^t(\mathcal{C})$ is the configuration resulting from applying t times the global rule of $\mathcal{A}$ from the initial configuration $\mathcal{C}$.

▶ **Definition 4.** For a given cellular automaton: a state q is *permanent* if a cell in state q remains in this state *regardless* of the states of its neighbors; a state q is *quiescent* if a cell in state q remains in this state if *all* its neighbors are also in state q.

The natural inputs of cellular automata of dimension d are *d-pictures*:

▶ **Definition 5.** Let Σ be a finite alphabet. For integers $d, n \geq 1$, a *picture of dimension d* (*d-picture*) and *side n over* Σ is a mapping $p : [\![1, n]\!]^d \to \Sigma$. We denote by $\Sigma^{(d)}$ the set of d-pictures over Σ. Any subset of $\Sigma^{(d)}$ is called a *d-picture language* over Σ.

▶ Remark. d-picture languages can capture a wide variety of decision problems if the alphabet Σ is sufficiently expressive. For instance, the product problem of boolean square matrices is a 2-picture problem over the three-part alphabet $\Sigma = \{0, 1\}^3$ that consists of square pictures M such that the projection of M over the last part of the alphabet is equal to the product of its projections over the first two parts.

▶ **Definition 6.** Given a picture $p : [\![1, n]\!]^d \to \Sigma$, we define the *picture configuration* associated with p with *permanent* or *quiescent* state[2] $q_0 \notin \Sigma$ as the function $\mathcal{C}_{p,q_0} : \mathbb{Z}^d \to \Sigma \cup \{q_0\}$ such that $\mathcal{C}_{p,q_0}(\mathbf{x}) = p(\mathbf{x})$ if $\mathbf{x} \in [\![1, n]\!]^d$ and $\mathcal{C}_{p,q_0}(\mathbf{x}) = q_0$ otherwise.

▶ **Definition 7.** Given a d-picture language $L \subseteq \Sigma^d$, we say that a cellular automaton $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$ such that $\Sigma \subseteq \mathcal{Q}$ with permanent states q_a and q_r (accepting and rejecting states) *recognizes* L with permanent state (quiescent state, respectively) $q_0 \in \mathcal{Q}\backslash(\Sigma\cup\{q_a, q_r\})$ *in time* $\tau : \mathbb{N} \to \mathbb{N}$ (for short, $\tau(n)$) if for any picture $p : [\![1, n]\!]^d \to \Sigma$, starting from the configuration $\mathcal{C}_{p,q_0}$ at time 0, the state of cell $\mathbf{n} = (n, \dots, n)$ of $\mathcal{A}$, called the *reference cell*, is q_a or q_r at time $\tau(n)$ with $\mathcal{A}^{\tau(n)}(\mathcal{C}_{p,q_0})(\mathbf{n}) = q_a$ if $p \in L$ and $\mathcal{A}^{\tau(n)}(\mathcal{C}_{p,q_0})(\mathbf{n}) = q_r$ if $p \notin L$.

▶ **Definition 8.** For $d \geq 1$, we call $\mathbf{DLIN}_{ca}^d$ the class of d-picture problems L for which there exist a d-CA $\mathcal{A}$ with quiescent state q_0 and a function $\tau(n) = O(n)$ such that L can be recognized by $\mathcal{A}$ in time $\tau(n)$. Such a problem is said to be *recognizable in linear time* .

The class $\mathbf{DLIN}_{\mathrm{ca}}^d$ is very robust: it is closed under many changes: neighborhoods, precise time/space bounds, input presentation, etc. The proof of the first part of our main result

[2] The condition that each cell outside the input domain $[\![1, n]\!]^d$ remains in a permanent state (resp. quiescent state) q_0 means that the computation space is exactly the set of input cells (resp. is not bounded).

will use the following restrictive characterization which is a consequence of a general linear acceleration theorem[3] (see *e.g.* [10, 19]).

▶ **Lemma 9** ([10]). $\mathbf{DLIN}^d_{ca}$ *is the class of d-picture problems that can be recognized in time* $n-1$ *by a d-CA of neighborhood* $\mathcal{N}_2 = \{-2,-1,0,1,2\}^d$ *with* permanent *state* q_0.

2.2 Picture structures and monotonic Horn formulas

The local nature of cellular automata acting on pictures is captured by logical formulas acting on first-order structures, the so-called *picture structures*, that represent these pictures. Before defining picture structures, let us detail their signatures. Given a dimension $d \geq 1$ and k alphabets $\Sigma_1, \ldots, \Sigma_k$, we denote by $\mathsf{sg}(d; \Sigma_1, \ldots, \Sigma_k)$ the signature below:

$$\mathsf{sg}(d; \Sigma_1, \ldots, \Sigma_k) = \{(Q^1_s)_{s\in\Sigma_1}, \ldots, (Q^k_s)_{s\in\Sigma_k}, \mathsf{min}, \mathsf{max}, \mathsf{suc}, \mathsf{pred}\}.$$

Here, each Q^i_s is a d-ary relation symbol, min and max are unary relation symbols, and suc and pred are unary function symbols.

▶ **Definition 10.** Let $p_1, \ldots, p_k$ be pictures of respective alphabets $\Sigma_1, \ldots, \Sigma_k$. We assume that the p_i's have the *same dimension* d and the *same side* n. The *picture structure* of the k-tuple $(p_1, \ldots, p_k)$ is the structure of signature $\mathsf{sg}(d; \Sigma_1, \ldots, \Sigma_k)$ defined as follows:

$$\mathcal{S}(p_1, \ldots, p_k) = \langle [\![1, n]\!], (Q^1_s)_{s\in\Sigma_1}, \ldots, (Q^k_s)_{s\in\Sigma_k}, \mathsf{min}, \mathsf{max}, \mathsf{suc}, \mathsf{pred}\rangle.$$

Symbols of $\mathsf{sg}(d; \Sigma_1, \ldots, \Sigma_k)$ are interpreted on $\mathcal{S}(p_1, \ldots, p_k)$ as follows, where we denote the same way a symbol and its interpretation:

- each Q^i_s is the set of cells of p_i labelled by s. Formally: $Q^i_s = \{a \in [\![1, n]\!]^d : p_i(a) = s\}$;
- min and max are the singleton sets $\{1\}$ and $\{n\}$, respectively;
- suc and pred are the successor and predecessor functions: that is $\mathsf{suc}(n) = n$ and $\mathsf{suc}(a) = a + 1$ for $a \in [\![1, n-1]\!]$; $\mathsf{pred}(1) = 1$ and $\mathsf{pred}(a) = a - 1$ for $a \in [\![2, n]\!]$.

In the following, we will freely use the natural notation $x + i$, for any fixed integer $i \in \mathbb{Z}$. It abbreviates $\mathsf{suc}^i(x)$ if $i > 0$, and $\mathsf{pred}^{-i}(x)$ if $i < 0$. For $i = 0$, it represents x.

We will use the usual definitions and notations in logic (see [4, 18, 9]). All formulas considered hereafter belong to *existential second-order logic*. More precisely, we shall focus on the following logic:

▶ **Definition 11.** $\mathrm{ESO}^d(\forall^{d+1}, \mathrm{arity}^{d+1})$ is the class of formulas of the form $\exists \mathbf{R} \forall \mathbf{x} \psi$, where $\mathbf{R} = (R_1, \ldots, R_r)$ is a tuple of $(d+1)$-ary relation symbols, $\mathbf{x} = (x_0, \ldots, x_d)$ is a $(d+1)$-tuple of first-order variables, and ψ is a quantifier-free first-order formula of signature $\mathsf{sg}(d; \mathbf{\Sigma}) \cup \mathbf{R}$ for some tuple of alphabets $\mathbf{\Sigma} = (\Sigma_1, \ldots, \Sigma_k)$.

Such a formula involves two sorts of predicate symbols: those of $\mathsf{sg}(d; \mathbf{\Sigma})$ are called *input predicates* and those of $\mathbf{R}$ are called *guessed predicates*.

It is proved in [13] that the above logic exactly characterizes $\mathbf{NLIN}^d_{\mathrm{ca}}$, the non-deterministic counterpart of $\mathbf{DLIN}^d_{\mathrm{ca}}$. The 'inclusion' $\mathbf{DLIN}^d_{\mathrm{ca}} \subseteq \mathrm{ESO}^d(\forall^{d+1}, arity^{d+1})$ immediately follows, but the converse inclusion is quite unlikely, since it entails $\mathbf{DLIN}^d_{\mathrm{ca}} = \mathbf{NLIN}^d_{\mathrm{ca}}$, which in turn implies $\mathbf{P} = \mathbf{NP}$. Nevertheless, this engages us in looking for a logic characterizing $\mathbf{DLIN}^d_{\mathrm{ca}}$ *inside* the logic $\mathrm{ESO}^d(\forall^{d+1}, arity^{d+1})$. A first restriction of this logic is naturally suggested by the Grädel's characterization of $\mathbf{P}$ already mentioned in the introduction:

[3] For such a result, we need to increase the set of states of the automaton.

▶ **Definition 12.** $\text{ESO-HORN}^d(\forall^{d+1}, \text{arity}^{d+1})$ brings together formulas $\exists \mathbf{R} \forall \mathbf{x} \psi$ among $\text{ESO}^d(\forall^{d+1}, \text{arity}^{d+1})$ whose quantifier-free part ψ is a conjunction of Horn clauses of the form[4] $\alpha_1 \wedge \ldots \wedge \alpha_m \to \alpha_0$ such that:

- each premise $\alpha_1, \ldots, \alpha_m$ is
  - either a *guessed atom* $R(x_0 + i_0, \ldots, x_d + i_d)$ with $R \in \mathbf{R}$ and $i_0, \ldots, i_d \in \mathbb{Z}$,
  - or an *input literal* $I(t_1 + i_1, \ldots, t_q + i_q)$ or $\neg I(t_1 + i_1, \ldots, t_q + i_q)$, with $I \in \mathsf{sg}(d; \mathbf{\Sigma})$, $t_1, \ldots, t_q \in \mathbf{x}$, and $i_0, \ldots, i_q \in \mathbb{Z}$;
- the conclusion literal α_0 is either a 'constant' – the boolean $\perp$ or an input literal – or a guessed atom[5] of the restricted form $R(x_0, \ldots, x_d)$ with $R \in \mathbf{R}$.

We will see that this new logic still contains $\mathbf{DLIN}_{\text{ca}}^d$ but that here again the converse inclusion is unlikely, as argued in Sec. 6. Whence the necessity of a further restriction of the logic, detailed in Def. 14. For now, let us give some motivation for this restriction.

The first-order part of an $\text{ESO-HORN}^d(\forall^{d+1}, \mathit{arity}^{d+1})$-formula can be viewed as a program whose execution, on a given picture structure taken as input, computes the guessed predicates from the input ones. Consider for instance the Horn clause $R(x-2, y-1) \wedge R'(x+1, y-2) \to R(x, y)$ built on guessed predicates R and R'. It establishes a dependence between the values of the guessed predicates (taken as a whole) at place (x, y) and their values at place $(x-2, y-1)$, on one hand, and at place $(x+1, y-2)$, on the other hand. This notion is formalized by the definition below.

▶ **Definition 13.** Let $\Phi = \exists \mathbf{R} \forall x_0, \ldots, x_d \psi$ be in $\text{ESO-HORN}^d(\forall^{d+1}, \mathit{arity}^{d+1})$. A *nonzero* tuple $(i_0, \ldots, i_d) \in \mathbb{Z}^{d+1}$ is an *induction vector* of Φ if there exists a Horn clause C in ψ and two guessed predicates R, R' in $\mathbf{R}$ such that C includes $R(x_0, \ldots, x_d)$ as its conclusion and $R'(x_0 + i_0, \ldots, x_d + i_d)$ among its hypotheses. The set of induction vectors of Φ is called its *induction system* .

The logic involved in the characterization of $\mathbf{DLIN}_{\text{ca}}^d$ that constitutes the core of this paper is defined as follows:

▶ **Definition 14.** A formula $\Phi \in \text{ESO-HORN}^d(\forall^{d+1}, \text{arity}^{d+1})$ with induction system $\mathcal{S}$ is *monotonic* and we write $\Phi \in \text{mon-ESO-HORN}^d(\forall^{d+1}, \text{arity}^{d+1})$ if there exist $a_0, \ldots, a_d \in \mathbb{Z}$ such that each induction vector $(v_0, \ldots, v_d) \in \mathcal{S}$ fulfils $a_0 v_0 + \cdots + a_d v_d < 0$. This condition is called the *monotonicity condition* .

In other words, there exists a hyperplane $a_0 x_0 + \cdots + a_d x_d = 0$, called a *reference hyperplane* of $\mathcal{S}$, such that each vector $(v_0, \ldots, v_d) \in \mathcal{S}$ belongs to the same strict half-space determined by this hyperplane, that means $a_0 v_0 + \cdots + a_d v_d < 0$. One also says that $\mathcal{S} \subset \mathbb{Z}^{d+1}$ *satisfies the monotonicity condition* w.r.t. the reference hyperplane.

We are now in a position to state formally the main result of the paper:

▶ **Theorem 15.** *For* $d \geq 1$, $\mathbf{DLIN}_{ca}^d = \text{mon-ESO-HORN}^d(\forall^{d+1}, \text{arity}^{d+1})$.

The two 'inclusions' underlying the above characterization are proved in Sec. 3 and 5.

---

[4] We will always assume that conjunction has priority over implication.
[5] Alternatively, in Horn formulas, 'guessed' predicates and 'guessed' atoms can be called more intuitively 'computed' predicates and 'computed' atoms.

3 $\mathbf{DLIN}_{\mathsf{ca}} \subseteq$ mon-ESO-HORN

▶ **Proposition 16.** *For $d \geq 1$, $\mathbf{DLIN}^d_{ca} \subseteq$ mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$.*

Proof. Let $L \subseteq \Sigma^{(d)}$ be a d-picture language in $\mathbf{DLIN}^d_{\text{ca}}$. By Lemma 9, there exists a CA $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}_2, \delta)$ of neighborhood $\mathcal{N}_2 = \{-2, -1, 0, 1, 2\}^d$ that recognizes L in time $\tau(n) = n - 1$ with *permanent* state q_0. Let $[\![1, n]\!]$ denote the interval of the n instants of the computation of $\mathcal{A}$ on a d-picture of side n; in particular, the initial and final instants are numbered 1 and n, respectively. We are going to construct a formula $\Phi \in$ mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$ that defines L, i.e., that expresses that the computation of $\mathcal{A}$ on a d-picture p accepts it. It is of the form $\Phi \equiv \exists (R_s)_{s \in \mathcal{Q}} \forall t \forall \mathbf{c} \psi$ where $\mathbf{c}$ denotes the d-tuple of variables $(c_1, \ldots, c_d)$ and, for $s \in \mathcal{Q}$, the intended meaning of the guessed atom $R_s(t, \mathbf{c})$ is the following: at the instant t, the cell $\mathbf{c}$ is in the state s. For simplicity of notation, let us assume $d = 1$. Also assume $n \geq 5$. The quantifier-free part ψ of Φ is the conjunction of three kinds of Horn clauses:

1. the *initialization clauses*: for each $s \in \Sigma$, the clause $\mathsf{min}(t) \wedge Q_s(c) \to R_s(t, c)$;
2. five kinds of *computation clauses* that compute the state at instant $t > 1$ of any cell c according to its possible neighborhoods for $\mathcal{N}_2 = \{-2, -1, 0, 1, 2\}$:
   (i) $c = 1$; (ii) $c = 2$; (iii) general case $c \in [\![3, n-2]\!]$; (iv) $c = n - 1$; (v) $c = n$.
   Let us consider the general case: for any 5-tuple of states $(s_{-2}, s_{-1}, s_0, s_1, s_2) \in (Q - \{q_0, q_a, q_r\})^5$, the clause
   $$\left(\begin{array}{c} R_{s_{-2}}(t-1, c-2) \wedge R_{s_{-1}}(t-1, c-1) \wedge \\ R_{s_0}(t-1, c) \wedge R_{s_1}(t-1, c+1) \wedge R_{s_2}(t-1, c+2) \end{array} \right) \to R_{\delta(s_{-2}, s_{-1}, s_0, s_1, s_2)}(t, c)$$
   computes the state at any instant $t > 1$ of any cell c in the interval $[\![3, n-2]\!]$, which can be tested by the use of $\neg\mathsf{min}()$ and $\neg\mathsf{max}()$ in the premises;
3. the *accepting clause* $R_{q_r}(t, c) \to \perp$, which says that the computation does not reject, and hence accepts, since by hypothesis each computation of $\mathcal{A}$ either accepts or rejects.

By construction, Φ belongs to ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$ and the induction system is $\{-1\} \times \{-2, -1, 0, 1, 2\}^d$, which has a reference hyperplane of equation $t = 0$. Hence, $\Phi \in$ mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1})$, which completes the proof. ◀

4 From the formula to the automaton: the example of palindromes

As the proof of the inclusion mon-ESO-HORN$^d(\forall^{d+1}, \text{arity}^{\,d+1}) \subseteq \mathbf{DLIN}^d_{\text{ca}}$ we will give in Sec. 5 is much more elaborate than its converse, we first give in this section its main ideas which are essentially of geometrical nature by now presenting the inductive definition of a "toy" problem by a monotonic Horn formula from which we will derive a cellular automaton that recognizes the problem.

Let Palindrome(Σ) denote the language of palindromes over a fixed alphabet Σ.

4.1 A monotonic Horn formula defining the language of palindromes

Let us prove that Palindrome(Σ) is definable in mon-ESO-HORN$^1(\forall^2, \text{arity}^{\,2})$. In addition to the set of input unary predicates $\texttt{Input} = \{\mathsf{min}, \mathsf{max}, (Q_s)_{s \in \Sigma}\}$ involved in the picture structure that represents a word $w = w_1 w_2 \ldots w_n \in \Sigma^*$, we need to consider three guessed binary predicates symbols $R_=$, $R_<$ and R_{noPal}. The first one is enforced to contain the equality relation. It is used to inductively map the second one on the usual strict linear order over $[\![1, n]\!]$. This is done with the clauses $\theta_1, \ldots, \theta_5$ below:

- $\theta_1 = \mathsf{min}(x) \wedge \mathsf{min}(y) \to R_=(x, y)$;
- $\theta_2 = \neg\mathsf{min}(x) \wedge \neg\mathsf{min}(y) \wedge R_=(x-1, y-1) \to R_=(x, y)$;
- $\theta_3 = \neg\mathsf{max}(x) \wedge R_=(x+1, y) \to R_<(x, y)$;
- $\theta_4 = \neg\mathsf{max}(x) \wedge R_<(x+1, y) \to R_<(x, y)$;
- $\theta_5 = R_<(x, x) \to \perp$.

The (set of) clauses θ_6 and θ_7 below inductively define R_{noPal} as the set of couples $(x, y) \in [\![1, n]\!]^2$ that fulfill: $x < y$ and the factor $w_x \ldots w_y$ of the input word w *is not* a palindrome. Then the clause θ_8 forces w to be a palindrome:

- $\theta_6 = R_<(x, y) \wedge Q_s(x) \wedge Q_{s'}(y) \to R_{\text{noPal}}(x, y)$, for all $s \neq s'$ in Σ;
- $\theta_7 = R_<(x, y) \wedge R_{\text{noPal}}(x+1, y-1) \to R_{\text{noPal}}(x, y)$;
- $\theta_8 = \mathsf{min}(x) \wedge \mathsf{max}(y) \wedge R_{\text{noPal}}(x, y) \to \perp$.

In conclusion, Palindrome(Σ) is defined by the following formula Φ_{pal}, over the structure $\mathcal{S}(w) = \langle [\![1, n]\!], (Q_s)_{s\in\Sigma}, \mathsf{min}, \mathsf{max}, \mathsf{suc}, \mathsf{pred}\rangle$ associated with an input word $w = w_1 \ldots w_n$:

$$\Phi_{\text{pal}} \equiv \exists R_=, R_<, R_{\text{noPal}} \forall x, y \bigwedge_{i \leq 8} \theta_i.$$

Moreover, Φ_{pal} belongs to ESO-HORN$^1(\forall^2, \text{arity } 2)$ and has $\mathcal{S} = \{(-1, -1), (1, 0), (1, -1)\}$ as its induction system (see Def. 13) Clearly, the system $\mathcal{S}$ satisfies the monotonicity condition of Def. 14 with the line of equation $-x + 2y = 0$ as its reference hyperplane. It follows:

▶ **Proposition 17.** Palindrome(Σ) $\in$ mon-ESO-HORN$^1(\forall^2, \text{arity } 2)$.

4.2 From Φ_{pal} to $\mathcal{A}_{\text{pal}}$

It remains to transform the formula Φ_{pal} above into a one-dimensional cellular automaton $\mathcal{A}_{pal}$ that recognizes the language Palindrome(Σ). For sake of simplicity, we first ignore the input literals and only take account of guessed atoms in the Horn clauses θ_i. Notice that in each clause whose conclusion is a guessed atom $R(x, y)$, $R \in \{R_=, R_<, R_{\text{noPal}}\}$, the guessed atoms occurring as premises have one of the following forms:

$$R'(x, y), \quad R'(x+1, y), \quad R'(x+1, y-1), \quad R'(x-1, y-1).$$

Intuitively, if one regards the set $D_t = \{(x, y) \in [\![1, n]\!]^2 \mid -x + 2y = t\}$ as the line of cells of a one-dimensional CA at instant t, then the conjunction of the above clauses θ_i can be regarded as the transition function of such a CA (see Fig. 1). More formally, in order to introduce the *time* parameter t, we eliminate one of the variables, x for example, and we regard the other variable, y, as the *space* variable c. That is, one makes the change of variables[6] : $t = -x + 2y$; $c = y$.

Let us now explain how the automaton $\mathcal{A}_{\text{pal}}$ to be constructed can take account of the input literals. For each point $(x, y) \in [\![1, n]\!]^2$, we call state$(x, y)$ the tuple of boolean values of all input and output atoms on x and y. That is,

$$\text{state}(x, y) = \begin{pmatrix} \mathsf{min}(x),\ \mathsf{min}(y),\ \mathsf{max}(x),\ \mathsf{max}(y), \\ (Q_s(x))_{s\in\Sigma},\ (Q_s(y))_{s\in\Sigma}, \\ R_=(x, y),\ R_<(x, y),\ R_{\text{noPal}}(x, y) \end{pmatrix},$$

where the values $R_=(x, y)$, $R_<(x, y)$, $R_{\text{noPal}}(x, y)$, are deduced by the Horn formula.

---

[6] There is an analogy between our method and the so-called *loop-skewing* or *polytope/polyhedron* method in compilation and parallel algorithms [6, 1, 7].

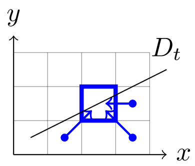


Figure 1 Induction system for guessed atoms before the change of variables.

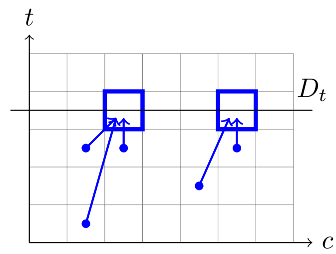


Figure 2 Induction system after the change of variables for guessed atoms (left) and input atoms (right).

We will now discuss the information transit in $\mathcal{A}_{\text{pal}}$; the initialization and termination of the computation will be discussed later. By the change of variables $(x, y) \mapsto (t = -x+2y, c = y)$ whose converse is the function $(t, c) \mapsto (x = -t + 2c, y = c)$, each input atom of the form $I(x)$ becomes $I(-t + 2c)$ and each input atom of the form $I(y)$ becomes $I(c)$. The CA we construct has to memorize in each cell c at instant t the boolean values $I(c)$ and $I(-t + 2c)$, for $I \in$ `Input`. This can be realized as follows:

(a) For each $I(c)$ (former $I(y)$): the CA *conserves* on cell c the boolean value $I(c)$ from an instant $t - 1$ to the next instant t;

(b) For each $I(-t + 2c)$ (former $I(x)$): because of the identity $-t + 2c = -(t - 2) + 2(c - 1)$, whence $I(-t + 2c) \equiv I(-(t - 2) + 2(c - 1))$, the CA only has to *move* to each cell c at instant t the boolean value $I(-(t - 2) + 2(c - 1))$ that is present at instant $t - 2$ on the cell $c - 1$.

All in all, the state of each point $P = (t, c) = (-x + 2y, y)$ is determined by the states of the following points (as shown on Fig. 2):

- $P_1 = (-(x - 1) + 2(y - 1), y - 1) = (t - 1, c - 1)$, $P_2 = (-(x + 1) + 2y, y) = (t - 1, c)$ and $P_3 = (-(x + 1) + 2(y - 1), y - 1) = (t - 3, c - 1)$, because of guessed atoms, and
- $P_2 = (t - 1, c)$ and $P_4 = (t - 2, c - 1)$ because of the above items (a) and (b), respectively, for input atoms.

Hence, the state of a cell c at instant t is determined by the states of: (i) cell $c - 1$ at instant $t - 1$; (ii) cell c at instant $t - 1$; (iii) cell $c - 1$ at instant $t - 3$; (iv) cell $c - 1$ at instant $t - 2$. Figures 1 and 2 below summarize the effects of the change of variables on the induction system.[7]

It seems that we have achieved the design of an automaton of neighborhood $\{-1, 0\}$ that recognizes the language $\textsc{Palindrome}(\Sigma)$ in *linear time* since $t = -x + 2y$ and $x, y \in [\![1, n]\!]$ imply $-n + 2 \leq t \leq 2n - 1$. However, it remains to describe both the initialization and the end of the computation.

The result and the initialization of the computation

The result of the computation is `accept` or `reject` according to whether $\mathcal{S}(w)$ does or does not satisfy the formula Φ_{pal}, where $\mathcal{S}(w)$ is the structure $\langle [\![1, n]\!], (Q_s)_{s \in \Sigma}, \mathsf{min}, \mathsf{max}, \mathsf{suc}, \mathsf{pred} \rangle$

[7] At first glance, Conditions (iii) and (iv) seem to contradict the requirement that the state of any cell c of a CA at instant t should be determined by the *only* states of its neighbour cells at the *previous* instant $t - 1$. However, we can overcome the problem by using the ability of a cell to memorize at any instant t its states at instants $t - 1$ and $t - 2$ with a finite number of states.

associated with the input word $w = w_1 \ldots w_n$. As this is testified by the clause $\theta_8 = \mathsf{min}(x) \wedge \mathsf{max}(y) \wedge R_{\text{noPal}}(x, y) \to \bot$, on the point of coordinates $(x = 1, y = n)$ which become after the change of variables $(t = -x + 2y = 2n - 1,\ c = y = n)$, the acceptance/rejection can be read on the state of cell $c = n$ at the instant $t = 2n - 1$ so that the final state q_a or q_r is obtained at the following instant $2n$.

The initialization of the computation requires some care in connection with the items (a) and (b) of the previous paragraph, about the input bits:

(1) *Initializing each* $I(c)$ (former $I(y)$): The state of each cell $c \in [\![1, n]\!]$ at the instant just before $-n + 2$, i.e. at instant $-n + 1$, should store the boolean value $I(c)$, for each $I \in \mathtt{Input}$.

(2) *Initializing each* $I(-t + 2c)$ (former $I(x)$): Because of the correspondence $x = -t + 2c$ or, equivalently, $c = (x + t)/2$, for all $x \in [\![1, n]\!]$, the boolean value $I(x)$ should be stored in the state of the cell $c = (x + t)/2$ at the maximal instant $t < -n + 2$ such that $(x + t)/2$ is an *integer*; that is the cell $c = (x - n)/2$ at instant $-n$ if $x - n \equiv 0 \pmod 2$ and $c = (x - n + 1)/2$ at instant $-n + 1$ if $x - n \equiv 1 \pmod 2$: see Fig. 3.

The two configurations at the successive instants $-n$ and $-n + 1$ described in items (1) and (2) are called *initialization* configurations. By construction, the space of both configurations – their informative cells, i.e. those in non-quiescent states – is included in the interval $[\![-\lceil n/2 \rceil + 1, n]\!]$.

According to our conventions, the initial configuration of the automaton should be the configuration $\mathcal{C}_{w,q_0}$ associated with the input word w, as defined in Def. 6. However, one can design a routine which, starting from configuration $\mathcal{C}_{w,q_0}$ (with quiescent state q_0), computes the two initialization configurations by using classical techniques of signals in CA's (such as seen in [3]) as shown on Fig. 4. Once each cell contains the right input information, the proper computation can begin. This start will be triggered by a synchronization of all cells of the interval $[\![-\lceil n/2 \rceil + 1, n + 1]\!]$ at time $-n + 1$. The subject of synchronization on cellular automata has been extensively studied (see [21]), here we merely assert that this process can be performed in parallel with no time increase. By a careful examination of this figure, we precisely observe that this precomputation is performed on the interval of cells $[\![-\lceil n/2 \rceil + 1, n + 1]\!]$ during the time interval $[\![-3n, -n + 1]\!]$.[8]

We have now achieved the design of a cellular automaton $\mathcal{A}_{\text{pal}}$ that recognizes in *linear time* the language PALINDROME(Σ) from the monotonic Horn formula Φ_{pal} that defines it.

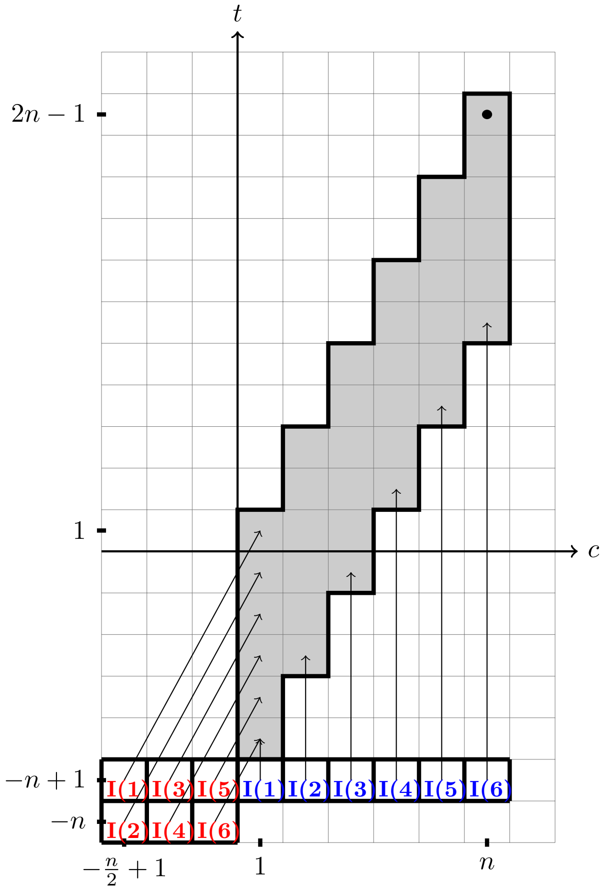
5 mon-ESO-HORN $\subseteq$ $\mathbf{DLIN}_{\mathsf{ca}}$

The main problem we have to deal with in the general case as in the previous example is the integration of the input to the computation of the CA to be constructed. For that purpose, we will need the following technical lemma whose proof is easy and left to the reader:

▶ **Lemma 18.** *Let $\mathcal{S} \subset \mathbb{Z}^{d+1}$ be an induction system satisfying the monotonicity condition w.r.t. some reference hyperplane. Then, $\mathcal{S}$ has another reference hyperplane of equation $a_0 x_0 + \cdots + a_d x_d = 0$ where each coefficient a_i ($i \in [\![0, d]\!]$) is a non-zero integer.*

---

[8] Notice that our numbering of instants is not canonical. It is only a convenient time scale for describing our algorithm. In particular, the initial instant of the (pre)-computation of the upper part of Fig. 4 is $-2n$ when n is *even* and $-2n - 1$ when n is *odd*, and the initial instant of the two signals of the lower part of Fig. 4 is $-3n$. We let the reader imagine the variants of Fig. 3 and Fig. 4 for the *odd* case.



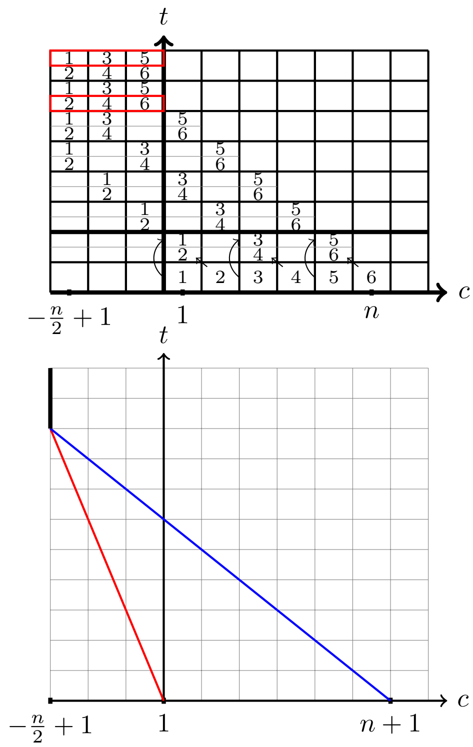


Figure 3 Initial positions and translation vectors for $I(x) = I(-t+2c)$ (in red) and $I(y) = I(c)$ (in blue) when n is *even* (here $n = 6$). The gray parallelogram is where the induction actually happens. The result of the computation lies at the upper right cell.

Figure 4 The linear precomputation of $I(x)$ can be done by stacking the information of the cells in the odd columns, then packing it to the left against a "wall" at $c = -\frac{n}{2} + 1$ (n *even*). The bottom figure shows how the wall can be constructed in linear time with two signals of slope $-\frac{1}{3}$ (*resp.* -1) starting in cell 1 (*resp.* $n+1$) at instant $-3n$.

We are now ready to prove the most difficult inclusion of Thm. 15:

▶ **Proposition 19.** *For each* $d \geq 1$, $\text{mon-ESO-HORN}^d(\forall^{d+1}, \text{arity}^{\,d+1}) \subseteq \mathbf{DLIN}^d_{ca}$.

Proof. Let L be a d-picture language defined by a formula $\Phi \equiv \exists R_1 \ldots \exists R_r \forall x_0 \ldots \forall x_d\, \psi$ in $\text{mon-ESO-HORN}^d(\forall^{d+1}, \text{arity}^{\,d+1})$ with an induction system $\mathcal{S}$ and a reference hyperplane $a_0x_0 + a_1x_1 + \cdots + a_dx_d = 0$, with coefficients $a_i \in \mathbb{Z}^*$ for all $i \in [\![0, d]\!]$, as justified by Lemma 18. For simplicity of notation, assume all the coefficients a_i are positive.

First, for sake of simplicity, let us ignore the input literals and take account of the only guessed atoms $R_i(x_0 + i_0, \ldots, x_d + i_d)$ in the Horn clauses. Intuitively, if one regards the hyperplane $H_t = \{(x_0, x_1, \ldots, x_d) \in [\![1, n]\!]^{d+1} \mid a_0x_0 + a_1x_1 + \cdots + a_dx_d = t\}$ as the set of cells of a d-dimensional CA at instant t, then the conjunction of Horn clauses ψ can be regarded as the transition function of such a CA. More formally, in order to introduce the *time* parameter t, we eliminate one of the variables, x_0 for example, and we regard the other variables, $x_1, \dots, x_d$, as the *space* variables, i.e. the respective d coordinates $c_1, \dots, c_d$ of a cell. More precisely, one makes the following change of variables:

$$\begin{pmatrix} t = a_0 x_0 + \cdots + a_d x_d, \\ c_1 = x_1, \dots, c_d = x_d \end{pmatrix}, \text{ whose converse is } \begin{pmatrix} x_0 = (t - a_1 c_1 - \cdots - a_d c_d)/a_0, \\ x_1 = c_1, \dots, x_d = c_d \end{pmatrix}.$$

As in Section 4.2, we associate with each point $\mathbf{x} = (x_0, \dots, x_d) \in [\![1, n]\!]^{d+1}$, the tuple $\text{state}(\mathbf{x})$ of boolean values of all input and guessed atoms on $\mathbf{x}$. That is,

$$\text{state}(\mathbf{x}) = \left((I(\mathbf{u}))_{\substack{I \in \texttt{Input} \\ \mathbf{u} \subsetneq \mathbf{x}}}, \; (R(\mathbf{x}))_{R \in \texttt{Guess}} \right).$$

Here, we denote by `Input` (resp. `Guess`) the set of input (resp. guessed) predicates occurring in the formula. Furthermore, $u \in \mathbf{x}$ means that u is any variable among $x_0, \dots, x_d$, while $\mathbf{u} \subsetneq \mathbf{x}$ means that $\mathbf{u}$ is any m-tuple built from those variables, where $m \leq d$ is the arity of I. Besides, the values of the guessed litterals $R(\mathbf{x})$, $R \in \texttt{Guess}$, are deduced by the Horn formula $\forall \mathbf{x} \psi$.

If one ignores the input literals, the state of each point

$$P = (t, c_1, \dots, c_d) = \left(\sum_{j=0}^{d} a_j x_j, \; x_1, \dots, x_d \right)$$

is determined by the states of the points

$$P_v = \left(\sum_{j=0}^{d} a_j (x_j + v_j), \; x_1 + v_1, \dots, x_d + v_d \right) = \left(t + \sum_{j=0}^{d} a_j v_j, \; c_1 + v_1, \dots, c_d + v_d \right)$$

for each vector $v = (v_0, \dots, v_d)$ of the induction system $\mathcal{S}$. In other words the state of the cell $(c_1, \dots, c_d)$ at instant t is determined by the states of the cells $(c_1 + v_1, \dots, c_d + v_d)$ at the respective *previous* instants $t + \sum_{j=0}^{d} a_j v_j$ for the vectors $v = (v_0, \dots, v_d) \in \mathcal{S}$. (Recall that $\sum_{j=0}^{d} a_j v_j < 0$, by hypothesis.)

Let us now explain how the CA we construct can take account of the input atoms, i.e. let us describe how the CA moves the input bits. The crucial point is that at least one of the $d+1$ variables $x_0, \dots, x_d$ does *not* occur in each input atom because the arity of each input predicate is at most d. This missing variable is used as a 'transport variable' of the values of the concerned input atom. As a *generic* example, let us consider the input atom $I(x_0, x_2, \dots, x_d)$ where I is an input predicate of arity d and where the variable x_1 does not occur[9]. After the above-mentioned change of variables, this atom becomes

$$I(\frac{1}{a_0}(t - a_1 c_1 \cdots - a_d c_d), \; c_2, \dots, c_d).$$

Because of the identity $(t - a_1) - a_1(c_1 - 1) - a_2 c_2 \cdots - a_d c_d = t - a_1 c_1 - a_2 c_2 \cdots - a_d c_d$ the automaton only has to *move* to each cell $(c_1, c_2, \dots, c_d)$ at instant t the boolean value

$$I(\frac{1}{a_0}((t - a_1) - a_1(c_1 - 1) - a_2 c_2 \cdots - a_d c_d), \; c_2, \dots, c_d)$$

[9] As we have seen in previous examples the case where one variable among $x_2, \dots, x_d$ does not occur in an input atom is similar; the case where x_0 does not occur or the case where the arity of the input predicate is less than d are easier to deal with as we have also seen.

which is stored at instant $t-a_1$ in the state of cell $(c_1-1,c_2,\ldots,c_d)$. In terms of cellular automaton, the values of the input atom $I(x_0,x_2,\ldots,x_d)$ are moved/transmitted by linear parallel "signals" which cover all the inductive space $[\![1,n]\!]^{d+1}$.

Time and initialization of the computation: Since the $d+1$ original variables $x_0,\ldots,x_d$ lie in $[\![1,n]\!]$, the domain of the time variable $t = a_0x_0+\cdots+a_dx_d$ is $[\![A,An]\!]$, where $A = a_0+\cdots+a_d$. As a consequence, the equation of the cell hyperplane at the *initial* instant (resp. *final* instant) in the space-time diagram is $t=A$ (resp. $t=An$)[10].

The initialization of the input values (input "signals") before the instant $t=A$ is the most delicate/technical part of the computation. It is sufficient to describe the initialization of the values of the input "signals" for our generic example[11] of input atom $\alpha \equiv I(x_0,x_2,\ldots,x_d)$ or, equivalently, $\alpha \equiv I(\frac{1}{a_0}(t-a_1c_1-\cdots-a_dc_d)\,,c_2,\ldots,c_d)$, for which c_1 (that is the missing variable x_1 of α) is the "transport" variable. To give the reader the geometric intuition of the following construction in the general case we invite her to consult Fig. 3 and 4 of Sec. 4.2 in the particular case of atom $\alpha \equiv I(x) \equiv I(-t+2c)$ of the formula that defines Palindrome.

Because of the correspondence $t = a_0x_0 + a_1c_1 + a_2x_2\cdots + a_dx_d$ or, equivalently, $c_1 = (t-a_0x_0-a_2x_2\cdots-a_dx_d)/a_1$, with $c_2=x_2,\ldots,c_d=x_d$, for all $(x_0,x_2,\ldots,x_d)\in[\![1,n]\!]^d$, the boolean value $I(x_0,x_2,\ldots,x_d)$ should be stored – for the initialization of its input "signal" – in the state of the cell $(c_1,x_2,\ldots,x_d)$ such that $c_1=(t_0-a_0x_0-a_2x_2\cdots-a_dx_d)/a_1$ at the maximal instant $t_0<A$ (depending on the tuple $(x_0,x_2,\ldots,x_d)$) such that the quotient $(t_0-a_0x_0-a_2x_2\cdots-a_dx_d)/a_1$ is an *integer*. Let i be the integer in $[\![0,a_1-1]\!]$ such that $A-a_0x_0-a_2x_2\cdots-a_dx_d \equiv -i \pmod{a_1}$. It is easy to verify that $t_0=A-a_1+i$. So, the boolean value $I(x_0,x_2,\ldots,x_d)$ should be stored/initialized at the instant $t_0=A-a_1+i$ in (the state of) the cell $(c_1,x_2,\ldots,x_d)$ where $c_1=(A-a_1+i-a_0x_0-a_2x_2\cdots-a_dx_d)/a_1$: see Fig. 3.

Note that for the atom $\alpha\equiv I(x_0,x_2,\ldots,x_d)$, there are a_1 distinct "initialization" configurations in the respective a_1 hyperplanes H_{t_0}, where $t_0=A-a_1+i$ with $i\in[\![0,a_1-1]\!]$, according to the possible values of the function $f(x_0,x_2,\ldots,x_d)=A-a_0x_0-a_2x_2\cdots-a_dx_d$ modulo a_1. Furthermore, one can verify that, by construction, the space of the "initialization" configurations – their informative cells, in non-quiescent states – is included in a hypercube of the form $[\![-bn,bn]\!]^d$, for some constant integer $b>0$.

Pre-computation and end of computation: The initial configuration of a d-CA that recognizes the d-picture language L should be the *picture configuration* $\mathcal{C}_{p,q_0}$ where p is the input picture. Therefore, there should be a *pre-computation* starting from $\mathcal{C}_{p,q_0}$ that computes the "initialization" configurations of the input atoms of Φ. By the classical technique of signals in CAs (see [3]) we have exemplified above in the case of Palindrome (see Fig. 4), this can be done in linear space and linear time.

Similarly, the result of the computation should be given by the accept/reject state, q_a or q_r, in the *reference* cell $\mathbf{n}=(n,\ldots,n)$. This is realized in linear time by gathering in the reference cell the possible contradictions deduced in cells for Horn clauses.

For lack of space, we have omitted to deal with loops in Horn clauses: the possible presence of guessed atoms of the form $R(t,\mathbf{c})$, i.e. without predecessor/successor functions, *both as conclusions and as hypotheses* of clauses of monotonic Horn formulas seemingly

[10] In the sequel, A always denote the sum $a_0+\cdots+a_d$. Also, recall that each a_i is positive.
[11] Here again, all the other examples have either exactly the same treatment or a simpler one.

contradicts the "strict monotonicity" of the induction. We leave the reader to cope with this point. This achieves the proof of Prop. 19 and Thm. 15. ◀

6 Optimality of our main result

It is natural to ask whether the monotonicity condition can be removed or weakened in our main result. This is unlikely because it would imply (as the reader can convince herself) the following time-space trade-off which would be a breakthrough in computational complexity:

▶ **Proposition 20.** *If we had* $\mathbf{DLIN}^d_{ca} = \text{ESO-HORN}^d(\forall^{d+1}, \text{arity}^{\,d+1})$ *or the weaker equality* $\mathbf{DLIN}^d_{ca} = \text{weak-mon-ESO-HORN}^d(\forall^{d+1}, \text{arity}^{\,d+1})$ *for a given* $d > 1$*, then any set of words recognizable by a 1-CA in time* n^d *on* n *cells would be recognizable by a* d*-CA in time* $O(n)$ *on* $O(n^d)$ *cells.*

Here, $\text{weak-mon-ESO-HORN}^d(\forall^{d+1}, \text{arity}^{\,d+1})$ denotes the variant of the class $\text{mon-ESO-HORN}^d(\forall^{d+1}, \text{arity}^{\,d+1})$ where the strict inequality $a_0x_0 + \cdots + a_dx_d < 0$ of the monotonicity condition is replaced by the non-strict inequality $a_0x_0 + \cdots + a_dx_d \leq 0$.

References

1. Uday Bondhugula, Albert Hartono, J. Ramanujam, and P. Sadayappan. A practical automatic polyhedral parallelizer and locality optimizer. In *Proceedings of the ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation, Tucson, AZ, USA, June 7-13, 2008*, pages 101–113, 2008. `doi:10.1145/1375581.1375595`.
2. Walter Bucher, II Culik, et al. On real time and linear time cellular automata. *RAIRO, Informatique théorique*, 18(4):307–325, 1984.
3. Marianne Delorme and Jacques Mazoyer. Signals on cellular automata. In Andrew Adamatzky, editor, *Collision-Based Computing*, pages 231–275. Springer London, London, 2002. `doi:10.1007/978-1-4471-0129-1_9`.
4. H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, 1995.
5. R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. M. Karp, editor, *Complexity of Computation, SIAM-AMS Proceedings*, pages 43–73, 1974.
6. Paul Feautrier. Some efficient solutions to the affine scheduling problem. part II. multidimensional time. *International Journal of Parallel Programming*, 21(6):389–420, 1992. `doi:10.1007/BF01379404`.
7. Paul Feautrier and Christian Lengauer. Polyhedron model. In *Encyclopedia of Parallel Computing*, pages 1581–1592. 2011. `doi:10.1007/978-0-387-09766-4_502`.
8. E. Grädel. Capturing complexity classes by fragments of second order logic. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference, Chicago, Illinois, USA, June 30 – July 3, 1991*, pages 341–352, 1991. `doi:10.1109/SCT.1991.160279`.
9. E. Grädel, Ph. G. Kolaitis, L. Libkin, M. Marx, J. Spencer, M. Y. Vardi, Y. Venema, and S. Weinstein. *Finite Model Theory and Its Applications*. Texts in Theoretical Computer Science. Springer, 2007.
10. A. Grandjean and V. Poupet. A Linear Acceleration Theorem for 2D Cellular Automata on All Complete Neighborhoods. In I. Chatzigiannakis, M. Mitzenmacher, Y. Rabani, and D. Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 115:1–115:12, Dagstuhl, Germany, 2016. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.ICALP.2016.115`.
11. E. Grandjean and F. Olive. Graph properties checkable in linear time in the number of vertices. *J. Comput. Syst. Sci.*, 68(3):546–597, 2004. `doi:10.1016/j.jcss.2003.09.002`.

12 E. Grandjean and F. Olive. Descriptive complexity for pictures languages. In P. Cégielski and A. Durand, editors, *Computer Science Logic (CSL'12) – 26th International Workshop/21st Annual Conference of the EACSL*, volume 16 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 274–288, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CSL.2012.274.

13 E. Grandjean and F. Olive. A logical approach to locality in pictures languages. *J. Comput. Syst. Sci.*, 82(6):959–1006, 2016. doi:10.1016/j.jcss.2016.01.005.

14 E. Grandjean and T. Schwentick. Machine-independent characterizations and complete problems for deterministic linear time. *SIAM J. Comput.*, 32(1):196–230, 2002. doi:10.1137/S0097539799360240.

15 N. Immerman. *Descriptive complexity.* Graduate texts in computer science. Springer, 1999.

16 J. Kari. Basic concepts of cellular automata. In G. Rozenberg, T. Bäck, and J. N. Kok, editors, *Handbook of Natural Computing*, volume 1, pages 3–24. Springer, 2012.

17 C. Lautemann, N. Schweikardt, and T. Schwentick. A logical characterisation of linear time on nondeterministic turing machines. In *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 4-6, 1999, Proceedings*, pages 143–152, 1999. doi:10.1007/3-540-49116-3_13.

18 L. Libkin. *Elements of Finite Model Theory.* Springer, 2004.

19 Jacques Mazoyer and Nicolas Reimen. A linear speed-up theorem for cellular automata. *Theoretical Computer Science*, 101(1):59–98, 1992.

20 T. Schwentick. Descriptive complexity, lower bounds and linear time. In *Computer Science Logic, 12th International Workshop, CSL'98, Annual Conference of the EACSL, Brno, Czech Republic, August 24-28, 1998, Proceedings*, pages 9–28, 1998. doi:10.1007/10703163_2.

21 Hiroshi Umeo. Firing squad synchronization problem in cellular automata. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 3537–3574. Springer New York, New York, NY, 2009. doi:10.1007/978-0-387-30440-3_211.