Iterative Arrays: Little Resources Big Size Impact

Martin Kutrib and Andreas Malcher

Institut für Informatik, Universität Giessen
Arndtstr. 2, 35392 Giessen, Germany
{kutrib,malcher}@informatik.uni-giessen.de

Abstract. We investigate the descriptional complexity of little resources added to deterministic one-dimensional real-time iterative arrays. More precisely, we study the impact of adding sublinearly more time obtaining time complexities strictly in between real time and linear time, adding dimensions, allow the communication cell to perform a few nondeterministic steps, and increase the number of bits that may be communicated to neighboring cells slightly. In all cases it is shown that there are arbitrary savings in the size of the descriptions of the arrays which cannot be bounded by any computable function.

1 Introduction

The approach to analyze the descriptional complexity, that is, the size of systems as opposed to the computational complexity seems to originate from [24], where the relative succinctness of regular languages represented by deterministic finite state and deterministic pushdown machines is studied. In general, suppose that a set of syntactical patterns, in our terms a formal language, has to be stored or transmitted. Then there is a natural interest to represent the patterns as succinctly as possible. As a simple example, we consider the representation of a regular language over the symbols a and b by a minimal deterministic finite state machine having, say n states. Then the size of the machine, that is, the length of its description given by a fixed set of, say x symbols, is roughly $2 \cdot n \cdot \log_x(n)$, since for any state and input symbol we have to write down the next state using the x symbols. It is well known, that any regular language can be represented by a nondeterministic finite state machine as well. Moreover, there are examples where the minimal deterministic machine equivalent to an n-state nondeterministic device requires 2^n states. So, the representation can be exponentially more succinct, when the resource 'nondeterminism' is added to finite state machines. Can we do better? What if deterministic pushdown machines are used to represent regular languages? In [26] it has been shown that in this case the order of magnitude of the size can be reduced double exponentially. So, adding the resource 'pushdown store' to finite state machines has a big impact on the size of the representation. Can we do still better? What if we add nondeterminism *and* a pushdown store? In [21] as one of the cornerstones of descriptional complexity theory it is shown that in this case there is no computable function serving as upper bound for the possible gain in economy of description. With other words, when regular languages are represented by nondeterministic pushdown machines, one can choose an arbitrarily large computable function f but the gain in economy of description eventually exceeds f. This qualitatively phenomenon is called *non-recursive trade-off* (see, for example, [9,10,11,15]).

Here we start with one of the simplest models for massively parallel computations, that is, with deterministic one-dimensional iterative arrays (IA), which sometimes are called cellular automata with sequential input, operating in real time. In connection with formal language processing IAs have been introduced in [7], where it was shown that the language family accepted by real-time IAs forms a Boolean algebra not closed under concatenation and reversal. In [6] it is shown that for every context-free language a two-dimensional linear-time IA parser exists. In [8] a real-time acceptor for prime numbers has been constructed. Pattern manipulation is the main aspect in [1]. A characterization of various types of IAs by restricted Turing machines and several results especially speed-up theorems are given in [12,13,14]. Several more results concerning formal languages can be found in [22,23,25]. We study here the impact on the descriptional complexity of deterministic one-dimensional real-time iterative arrays when little resources are added. In particular, we add sublinearly more time obtaining time complexities strictly in between real time and linear time. Then we add dimensions, allow the communication cell to perform a few nondeterministic steps, and increase the number of bits that may be communicated to neighboring cells slightly. In all cases non-recursive trade-offs are proved. So, from a compressibility point of view it is worthwile to add them.

In the next section we recall the necessary definitions and notions, and the basics of descriptional complexity theory. In the following four sections we present our results. The proofs are omitted here owing to space restrictions.

2 Preliminaries and Definitions

We denote the rational numbers by $\mathbb{Q}$ and the non-negative integers by $\mathbb{N}$. The empty word is denoted by λ, the reversal of a word w by w^R, and for the length of w we write $|w|$. The cardinality of a set M is denoted by $|M|$ and its powerset by 2^M. We write $\subseteq$ for set inclusion, and $\subset$ for strict set inclusion.

An iterative array is an infinite linear array of finite automata, sometimes called cells. We identify the cells by natural numbers. Each cell except the origin is connected to its both nearest neighbors (one to the left and one to the right). The input is supplied sequentially to the distinguished communication cell at the origin which is connected to its immediate neighbor to the right only. For this reason, we have two different local transition functions. The state transition of all cells but the communication cell depends on the current state of the cell itself and the current states of its both neighbors. The state transition of the communication cell additionally depends on the current input symbol (or if the whole input has been consumed on a special end-of-input symbol). The finite automata work synchronously at discrete time steps. Initially they are in the so-called quiescent state.

Definition 1. *An* iterative array *(IA) is a system* $\langle S, A, \#, F, s_0, \delta, \delta_0 \rangle$*, where*

1. *S is the finite, nonempty set of* cell states*,*
2. *A is the finite, nonempty set of* input symbols*,*
3. *$\# \notin A$ is the* end-of-input symbol*,*
4. *$F \subseteq S$ is the set of* accepting states*,*
5. *$s_0 \in S$ is the* quiescent state*,*
6. *$\delta : S^3 \to S$ is the* local transition function for non-communication cells *satisfying $\delta(s_0, s_0, s_0) = s_0$,*
7. *$\delta_0 : (A \cup \{\#\}) \times S^2 \to S$ is the* local transition function for the communication cell.

Let M be an IA. A configuration of M at some time $t \geq 0$ is a description of its global state which is actually a pair (w_t, c_t), where $w_t \in A^*$ is the remaining input sequence and $c_t : \mathbb{N} \to S$ is a mapping that maps the single cells to their current states. The configuration (w_0, c_0) at initial time 0 is defined by the input w_0 and the mapping $c_0(i) = s_0$, $i \geq 0$, while subsequent configurations are chosen according to the global transition function Δ. Let (w_t, c_t), $t \geq 0$, be a configuration. Then its successor configuration $(w_{t+1}, c_{t+1}) = \Delta\big((w_t, c_t)\big)$ is defined as $c_{t+1}(i) = \delta\big(c_t(i-1), c_t(i), c_t(i+1)\big)$, $c_{t+1}(0) = \delta_0\big(a, c_t(0), c_t(1)\big)$, where $i \geq 1$, and $a = \#$, $w_{t+1} = \lambda$ if $w_t = \lambda$, and $a = a_1$, $w_{t+1} = a_2 a_3 \cdots a_n$ if $w_t = a_1 a_2 \cdots a_n$. Thus, the global transition function Δ is induced by δ and δ_0.

An input w is accepted by an IA M if at some time i during its course of computation the communication cell enters an accepting state. The *language accepted* by M is defined as $L(M) = \{\, w \in A^* \mid w \text{ is accepted by } M \,\}$. Let $t : \mathbb{N} \to \mathbb{N}$ be a mapping. If all $w \in L(M)$ are accepted with at most $t(|w|)$ time steps, then $L(M)$ is said to be of time complexity t. The family of all languages that are accepted by IAs with time complexity t is denoted by $\mathscr{L}_t(\text{IA})$. The index is omitted for arbitrary time. If t is the function $n+1$, acceptance is said to be in *real time* and we write $\mathscr{L}_{rt}(\text{IA})$.

2.1 Descriptional Complexity

We recall some notation for descriptional complexity. Following [11] we say that a *descriptional system* $\mathcal{S}$ is a set of finite descriptors such that each $D \in \mathcal{S}$ describes a formal language $L(D)$, and the underlying alphabet alph(D) over which D represents a language can be read off from D. The *family of languages represented* (or *described*) by $\mathcal{S}$ is $\mathscr{L}(\mathcal{S}) = \{\, L(D) \mid D \in \mathcal{S} \,\}$. For every language L, the set $\mathcal{S}(L) = \{\, D \in \mathcal{S} \mid L(D) = L \,\}$ is the set of its descriptors in $\mathcal{S}$. A *complexity measure* for a descriptional system $\mathcal{S}$ is a total computable mapping $c : \mathcal{S} \to \mathbb{N}$.

Example 2. Iterative arrays can be encoded over some fixed alphabet such that their input alphabets can be extracted from the encodings. The set of these encodings is a descriptional system $\mathcal{S}$, and $\mathscr{L}(\mathcal{S})$ is $\mathscr{L}(\text{IA})$.

Examples for complexity measures for IAs are the total number of symbols, that is, the *length of the encoding* (length), or the total *number of transitions* (trans) in δ and δ_0. □

Here we only use complexity measures that (with respect to the underlying alphabets) are related to length by a computable function. If there is a total computable function $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that, for all $D \in \mathcal{S}$, $\mathsf{length}(D) \leq g(c(D), |\mathrm{alph}(D)|)$, then c is said to be an *s-measure*. If, in addition, for any alphabet A, the set of descriptors in $\mathcal{S}$ describing languages over A is recursively enumerable in order of increasing size, then c is said to be an *sn-measure*. Clearly, length and trans are sn-measures for iterative arrays.

Whenever we consider the relative succinctness of two descriptional systems $\mathcal{S}_1$ and $\mathcal{S}_2$, we assume the intersection $\mathscr{L}(\mathcal{S}_1) \cap \mathscr{L}(\mathcal{S}_2)$ to be non-empty. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be descriptional systems with complexity measures c_1 and c_2, respectively. A total function $f : \mathbb{N} \to \mathbb{N}$ is an *upper bound* for the increase in complexity when changing from a descriptor in $\mathcal{S}_1$ to an equivalent descriptor in $\mathcal{S}_2$, if for all $D_1 \in \mathcal{S}_1$ with $L(D_1) \in \mathscr{L}(\mathcal{S}_2)$, there exists a $D_2 \in \mathcal{S}_2(L(D_1))$ such that $c_2(D_2) \leq f(c_1(D_1))$.

If there is no recursive, that is, computable function serving as upper bound, the *trade-off is said to be non-recursive.* That is, whenever the trade-off from one descriptional system to another is non-recursive, one can choose an arbitrarily large recursive function f but the gain in economy of description eventually exceeds f when changing from the former system to the latter. In fact, the non-recursive trade-offs are independent of particular sn-measures, as any two sn-measures c_1 and c_2 for some descriptional system $\mathcal{S}$ are related by a recursive function. So, a non-recursive trade-off exceeds any difference caused by applying two sn-measures. For establishing non-recursive trade-offs the following general result is useful.

Theorem 3 ([11]). *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two descriptional systems for recursive languages such that any descriptor D in $\mathcal{S}_1$ and $\mathcal{S}_2$ can effectively be converted into a Turing machine that decides $L(D)$, and let c_1 be a measure for $\mathcal{S}_1$ and c_2 be an sn-measure for $\mathcal{S}_2$. If there exists a descriptional system $\mathcal{S}_3$ and a property P that is not semi-decidable for descriptors from $\mathcal{S}_3$, such that, given an arbitrary $D_3 \in \mathcal{S}_3$, (i) there exists an effective procedure to construct a descriptor D_1 in $\mathcal{S}_1$, and (ii) D_1 has an equivalent descriptor in $\mathcal{S}_2$ if and only if D_3 does not have property P, then the trade-off between S_1 and S_2 is non-recursive.*

In the following we show all non-recursive trade-offs between iterative arrays having little additional resources and those that do not have these resources by reduction of the finiteness problem for linearly space bounded Turing machines. In order to apply Theorem 3, we use the family of deterministic linearly space bounded one-tape one-head Turing machines, so-called linear bounded automata (LBA), as descriptional system $\mathcal{S}_3$. Property P is *infiniteness*, which is not semi-decidable for LBAs. Next, given an arbitrary LBA M, that is, a descriptor $D_3 \in \mathcal{S}_3$, we must construct an iterative array with the additional resources, that is, a descriptor D_1 in $\mathcal{S}_1$, that has an equivalent iterative array without the resources, that is, a descriptor in $\mathcal{S}_2$, if and only if M accepts a finite language, that is, D_3 does not have property P.

For the reduction, we consider strings which record all configurations of an accepting computation of a given LBA. Without loss of generality and for technical reasons, one can assume that any accepting computation has at least three and, in general, an odd number of steps. Therefore, it is represented by an even number of configurations. Moreover, it is assumed that the LBAs get their input in between two endmarkers, and that a configuration is halting if and only if it is accepting.

Let Q be the state set of some LBA M, where q_0 is the initial state, $T \cap Q = \emptyset$ is the tape alphabet containing the endmarkers $\rhd$ and $\lhd$, and $\Sigma \subset T$ is the input alphabet. A configuration of M can be written as a string of the form $\rhd T^* Q T^* \lhd$ such that, $\rhd t_1 t_2 \cdots t_i q t_{i+1} \cdots t_n \lhd$ is used to express that $\rhd t_1 t_2 \cdots t_n \lhd$ is the tape inscription, M is in state q, and scans tape symbol t_{i+1}. We consider words of the form $w_1 \$ w_3 \$ \cdots \$ w_{2k-1} ¢ w_{2k}^R \$ \cdots \$ w_4^R \$ w_2^R$, where w_i are configurations, $\$$ and $¢$ are symbols not appearing in w_i, w_1 is an initial configuration of the form $q_0 \Sigma^*$, w_{2k} is an accepting, that is, halting configuration, and w_{i+1} is the successor configuration of w_i, for $1 \leq i \leq 2k$. Let $\$'$ and $¢'$ be new symbols and T' and Q' primed copies of T and Q. The set of *valid computations* VALC(M) is now defined to be the set of words $\varphi^{-1}(w)$, where w is a word of the form above and φ is the homomorphism defined by $\varphi(a) = a$ if $a \in T \cup Q \cup \{\$, ¢\}$, and $\varphi(a') = a$ if $a' \in T' \cup Q' \cup \{\$', ¢'\}$. The set of *invalid computations* INVALC(M) is the complement of VALC(M) with respect to the *coding alphabet* $\Lambda_M = \{\$, \$', ¢, ¢'\} \cup T \cup T' \cup Q \cup Q'$. By using the redundant primed symbols we ensure that for any $u \in$ VALC(M) there are are at least $2^{|u|}$ further strings of length $|u|$ in VALC(M)) [19].

Since the language class $\mathscr{L}_{rt}$(IA) is closed under inverse homomorphisms, essentially, the following crucial theorem has been shown in [18].

Theorem 4. *Let M be an LBA. Then real-time iterative arrays that accept VALC(M) and INVALC(M) can effectively be constructed.*

3 Time

Basically, for any non-trivial computation an iterative array has to read at least one end-of-input symbol. Therefore, real time is defined to be $(n+1)$-time. In [2] it has been shown that there exists an infinite dense and strict time hierarchy between real time and linear time. So adding a little bit more time yields a strictly stronger class of iterative arrays. Here we consider the differences of their descriptional complexities.

In order to deal with infinite dense time hierarchies in almost all cases reasonable time bounding functions are required. Usually the notion reasonable is substantiated in terms of the computability or constructibility of the function with respect to the device in question. Here we consider so-called *IA-constructible* functions. A strictly increasing function $f : \mathbb{N} \to \mathbb{N}$ is IA-constructible, if there exists an IA such that on empty input the leftmost cell enters an accepting state exactly at all time steps $f(i)$, $1 \leq i$. Note that since all IA-constructible functions f are necessarily strictly increasing, for their inverses we have $f^{-1}(n) \leq n$, for all $n \geq 1$.

The family of IA-constructible functions is very rich. It includes $n!$, k^n, n^k, $n + \lfloor\sqrt{n}\rfloor$, $n + \lfloor\log\rfloor$, etc., where $k \geq 1$ is an integer. It is closed under the operations such as addition of constants, addition, iterated addition, multiplication, composition, minimum, maximum etc. [20]. Further results can be found in [4,5].

Now, let M be an LBA, r^{-1} be an IA-constructible function, and define a function h_r as $h_r(n) = r^{-1}((n+1)^2)$, and a language

$$L_{r,M} = \{\, \$^{h_r(m)-(m+1)^2+1} w_1 \$ w_2 \$ \cdots \$ w_m ¢ y \mid m \geq 1, w_i \in A_M^m, 1 \leq i \leq m, \\ y \in \text{VALC}(M) \text{ and } \exists\, 1 \leq j \leq m : y = w_j \,\}.$$

Lemma 5. *Let M be an LBA and $r_1, r_2 : \mathbb{N} \to \mathbb{N}$ be two increasing functions so that $r_2 \in o(r_1)$ and r_1^{-1} is IA-constructible. Then $L_{r_1,M} \in \mathscr{L}_{n+r_2(n)}(IA)$ if and only if $L(M)$ is finite.*

In order to apply Theorem 3 to show the non-recursive trade-off between iterative arrays with time complexity $n + r_1(n)$ and $n + r_2(n)$ as mentioned above, it now suffices to show $L_{r_1,M} \in \mathscr{L}_{n+r_1(n)}(\text{IA})$. In [2] an $(n + r_1(n))$-time IA for a language L_{r_1} has been constructed, where essentially, L_{r_1} is derived from $L_{r_1,M}$ by defining the subwords w_i over a binary alphabet and omitting the condition that the suffix following ¢ has to belong to VALC(M). By Theorem 4 it is not hard to extend the construction to language $L_{r_1,M}$. A corresponding IA simply can check the suffix on another track. So, the next theorem follows.

Theorem 6. *Let $r_1, r_2 : \mathbb{N} \to \mathbb{N}$ be two increasing functions so that $r_2 \in o(r_1)$ and r_1^{-1} is IA-constructible. Then the trade-off between $(n+r_1(n))$-time IAs and $(n + r_2(n))$-time IAs is non-recursive.*

4 Nondeterminism

This section is devoted to nondeterministic iterative arrays, where the nondeterminism is regarded as a limited resource. The ability to perform nondeterministic transitions is restricted to the communication cell, all the other automata are deterministic ones. Moreover, the number of allowed nondeterministic transitions is limited dependent on the length of the input. Such iterative arrays with limited nondeterministic communication cell have been introduced and investigated in [3]. In particular, it has been shown that the number of nondeterministic transitions can be reduced by any constant without decreasing the computational capacity, and that there is a strict and dense infinite hierarchy of language classes dependent on sublogarithmic limits of the nondeterminism.

In order to define so-called gG-IA (g guess IA) the transition function δ_0 for the communication cell is replaced by a nondeterministic local transition function $\delta_{nd} : (A \cup \{\#\}) \times S^2 \to 2^S$ and a deterministic local transition function $\delta_d : (A \cup \{\#\}) \times S^2 \to S$. The number of allowed nondeterministic transitions is given by a mapping $g : \mathbb{N} \to \mathbb{N}$ dependent on the length n of the input, such that the first $g(n)$ transitions of the communication cell are nondeterministic according to δ_{nd}, and all subsequent transitions are deterministic according to δ_d. The fact that the nondeterministic transitions have to be applied before the deterministic ones is not a serious restriction since nondeterministic transitions for later time steps can be guessed and stored in advance.

Now, let M be an LBA, $r \in o(\log)$ and r^{-1} be an IA-constructible function, and define a function h_r as $h_r(n) = 2^{r(n)}$ and a language

$$\tilde{L}_{r,M} = \{\, \$^{l+1} w_1 \$ w_2 \$ \cdots \$ w_j ¢ y \mid \exists\, n \geq 1 : j = h_r(n), l = n - (j+1)\cdot(|y|+1),$$
$$y \in \text{VALC}(M), w_i \in \Lambda_M^{|y|}, 1 \leq i \leq j, \text{ and } \exists\, 1 \leq i \leq j : y = w_i^R \,\}.$$

Clearly, function h_r is increasing since r is. Moreover, since $r \in o(\log)$, it follows $\lim_{n\to\infty} \frac{h_r(n)}{n^k} = \lim_{n\to\infty} \frac{2^{r(n)}}{2^{\log(n)\cdot k}} = 0$ and, thus, $h_r(n) \in o(n^k)$, for all $k \in \mathbb{Q}$, $0 < k$. Therefore, the function $\frac{n}{2h_r(n)}$ is unbounded and for all m there is an n so that $m+1 \geq \frac{n}{2h_r(n)} \geq m$, which implies $n \geq m \cdot 2 \cdot h_r(n) \geq (h_r(n)+1)\cdot(m+1)$, for m large enough. This in turn implies that $\tilde{L}_{r,M}$ includes at least one word for every word in VALC(M).

Lemma 7. *Let M be an LBA and $r_1, r_2 : \mathbb{N} \to \mathbb{N}$ be two increasing functions so that $r_2 \in o(r_1)$, $r_1 \in o(\log)$, and r_1^{-1} is IA-constructible. Then $\tilde{L}_{r_1,M} \in \mathscr{L}_{rt}(r_2\,G\text{-}IA)$ if and only if $L(M)$ is finite.*

Similar as in Section 3, now Theorem 3 can be applied to show the non-recursive trade-off between iterative arrays with $r_1(n)$ and $r_2(n)$ guesses as mentioned above.

Theorem 8. *Let $r_1, r_2 : \mathbb{N} \to \mathbb{N}$ be two increasing functions so that $r_2 \in o(r_1)$, $r_1 \in o(\log)$, and r_1^{-1} is IA-constructible. Then the trade-off between real-time r_1 G-IA and real-time r_2 G-IA is non-recursive.*

5 Dimensions

In this section, we will show that there exist non-recursive trade-offs between $(d+1)$-dimensional and d-dimensional real-time iterative arrays. To define a multidimensional IA we adopt Definition 1 by replacing the local transition functions δ and δ_0 by $\delta : S^{2d+1} \to S$ satisfying $\delta(s_0, s_0, \ldots, s_0) = s_0$, and $\delta_0 : (A \cup \{\#\}) \times S^{d+1} \to S$. Configurations of d-dimensional IAs (denoted by IA^d) and the global transition function are straightforwardly defined, where c_t is now a mapping from $\mathbb{N}^d$ to S.

In [17], a dimension hierarchy is shown for IAs with restricted communication. Let us first describe the witness languages used there for the dimension hierarchy. We will then modify these languages suitably to obtain non-recursive trade-offs between real-time IA^{d+1} and real-time IA^d, for all $d \geq 1$.

For any dimension $d \geq 2$, a language L_d is defined as follows. First, consider the following series of regular sets: $X_1 = \$\{a,b\}^+$ and $X_{i+1} = \$X_i^+$, for $i \geq 1$ Due to the separator symbol \$, every word $u \in X_{i+1}$ can uniquely be decomposed into its subwords from X_i. So, the projection on the jth subword can be defined as usual: Let $u = \$u_1 \cdots u_m$, where $u_j \in X_i$, for $1 \leq j \leq m$. Then $u[j]$ is defined to be u_j, if $1 \leq j \leq m$, otherwise $u[j]$ is undefined. Now define the language

$$\begin{aligned} M_d = \{\, & u¢e^{x_d}\$ \cdots \$e^{x_1}\$e^{2x}\$v \mid u \in X_d \text{ and } 1 \leq x_i, 1 \leq i \leq d, \\ & \text{and } x = x_1 + \cdots + x_d \text{ and } v = u[x_d][x_{d-1}] \cdots [x_1] \text{ is defined} \,\}. \end{aligned}$$

Finally, the language L_d is defined as the homomorphic image of M_d using a suitable homomorphism h. It is shown in [17] that L_{d+1} can be accepted by a $(d+1)$-dimensional real-time IA with restricted communication, but not by any d-dimensional real-time IA.

Now, we modify the set M_d in such a way that the prefix u is interleaved symbol by symbol with some word from the set VALC(M) for some LBA M. The remaining symbols ¢, \$, e and the suffix $v \in \{a, b\}$ are repeated once. Additionally, the prefix u may end with some dummy symbols c. Thus, we obtain

$$\begin{aligned} L_{d,M} = \{\, & u_1w_1u_2w_2 \cdots u_tw_t¢¢e^{2x_d}\$\$ \cdots \$\$e^{2x_1}\$\$e^{4x}\$\$vv \mid u_j \in \{a, b, c, \$\}, \\ & w_j \in \varLambda_M, 1 \leq j \leq t, u = u_1u_2 \cdots u_t \in X_d\, c^*, w = w_1w_2 \cdots w_t \in \text{VALC}(M), \\ & 1 \leq x_i, 1 \leq i \leq d, x = x_1 + \cdots + x_d, \text{ and } v = u[x_d][x_{d-1}] \cdots [x_1] \text{ is defined} \,\}. \end{aligned}$$

Lemma 9. *Let M be an LBA and $d \geq 1$ be a constant. Then $L_{d+1,M} \in \mathscr{L}_{rt}(IA^d)$ if and only if $L(M)$ is finite.*

Lemma 10. *Let M be an LBA and $d \geq 1$ be a constant number. Then language $L_{d+1,M}$ belongs to $\mathscr{L}_{rt}(IA^{d+1})$.*

Lemma 9, Lemma 10, and Theorem 3 show the following result.

Theorem 11. *Let $d \geq 1$ be a constant number. Then the trade-off between real-time IA^{d+1} and real-time IA^d is non-recursive.*

6 Communication

Now we turn to consider d-dimensional real-time IA^d with restricted communication. Basically, these are real-time IA^d whose bandwidth of the inter-cell communication links is limited. In the general case, in every step the states of the cells are transmitted. Here the limitation is modeled by a set of messages that can be sent, where the number k of different messages is independent of the number of states. We denote these devices by IA^d_k. Formally, we add to the definition the set of messages B and communication functions $b_i : S \to B$, $1 \leq i \leq 2d$, that determine the messages to be sent to neighbors. Furthermore, we replace δ and δ_0 by $\delta : S \times B^{2d} \to S$ satisfying $\delta(s_0, (b_1(s_0), b_2(s_0), \ldots, b_{2d}(s_0)) = s_0$, and $\delta_0 : (A \cup \{\#\}) \times S \times B^d \to S$.

It is shown in [17] that for real-time IA^d with $d \geq 1$ there exists a proper hierarchy dependent on the number of possible messages, that is, real-time IA^d

that can communicate $k + 1$ different messages are more powerful than those that can communicate k different messages, for $k \geq 1$. For $d \geq 1$ and any number of messages $k \geq 2$ we define an alphabet $A_{d,k} = \{a_0, \dots, a_{k^d-1}\}$ and a language $\hat{L}_{d,k}$ as

$$\begin{aligned}\hat{L}_{d,k} = \{\, & e^x\$u_1u_2\cdots u_m \mid x \geq 1 \text{ and } m \geq 2x-1 \\ & \text{and } u_i \in A_{d,k}, 1 \leq i \leq m, \text{ and } u_j = u_{j+2x-1}, 1 \leq j \leq m-(2x-1)\,\}.\end{aligned}$$

Then, it is shown [17] for $d \geq 1$ and $k \geq 2$ that $\hat{L}_{d,k+1}$ belongs to $\mathscr{L}_{rt}(\mathrm{IA}^d_{k+1})$, but cannot be accepted by any real-time IA^d_k. Now, we modify these languages in order to obtain non-recursive trade-offs. For a constant $c \geq 1$ and an alphabet A, let h_c be the homomorphism $h_c(a) = a^c$, for all $a \in A$. For an LBA M, let S be the state set of the real-time IA accepting VALC(M) constructed in Theorem 4. Now set $c = \lceil \log_2(|S|) \rceil$ and consider the set $h_c(\mathrm{VALC}(M))$. It has been shown in [16] that a real-time IA_2 accepting $h_c(\mathrm{VALC}(M))$ can effectively be constructed. The main idea is to simulate one transition of the IA accepting VALC(M) by c transitions. During these transitions the binary encoded states of the cells are communicated. A cell that receives the cth bit with the cth transition, then simulates the original transition. Finally, the communication cell can verify whether each input symbol is repeated c times. With these prerequisites, we define the languages

$$\begin{aligned}L_{d,k,M} = \{\, & ew_1ew_2\cdots ew_x\$^{2x}u_1u_2\cdots u_m \mid x \geq 1 \text{ and } m \geq 2x-1 \\ & \text{and } u_i \in A_{d,k}, 1 \leq i \leq m, \text{ and } u_j = u_{j+2x-1}, 1 \leq j \leq m-(2x-1), \\ & w_t \in A_M, 1 \leq t \leq x, \text{ and } w_1w_2\cdots w_x \in h_{c+1}(\mathrm{VALC}(M))\,\}.\end{aligned}$$

Lemma 12. *Let M be an LBA and $d, k \geq 1$ be constants. Then language $L_{d,k+1,M}$ belongs to $\mathscr{L}_{rt}(IA^d_k)$ if and only if $L(M)$ is finite.*

Lemma 13. *Let M be an LBA and $d, k \geq 1$ be constant numbers. Then language $L_{d,k+1,M}$ belongs to $\mathscr{L}_{rt}(IA^d_{k+1})$.*

Lemma 12, Lemma 13, and Theorem 3 show the following result.

Theorem 14. *Let $d, k \geq 1$ be constant numbers. Then the trade-off between real-time IA^d_{k+1} and real-time IA^d_k is non-recursive.*

References

1. Beyer, W.T.: Recognition of topological invariants by iterative arrays. Tech. Rep. TR-66. MIT, Cambridge, Proj. MAC (1969)
2. Buchholz, T., Klein, A., Kutrib, M.: Iterative Arrays with Small Time Bounds. In: Nielsen, M., Rovan, B. (eds.) MFCS 2000. LNCS, vol. 1893, pp. 243–252. Springer, Heidelberg (2000)
3. Buchholz, T., Klein, A., Kutrib, M.: Iterative arrays with limited nondeterministic communication cell. In: Words, Languages and Combinatorics III, pp. 73–87. World Scientific Publishing (2003)

4. Buchholz, T., Kutrib, M.: Some relations between massively parallel arrays. Parallel Comput. 23, 1643–1662 (1997)
5. Buchholz, T., Kutrib, M.: On time computability of functions in one-way cellular automata. Acta Inform. 35, 329–352 (1998)
6. Chang, J.H., Ibarra, O.H., Palis, M.A.: Parallel parsing on a one-way array of finite-state machines. IEEE Trans. Comput. C-36, 64–75 (1987)
7. Cole, S.N.: Real-time computation by n-dimensional iterative arrays of finite-state machines. IEEE Trans. Comput. C-18, 349–365 (1969)
8. Fischer, P.C.: Generation of primes by a one-dimensional real-time iterative array. J. ACM 12, 388–394 (1965)
9. Goldstine, J., Kappes, M., Kintala, C.M.R., Leung, H., Malcher, A., Wotschke, D.: Descriptional complexity of machines with limited resources. J. UCS 8, 193–234 (2002)
10. Gruber, H., Holzer, M., Kutrib, M.: On measuring non-recursive trade-offs. J. Autom., Lang. Comb. 15, 107–120 (2010)
11. Holzer, M., Kutrib, M.: Descriptional complexity – An introductory survey. In: Scientific Appl. of Language Methods, pp. 1–58. Imperial College Press (2010)
12. Ibarra, O.H., Jiang, T.: On one-way cellular arrays. SIAM J. Comput. 16, 1135–1154 (1987)
13. Ibarra, O.H., Palis, M.A.: Some results concerning linear iterative (systolic) arrays. J. Parallel Distributed Comput. 2, 182–218 (1985)
14. Ibarra, O.H., Palis, M.A.: Two-dimensional iterative arrays: Characterizations and applications. Theoret. Comput. Sci. 57, 47–86 (1988)
15. Kutrib, M.: The phenomenon of non-recursive trade-offs. Int. J. Found. Comput. Sci. 16, 957–973 (2005)
16. Kutrib, M., Malcher, A.: Computations and decidability of iterative arrays with restricted communication. Parallel Process. Lett. 19, 247–264 (2009)
17. Kutrib, M., Malcher, A.: Cellular automata with limited inter-cell bandwidth. Theoret. Comput. Sci. 412, 3917–3931 (2011)
18. Malcher, A.: On the descriptional complexity of iterative arrays. IEICE Trans. Inf. Syst. E87-D(3), 721–725 (2004)
19. Malcher, A., Mereghetti, C., Palano, B.: Sublinearly space bounded iterative arrays. Int. J. Found. Comput. Sci. 21, 843–858 (2010)
20. Mazoyer, J., Terrier, V.: Signals in one-dimensional cellular automata. Theoret. Comput. Sci. 217, 53–80 (1999)
21. Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars, and formal systems. In: Symposium on Switching and Automata Theory, SWAT 1971, pp. 188–191. IEEE (1971)
22. Seiferas, J.I.: Linear-time computation by nondeterministic multidimensional iterative arrays. SIAM J. Comput. 6, 487–504 (1977)
23. Smith III, A.R.: Real-time language recognition by one-dimensional cellular automata. J. Comput. System Sci. 6, 233–253 (1972)
24. Stearns, R.E.: A regularity test for pushdown machines. Inform. Control 11, 323–340 (1967)
25. Terrier, V.: On real time one-way cellular array. Theoret. Comput. Sci. 141, 331–335 (1995)
26. Valiant, L.G.: Regularity and related problems for deterministic pushdown automata. J. ACM 22, 1–10 (1975)