I F I G
RESEARCH
REPORT

INSTITUT FÜR INFORMATIK

On time reduction and simulation in cellular spaces

Thomas Buchholz   Andreas Klein
Martin Kutrib

IFIG Research Report 9703
December 1997

Institut für Informatik
JLU Gießen
Arndtstraße 2
D-35392 Giessen, Germany
Tel: +49-641-99-32141
Fax: +49-641-99-32149
mail@informatik.uni-giessen.de
www.informatik.uni-giessen.de

JUSTUS-LIEBIG-
UNIVERSITÄT
GIESSEN

IFIG Research Report
IFIG Research Report 9703, December 1997

On time reduction and simulation in cellular spaces

Thomas Buchholz [1] Andreas Klein
Martin Kutrib [2]

Institute of Informatics, University of Giessen
Arndtstr. 2, D-35392 Giessen, Germany

Abstract. We prove a generalized and corrected version of the time reduction theorem for cellular spaces. One basic tool for investigations in this field is the concept of simulation between systems of automata. We propose a general formal definition of the intuitive concept and relate it to the notions which appear in the literature.

[1] E-mail: buchholz@informatik.uni-giessen.de
[2] E-mail: kutrib@informatik.uni-giessen.de

1 Introduction

Systems of (homogeneous) interconnected parallel acting automata are an interesting field for investigations. Besides they are interesting of its own they have a lot of applications. Studies have been done e.g. in view of systems that are capable of nontrivial selfreproduction [24], systems that are models for parallel computers or real phenomena (cf. [15]).

The specification of such a system includes the type and specification of the single automata (these are in almost all cases finite or pushdown automata), the interconnection scheme (which sometimes imply a dimension of the system), a local and/or global transformation and the input and output modes.

One kind of system is of special interest: the *cellular spaces*. In this well-investigated model homogeneously connected deterministic finite automata work synchronously at discrete time steps. The single automata, often called cells, are identified by vectors from $\mathbb{Z}^d$ for a fixed $d \in \mathbb{N}$. The state set includes a so-called quiescent state q_0 such that a cell remains in state q_0 if all of its neighbours are in state q_0. At initial time a finite connected subset $S \subset \mathbb{Z}^d$ of cells, which are the only non-quiescent ones, are carrying the input.

A conceptual problem arises with the end of computations. From the definitions it follows that the system will never halt. A common way of defining final configurations is to define a predicate these configurations have to fulfil and, additionally, to require that final configurations are stable in some sense, i.e. a final configuration leads always to a final configuration.

The most popular kinds of computations are concerned with pattern transformation and manipulation or even formal language processing (e.g. [3, 7, 14, 19]).

Nowadays the investigations in the field of cellular spaces theory are often restricted to spaces with a certain normalized neighbourhood (the von-Neumann neighbourhood). This restriction should be not a strict one since for some cases it has been shown that a given space can be simulated by a normalized one, eventually even with a speed-up [17]. Unfortunately, the theorem is not fully correct. Fortunately, as far as we know other results obtained with the help of the theorem are from areas in which the theorem holds (e.g. [2, 18]).

The main object of the present paper is to prove a generalized and corrected version of the time reduction theorem. Since such considerations are based on simulations and due to the fact that sometimes constructions given in proofs do not meet their definition of simulation [2, 17] we will propose a general definition of what one might have in mind if an automata system simulates another one. We will discuss our notions and relate them to notions which appear in the literature.

2 Simulations

In order to formalize the intuitive notion of simulation we have clearly to define the systems under consideration which, of course, leads to restrictions. On the other hand, there is a natural interest in having feasible formalizations as general as possible.

In the following we are going to propose notions of simulation between systems of automata. The only assumptions on these systems are that they have to work at discrete time steps according to a global transition function and that at every time step their overall state is described by a so-called configuration.

In order to be not too restrictive we are interested in simulations that relate particular configurations rather than sequences of configurations. Another notion which forces the simulating system to use essentially the same algorithm as the simulated system is defined in [5]. There the similarity of space-time diagrams is under consideration which leads to the so-called topological simulation.

At the end of some computation a system should have produced the result. Generally, the result will be a configuration but sometimes we will have to interpret that configuration to obtain the answer we are waiting for. Moreover, sometimes different configurations will give the same answer (e.g., at language recognition we like to get the answer 'yes' or 'no' but there may be more than two computations producing different resulting configurations). Since such interpretations are strongly dependent on the system and the problem to solve, we do not provide a full interpretation but an equivalent relation where each equivalent class contains exactly the configurations that give the same answer.

For a given equivalent relation I we denote the set of equivalent classes by $[I]$ and a class with member c by $[c]$.

In the sequel A resp. B always denote systems with (global) transitions τ resp. τ' and configuration sets C resp. C'. All mappings are assumed to be effectively computable.

Definition 1 *Let $I \subseteq C \times C$ be an equivalent relation on the set of configurations of system A, and let $\forall\, c \in C : \phi_c : \mathbb{N} \to \mathbb{N}$ be a strictly increasing partial function. System B simulates system A with respect to I and the ϕ_c if and only if there exist an injective mapping $f : C \to C'$, a mapping $g : C' \to [I]$ such that the following holds:*

$$\forall\, c \in C : \forall\, t \in dom(\phi_c) : \quad \tau^t(c) \in g({\tau'}^{\phi_c(t)}(f(c)))$$

f may be seen as the encoding function which transforms an initial configuration c to the corresponding one in the simulating system. The time steps at which configurations are related via the decoding function g are

$$
\begin{tikzcd}
c_0 \arrow[r, "\tau^{t_1}"] \arrow[d, "f"'] & c_1 \in [c_1] \arrow[r, "\tau^{t_2 - t_1}"] & c_2 \in [c_2] \arrow[r, "\tau^{t_3 - t_2}"] & c_3 \in [c_3] \arrow[r, "\tau^{t_4 - t_3}"] & {} \\
c'_0 \arrow[r, "\tau'^{\phi_{c_0}(t_1)}"] & c'_1 \arrow[u, "g"] \arrow[r, "\tau'^{\phi_{c_0}(t_2) - \phi_{c_0}(t_1)}"] & c'_2 \arrow[u, "g"] \arrow[r, "\tau'^{\phi_{c_0}(t_3) - \phi_{c_0}(t_2)}"] & c'_3 \arrow[u, "g"] \arrow[r, "\tau'^{\phi_{c_0}(t_4) - \phi_{c_0}(t_3)}"] & \cdots
\end{tikzcd}
$$

Figure 1: Illustration of a simulation.

defined by the mapping ϕ_c. The function f must not be too powerful compared to τ and τ' such that the computation will be performed by the automata system and not by the encodings. It is natural at least to require f to be injective.

Due to the mappings ϕ_c the ratio of the time steps between two consecutive related configuration pairs is not necessarily a constant as is e.g. in [1, 6, 16, 17, 23, 26]. Instead it may depend on the initial configuration (for all $c \in C$ there must exist a ϕ_c) or the elapsed time.

To circumvent certain problems with simulations in [8] the so-called R-simulation has been introduced. It is a weak concept since only the results of the computations are related. Whether or not a configuration is a result has to be determined algorithmically. (In our terms R-simulation is covered by the cases where $\forall\, c \in C : |dom(\phi_c)| = 1$.)

Example 2 The famous open $3x + 1$-problem is the question whether there exists for all $x \in \mathbb{N}$ a $d \in \mathbb{N}$ such that $D^d(x) = 1$, where $D : \mathbb{N} \to \mathbb{N}$ is a mapping defined as follows:

$$
D(x) = \begin{cases} 3x + 1 & \text{if } x \text{ is odd} \\ \frac{x}{2} & \text{otherwise} \end{cases}
$$

According to the problem we define the recursively enumerable formal language $L_D := \{\mathtt{a}^x \mid \exists\, d \in \mathbb{N} : D^d(x) = 1\}$.

A Turing machine $\mathcal{T}$ which accepts L_D may work as follows:

Starting with the word $\mathtt{a}^x$ on its tape it computes one application of D. Subsequently it checks whether the new tape content is the word $\mathtt{a}$ or not and moves the read/write head to the leftmost non-blank square. If the check succeeds it stops in an accepting state otherwise it repeats the computation. (Note: the computation will never halt if there does not exist a appropriated d.) The time needed by $\mathcal{T}$ to compute one application of D depends on x. In both cases it is possible to perform the task in $\mathcal{O}(x^2)$ time steps, say in exactly $k \cdot x^2$, $k \in \mathbb{N}$ constant, time steps.

Now we want to simulate $\mathcal{T}$ by a cellular space $\mathcal{S}$ and additionally we wish to be able to recompute the initial value x. For that $\mathcal{S}$ simply can store a symbol $+$ or $-$, dependent on the parity of x, for every application of D.
Even if we provide a fssp synchronization before every application of D

the space can perform one application in $\mathcal{O}(x)$ time steps, say in exactly $l \cdot x$, $l \in \mathbb{N}$ constant, time steps.

Let us consider the elements of the simulation in detail.
The equivalent relation consists of two classes: accepting and non-accepting configurations. They correspond to the two possible results we may expect at language recognition.

The mapping f is obvious: every cell gets the content of a square of the tape as its state. Since at initial time $\mathcal{T}$ is always in the starting state f is injective.

g maps a configuration of $\mathcal{S}$ to the final configuration of $\mathcal{T}$ if it consists of just one symbol a, to a non-final configuration otherwise. Thereby each square gets the symbol that is the state of the corresponding cell. The read/write head is placed at the leftmost non-blank square and set to a final resp. to the starting state. The extra information stored for reversibility is ignored. Observe that due to the extra information g may be not injective.

Let x_c be the number of as in the configuration c corresponding to the word a^x.

$$\forall\, c \in C : dom(\phi_c) = \left\{ \sum_{i=0}^{n} k \cdot (D^i(x_c))^2 \mid n \in \mathbb{N}_0 \right\}$$

and

$$\phi_c\left(\sum_{i=0}^{n} k \cdot (D^i(x_c))^2 \right) = \sum_{i=0}^{n} l \cdot D^i(x_c).$$

Obviously, the decoding function g can not be chosen to be injective. Even if the classes of the equivalent relation are singletons g is often not injective since there might be parts of computation in the simulating system which have no counterparts in the simulated system. E.g. in [22] an additional dimension is added to make the computation reversible.

If we set $\phi_c(t) = t$, $dom(\phi_c) = \mathbb{N}$ for all $c \in C$ then we obtain the simulation defined in [6]. There a generalization to reversible computations is considered also. For that (in our terms) $dom(\phi_c)$ is set to $\mathbb{Z}$, which requires τ and τ' to be invertible. However, since $\phi_c(t) = t$ the simulations are restricted to real-time.

Naturally, the time factor plays an important role. Since in definition 1 no further assumptions are made for the domain and range of ϕ_c it might be difficult to determine the overall simulation time. If it is possible to choose the points in time such that the differences between successive ones are constant, that becomes an easy task.

Definition 3 *Let system B simulate system A, and let for all $c \in C$ $\{t_1^c, \ldots, t_i^c, \ldots\}$ be the domain of ϕ_c in ascending order.*

a) The simulation is ultimately uniform if and only if $\forall\, c \in C$:

$$|dom(\phi_c)| > 1 \text{ and } \exists\, j < |dom(\phi_c)| : \forall\, i > j :$$

$$(t_i^c - t_{i-1}^c = t_{j+1}^c - t_j^c \wedge \phi_c(t_i^c) - \phi_c(t_{i-1}^c) = \phi_c(t_{j+1}^c) - \phi_c(t_j^c)).$$

b) Define for all $c \in C$: $t_0^c = 0$ and $\phi_c(0) = 0$. The simulation is uniform if and only if it is ultimately uniform for $j = 0$.

$$\begin{tikzcd}
c_0 \arrow[r, "\tau^{t_1}"] \arrow[d, "f"'] & c_1 \in [c_1] \arrow[r, "\tau^{t_1}"] & c_2 \in [c_2] \arrow[r, "\tau^{t_1}"] & c_3 \in [c_3] \arrow[r, "\tau^{t_1}"] & \cdots \\
c_0' \arrow[r, "\tau'^{\phi_{c_0}(t_1)}"] & c_1' \arrow[u, "g"] \arrow[r, "\tau'^{\phi_{c_0}(t_1)}"] & c_2' \arrow[u, "g"] \arrow[r, "\tau'^{\phi_{c_0}(t_1)}"] & c_3' \arrow[u, "g"] \arrow[r, "\tau'^{\phi_{c_0}(t_1)}"] & {}
\end{tikzcd}$$

Figure 2: Illustration of a uniform simulation.

The definition of uniformity of simulations strengthens the notion. One can expect that the algorithms in the simulating and simulated systems are closer related.

The cases $t_1 = T \cdot k$ and $\phi_c(t_1) = T$ are of particular interest. They mark a k-fold speed-up and the simulation is called to be done in $1/k$ times real-time.

Ultimately uniform simulations are induced by techniques of algorithm design in this field which require the preprocessing of inputs by the simulating system itself. Afterwards the 'main' simulation can take place in a uniform manner. A common technique is the space compression of configurations [13]. Another example is the computation of the encoding function by the simulating system itself. (In this case the mapping f is the identity.)

A picturesque example is the simulation of a cellular space by an *iterative array* (IA). An iterative array is simply a cellular space where the input is supplied sequentially to the cell at the origin. All other cells are initially quiescent. It is known that in case of real-time language recognition cellular spaces are more powerful than iterative arrays [20].

Nevertheless in a first phase an IA can read the input whereby the symbols are stored in consecutive cells. After a subsequent fssp synchronization the preprocessing is done and the IA simulates the cellular space one to one.

The following example is due to [17].

Example 4 An *one-way cellular space* (OCS) is a cellular space where each cell is connected to its right immediate neighbour only. Obviously, the flow of information is restricted to one-way from right to left. An OCS can simulate an CS in two times real-time according to the following:

Consider cell x with neighbours $x-1$ and $x+1$ in CS. In order to simulate one state change of x in an OCS there must be a cell which can collect the information (states) of the cells $x-1$, x and $x+1$. Due to the restricted neighbourhood cell $x-1$ can get the state of cell x in one and the states of cell $x+1$ in another one time step. Thus each cell x in CS is simulated by cell $x-1$ in OCS. For simplicity we may assume that a cell $x-1$ becomes quiescent if the cells x and $x+1$ are quiescent.

Here we have $f=$ identity, $g(c'_k)$ is simply a k-translation to the right, $\forall\, c \in C : dom(\phi_c) = \mathbb{N}$ and $\phi_c(n) = 2n$. The simulation is uniform for $t_1 = 1$ and $\phi_c(t_1) = 2$.

The simulated configuration of the CS in the example above shifts in time through the space in the OCS. That is the reason why g must not be required to be injective. If CS gets into a loop the corresponding configurations in OCS differ in their position relative to the origin.

Another concept which strengthens simulation is given in the following definition.

Definition 5 *Let system B simulate system A. The simulation is strong if and only if*

$$\forall\, c \in C : |[c]| = 1 \wedge dom(g) = range(f) \wedge g(f(c)) = [c].$$

$$\begin{tikzcd}
c_0 \arrow[r, "\tau^{t_1}"] \arrow[d, "f"'] & c_1 \arrow[r, "\tau^{t_1}"] \arrow[d, "f"'] & c_2 \arrow[r, "\tau^{t_1}"] \arrow[d, "f"'] & c_3 \arrow[r, "\tau^{t_1}"] \arrow[d, "f"'] & {} \\
c'_0 \arrow[r, "\tau'^{\phi_{c_0}(t_1)}"] & c'_1 \arrow[r, "\tau'^{\phi_{c_0}(t_1)}"] & c'_2 \arrow[r, "\tau'^{\phi_{c_0}(t_1)}"] & c'_3 \arrow[r, "\tau'^{\phi_{c_0}(t_1)}"] & {}
\end{tikzcd} \quad \cdots$$

Figure 3: Illustration of a strong uniform simulation.

One of the aspects of strongness in simulations is the injectivity of the function g (which follows immediately from the definition). With that we are going to compare strong and uniform simulations.

Since the shifts of configurations in example 4 require g not to be injective the presented simulation is not strong (but uniform). Generally, strong simulations do not allow shifting configurations.

Now let us consider example 2, where the non-uniformity can immediately be seen from the equations defining ϕ_c. Since due to the extra information g can not be injective the simulation is neither uniform nor strong.

On the other hand, if we do not provide the mechanism to recompute the initial value x then g is an injective function. (In fact, the mechanism is the only reason why g is not injective.) Moreover, a closer look at f and g lets one observe that the simulation is now strong (but not uniform).

The simulations defined in [17] are in our terms covered by simulations which are both strong and uniform with $dom(\phi_c) = \mathbb{N}$. Several examples may be obtained from the time reduction theorem in section 4.

In order to classify a simulation more precisely we need to express whether it is complete (i.e., whether the whole computation is simulated). Therefore we have to distinguish between finite and infinite computations. Moreover, there are at least two more natural characteristics for which the denotations are defined below.

Definition 6 *Let system B simulate system A.*

a) *The simulation is complete if and only if $\forall\, c \in C : |dom(\phi_c)| = \infty$ or the computation of A is finite, takes t_f time steps, and $\phi_c(t_f)$ is defined.*
b) *Let $k \in \mathbb{N} \cup \{\infty\}$ be a constant. The simulation is k-close if and only if $\forall\, c \in C : |dom(\phi_c)| = k$.*
c) *The simulation is direct if and only if $C \subseteq C'$ and f is the identity on C.*

Intuitively, simulations which use essentially the same algorithms are 'closer' than simulations which simply compute the same input-output function. In order to express how close related the computations are we consider the domain of the mapping ϕ_c. If the simulation is 1-close then we have the same input-output function only (e.g. R-simulation [8]). Examples of ∞-close simulations with $dom(\phi_c) = \mathbb{N}$ are given in [9] where cellular spaces are simulated by some kind of restricted Turing machines, the so-called sweeping machines. Thereby for each configuration of the space there is a related configuration of the sweeping machine. In [16] simulations between cellular spaces on Cayley graphs are investigated. The simulation defined there in our terms is covered by simulation which are strong, uniform, ∞-close and complete and where $dom(\phi_c) = \mathbb{N}$ and $\phi_c(t) = T \cdot t$, $T \in \mathbb{N}$ constant, holds.

The notion direct is slightly of another flavour. It is introduced to express that there is no initial encoding necessary. Observe that corresponding simulations may be ∞-close, too. For example, two cellular spaces with different neighbourhoods simulating each other in a straightforward way.

3 Cellular spaces

In this section we shall give a formal definition of cellular spaces and provide some details of their underlying topology.

We denote the integers by $\mathbb{Z}$, the positive natural numbers $\{1, 2, \ldots\}$ by $\mathbb{N}$ and for $\mathbb{N} \cup \{0\}$ we write $\mathbb{N}_0$. $\mathbb{Z}^d$ denotes the set of all d-tuples of elements of $\mathbb{Z}$. The field of fractions of $\mathbb{Z}$ is denoted by $\mathbb{Q}$, and $\mathbb{Q}^d$ denotes the d-dimensional vector space over $\mathbb{Q}$.

Formally, a cellular space is defined as follows.

Definition 7 *A cellular space (CS) is a system $\mathcal{S} = (S, d, N, \sigma, q_0)$, where*

a) S is the finite, nonempty set of states,
b) $d \in \mathbb{N}$ is the dimension,
c) $N = (a_1, \ldots, a_n)$ is the neighbourhood index (of degree n), i.e. an n-tuple of distinct elements of $\mathbb{Z}^d$,
d) $\sigma : S^n \to S$ is the local transition function, and
e) $q_0 \in S$ is the quiescent state for which $\sigma(q_0, \ldots, q_0) = q_0$ holds.

Sometimes it is convenient to consider the set of neighbours instead of the neighbourhood index, which is actually a vector. Since such cases are easily resolved from the context we do not use a different notation.

A *configuration* of $\mathcal{S}$ at time $t \in \mathbb{N}_0$ describes the states of all the cells. It is formalized as mapping $c_t : \mathbb{Z}^d \to S$. The state of cell $x \in \mathbb{Z}^d$ at time $t+1$ is computed according to

$$c_{t+1}(x) = \sigma(c_t(x + a_1), \ldots, c_t(x + a_n)).$$

Let C denote the set of all configurations of $\mathcal{S}$. The local transition function induces a *global transition function* $\tau : C \to C$ describing the evolution of $\mathcal{S}$ dependent on the initial configuration.

In the sequel S, d, N, σ, τ and q_0 are denoting the set of states, dimension, neighbourhood index, local transition function, global transition function and quiescent state of a cellular space $\mathcal{S}$. Similarly $\mathcal{S}'$ consists of a state set S', has dimension d' and so on.

Now the underlying lattice structure given by $\mathbb{Z}^d$ is considered with respect to the neighbourhood.

It is well known that the set $\mathbb{Z}^d$ forms a module over $\mathbb{Z}$ with the usual pointwise sum of d-tuples and multiplication of an integer with a d-tuple. If X is a subset of $\mathbb{Z}^d$, then $\langle X \rangle$ denotes the span of X, i.e. the smallest submodule of Z^d containing X. Clearly,

$$\langle X \rangle = \left\{ \sum_{i=1}^{n} \varepsilon_i x_i \mid n \in \mathbb{N}, x_i \in X, \varepsilon_i = \pm 1 \right\}$$

holds.

It is known that every submodule X of $\mathbb{Z}^d$ has a basis of t, $t \leq d$, elements, i.e. the basis spans X and any element of X is uniquely expressible as a sum of multiples of basis elements [11]. A similar property is well known for all subspaces of a vector space over some field. If $h : X \to Y$ is a module homomorphism, then $h(X)$ denotes the image of h and $ker\, h$ the kernel of h, i.e. the submodule of X consisting of all elements mapped to zero by h. As usual, the factor module of a module X and a submodule Y of X is denoted by X/Y.

For nonempty subsets $X, Y \subseteq \mathbb{Z}^d$ we define their sum $X + Y$ pointwise, and the multiples of X are recursively defined by $1X := X$ and $(k + 1)X := kX + X$ for all positive integers k.

Based on the span of the neighbourhood index we now may classify some cellular spaces.

Definition 8 *A d-dimensional cellular space with neighbourhood index N is weakly connected iff $\langle N \rangle = \mathbb{Z}^d$. It is strongly connected iff $\langle kN \rangle = \mathbb{Z}^d$ for all positive integers k.*

Remark.

a) There exist weakly but not strongly connected cellular spaces: Let $d = 1$ and $N = (-1, 1)$ then $\langle N \rangle = \mathbb{Z}$ but $\langle 2N \rangle = \langle -2, 0, 2 \rangle = 2\mathbb{Z}$.
b) Let $\mathcal{S}$ be a weakly connected cellular space. If its neighbourhood index contains the 0 it is strongly connected since $0 \in N$ implies $N \subseteq kN$ for all positive integers k.

A cellular space that is not weakly connected is sometimes called a *laminated* cellular space. It can be regarded as a disjoint union of independently working copies of some cellular space. For a general investigation of so-called laminations we refer to [25].

Using familiar proof techniques the following two results concerning modules over $\mathbb{Z}$ can be derived. The first pays tribute to the fact that every proper submodule of $\mathbb{Z}^d$ can be extended to a proper submodule having a basis of d elements.

Lemma 9 *Each proper submodule of $\mathbb{Z}^d$ is contained in some proper submodule of $\mathbb{Z}^d$ which is isomorphic to $\mathbb{Z}^d$.*

Proof. Let X be a proper submodule of $\mathbb{Z}^d$. W.l.o.g. we may assume that X has a basis $x_1, \dots, x_t$ of t, $t < d$, elements.

$x_1, \dots, x_t$ are linearly independent elements of $\mathbb{Q}^d$. So $x_1, \dots, x_t$ can be extended to a basis $x_1, \dots, x_t, \bar{x}_{t+1}, \dots, \bar{x}_d$ of $\mathbb{Q}^d$ where $\bar{x}_{t+1}, \dots, \bar{x}_d \in \mathbb{Q}^d$. Now we can easily find a scalar multiple x_i of $\bar{x}_i$, $i = t + 1, \dots, d$ such that $x_1, \dots, x_{d-1}, 2x_d$ are vectors from $\mathbb{Z}^d$ which form a basis of $\mathbb{Q}^d$.

Let Y be the span of $x_1, \dots, x_{d-1}, 2x_d$ in $\mathbb{Z}^d$. Since $x_1, \dots, x_{d-1}, 2x_d$ is a basis of $\mathbb{Q}^d$, every element of Y is uniquely expressible as a sum of multiples of that basis and, hence, is a basis of Y. Since $x_d \notin Y$ and a bijection between that basis of Y and the standard basis of $\mathbb{Z}^d$ can easily be uniquely extended to an isomorphism, Y is a proper submodule of $\mathbb{Z}^d$ which is isomorphic to $\mathbb{Z}^d$ and contains X. □

The second result is somewhat technical. It plays an important role in the next section.

Lemma 10 *Let $h : \mathbb{Z}^{d'} \to \mathbb{Z}^d$ be a homomorphism and K a finite nonempty subset of $\mathbb{Z}^d$ such that $h(\mathbb{Z}^{d'}) + K = \mathbb{Z}^d$. Then $d' \geq d$ and $\mathbb{Z}^d$ and $\mathbb{Z}^{d'}/ker\, h$ are isomorphic. Moreover $d' = d$ if and only if h is injective.*

Proof. Extend h in the unique way to a vector space homomorphism $\bar{h}$ from $\mathbb{Q}^d$ into $\mathbb{Q}^{d'}$. Let $H := h(\mathbb{Z}^{d'})$ and $\bar{H} := \bar{h}(\mathbb{Q}^{d'})$ and suppose $dim\, \bar{H} < d$. Then there exists an $\bar{x} \in \mathbb{Q}^d \setminus \bar{H}$ and consequently we find some scalar multiple x of $\bar{x}$ belonging to $\mathbb{Z}^{d'} \setminus H$. Consider the cosets $cx + H$, $c \in \mathbb{Z}$, of H in $\mathbb{Z}^d$. Since $(c-d)x \in H \subseteq \bar{H}$ if and only if $c = d$ for $c, d \in \mathbb{Z}$, these cosets are distinct. Further $H + K = \mathbb{Z}^d$ implies that every such coset has some representative in K. Since K is finite we have a contradiction. So $d' \geq dim\, \bar{H} \geq d$ and, clearly, $d' = d$ if and only if h is injective.

Since by the homomorphism theorem $\mathbb{Z}^{d'}/ker\, h$ and H are isomorphic it suffices to show that H and $\mathbb{Z}^d$ are isomorphic. Therefore let $x_1, \ldots, x_t$, $t \leq d$, be a basis of H. Obviously, $x_1, \ldots, x_t$ is a basis of $\bar{H}$, too. Since $dim\, \bar{H} \geq d$ it follows $t = d$. Finally, the extension of a bijection of $x_1, \ldots, x_d$ onto the standard basis of $\mathbb{Z}^d$ is an isomorphism mapping H onto $\mathbb{Z}^d$. □

4 Time Reduction

The present section is devoted to the time reduction theorem for cellular spaces. In the notions previously introduced in section 2 time reduction means a strong, uniform, ∞-close and complete simulation of a cellular space by some (other) cellular space. Since the directness is not required the simulating cellular space may start with a configuration that is an (injective) encoding of the initial configuration of the simulated cellular space. Therefore, we distinguish time reduction from speed-up where an acceleration of the computation under exactly the same input must be achieved.

The time reduction theorem was originally shown by Cole [4] for iterative arrays. Later on Smith [17] adapted the idea to cellular spaces. Unfortunately, his proof concentrates on the choice of an appropriate neighbourhood index for the simulating space. It does not regard the necessity to simulate all the cells of the simulated space.

For a simplified discussion we recall the original lemma from [17] in our terms:

Let $\mathcal{S}$ and $\mathcal{S}'$ be d-dimensional cellular spaces. Let h be an injective homomorphism from the additive group $\mathbb{Z}^d$ into $\mathbb{Z}^d$, and let $K \subseteq \mathbb{Z}^d$ be a finite set of points. Define the state set of a cell in $\mathcal{S}'$ at a point x to be the Cartesian product of the state sets of the cells at points in $\{h(x)\} + K$ in $\mathcal{S}$. Then a sufficient condition that a transition function exists for simulation of $\mathcal{S}$ by $\mathcal{S}'$ in $1/k$ times real time is that

$$h(N') + K \supseteq kN + K.$$

Observe, that the encoding f of configurations from $\mathcal{S}$ is implicitly given by the way the state set of $\mathcal{S}'$ is defined.

The following example causes problems. It is based on the possibility that the simulated space need not to be strongly connected.

Example 11 For simplicity we call a cell x of $\mathcal{S}$ *imitated* by $\mathcal{S}'$ if there exists a cell x' of $\mathcal{S}'$ such that $x \in \{h(x')\} + K$.

Now let $\mathcal{S}$ be a cellular space such that $\langle kN \rangle \neq \mathbb{Z}^d$ for some positive integer k, i.e. $\mathcal{S}$ is not strongly connected. Then lemma 9 ensures the existence of an isomorphism h from $\mathbb{Z}^d$ onto a proper submodule of $\mathbb{Z}^d$ that contains kN. Further let $K := \{0\}$ and $N' := h^{-1}(kN)$.

We obtain $h(N') + K \supseteq kN + K$ and, therefore, one might conclude that there exists a local transition function such that $\mathcal{S}'$ simulates $\mathcal{S}$ in the described manner. However, $h(\mathbb{Z}^d) + K = h(\mathbb{Z}^d) \neq \mathbb{Z}^d$ and, hence, $\mathcal{S}'$ imitates not all cells of $\mathcal{S}$.

Now we have the situation that the encoding is not injective (if $|S| > 1$) and, thus, we have no simulation of $\mathcal{S}$ by $\mathcal{S}'$ at all.

Observe, that the simulation defined in [17] requires an injective encoding, too. Even if we would define simulations without the injectivity of encodings there can arise problems. In [10, 12, 21] language recognition is investigated where a configuration is accepting if all of its cells are in an accepting state simultaneously.

Example 12 Define the *stutterer language* L over $\{a, b\}$ as all words in which the symbols at the odd positions are all the same and the symbols at the even positions are all the same. ($abababa, bbb, baba \in L$ but $aab \notin L$.) The cellular space $\mathcal{S} = (S, 1, N, \sigma, q_0)$ with $N = (-2, 0, 2)$ can recognize L in one time step. Every cell accepts iff both of its neighbours are in the same state as the cell itself or in the quiescent state. Otherwise the cells enter a rejecting state.

We now apply the original time reduction lemma with $N' = (-1, 0, 1)$, $h(x) = 2x$, $K = \{0\}$ and $k = 1$. Then all assumptions are fulfilled, but

$\mathcal{S}'$ simulates only the cells at even positions. Therefore, $\mathcal{S}'$ would also accept words like $abaa \notin L$, where all symbols at the even positions are the same but the symbols at the odd positions are different.

The problem in the example arises because $\langle N \rangle = 2\mathbb{Z} \neq \mathbb{Z}$. Now we are going to propose the following corrected and generalized version of the time reduction theorem.

Definition 13 *A strong, uniform, ∞-close and complete simulation where $dom(\phi_c) = k\mathbb{N}$ and $\phi_c(k \cdot t) = t$ for some positive integer k is called a $1/k$-time-reduction.*

Theorem 14 *Let $\mathcal{S}$ and $\mathcal{S}'$ be d- resp. d'-dimensional cellular spaces. Let h be an homomorphism from $\mathbb{Z}^{d'}$ into $\mathbb{Z}^d$, and let K be a finite subset of $\mathbb{Z}^d$. Define the state set of a cell in $\mathcal{S}'$ at a point x' to be the Cartesian product of the state sets of the cells at points in $\{h(x')\} + K$ in $\mathcal{S}$. Then a sufficient condition that a local transition function exists for a $1/k$-time-reduction of $\mathcal{S}$ by $\mathcal{S}'$ is that the time reduction condition*

$$h(N') + K \supseteq kN + K$$

and the covering condition

$$h(\mathbb{Z}^{d'}) + K = \mathbb{Z}^d$$

hold.

Proof. The proof is straightforward if we observe the original proof and the fact that the covering condition clearly implies that the implicitly given encoding is injective. Moreover, by the covering condition all the cells of $\mathcal{S}$ are imitated. □

As mentioned before the problems arise if the simulated space is not strongly connected. Another way to correct the theorem would have been to add that condition to the theorem. But the covering condition is a weaker one:

Define in example 12 $K = \{0, 1\}$. Now the covering condition holds, but the recognizer for the stutterer language is not strongly connected. For the converse we can show the following lemma.

Lemma 15 *If $\mathcal{S}$ is strongly connected then the time reduction condition implies the covering condition.*

Proof. Obviously $h(\mathbb{Z}^{d'}) + K \subseteq \mathbb{Z}^d$. To prove the converse inclusion let $x \in \mathbb{Z}^d$ be some cell of $\mathcal{S}$ and let $x_0 \in K$. Since $\mathcal{S}$ is strongly connected kN spans $\mathbb{Z}^d$. Therefore there exists a sequence $x_0, x_1, \ldots, x_p = x$ such that $x_{i+1} \in x_i + kN$ or $x_i \in x_{i+1} + kN$ for $0 \leq i < p - 1$. (Note that kN need not to be symmetric, i.e. kN may differ from $-kN$.)

Proceeding by induction we show that for each x_i there is an $x'_i \in \mathbb{Z}^{d'}$ such that $x_i \in h(x'_i) + K$. Trivially choosing $x'_0 := 0$ we may now assume that we have found an x'_i for some $0 \leq i < p$. Corresponding to the different possible relations between x_i and x_{i+1} we distinguish the following two cases.

Case 1: $x_{i+1} \in x_i + kN$.

Since $x_i \in h(x'_i) + K$ and the time reduction condition holds it follows

$$x_{i+1} \in x_i + kN \subseteq h(x'_i) + K + kN \subseteq h(x'_i + N') + K.$$

So there exists some $x'_{i+1} \in x'_i + N'$ such that $x_{i+1} \in h(x'_{i+1}) + K$.

Case 2: $x_i \in x_{i+1} + kN$.

Let $y_j := x_i + j(x_i - x_{i+1})$ for $j \in \mathbb{N}_0$. Then $y_{j+1} \in y_j + kN$ and using the same argument as in the previous case we find $y'_j \in \mathbb{Z}^{d'}$ such that $y_j = h(y'_j) + a_j$ for $a_j \in K$ and $j \in \mathbb{N}_0$. Since K is finite there are indices $l < m$ for which $a_l = a_m$. It follows

$$y_m + c(y_m - y_l) = h(y'_m + c(y'_m - y'_l)) + a_m$$

for all integers $c \in \mathbb{Z}$. In particular there exists some positive integer $\bar{c}$ and some $\bar{x}' \in \mathbb{Z}^{d'}$ such that

$$\bar{x} := x_i + \bar{c}(x_{i+1} - x_i) \in h(\bar{x}') + K.$$

Applying the argument of the first case once more to the sequence $\bar{x}, \bar{x} + (x_i - x_{i+1}), \ldots, x_{i+1}$ the existence of an x'_{i+1} with $x_{i+1} \in h(x'_{i+1}) + K$ is ensured.

Since $x = x_p \in h(x'_p) + K$ it follows $\mathbb{Z}^d \subseteq h(\mathbb{Z}^{d'}) + K$, i.e. the covering condition holds. □

In theorem 14 no restrictions on the dimension of $\mathcal{S}$ resp. $\mathcal{S}'$ have been made. It follows from lemma 10 that $d' \geq d$ which means that for time reduction the dimension of the simulating space have to be at least as large as the dimension of the simulated space. The next result shows that it suffices to have the same dimension.

Theorem 16 *Let $\mathcal{S}$ be a d-dimensional cellular space. Then the conditions of the time reduction theorem imply the existence of a d-dimensional cellular space $\mathcal{S}''$ for a $1/k$-time-reduction of $\mathcal{S}$ by $\mathcal{S}''$.*

Proof. By the homomorphism theorem there exists a unique isomorphism μ from $\mathbb{Z}^{d'}/ker\,h$ onto $h(\mathbb{Z}^{d'})$. By lemma 10 there is further a isomorphism ψ from $\mathbb{Z}^d$ onto $\mathbb{Z}^{d'}/ker\,h$. Define η to be the composition of μ and ψ, i.e. $\eta(x) = \mu(\psi(x))$ for all $x \in \mathbb{Z}^d$, and let $N'' := \{\psi^{-1}(x' + ker\,h) \mid x' \in N'\}$ be the neighbourhood index of some d-dimensional cellular space $\mathcal{S}''$. Then clearly $\eta(N'') = h(N)$ and $\eta(\mathbb{Z}^d) = h(\mathbb{Z}^{d'})$. Moreover, the state set of a cell in $\mathcal{S}''$ at a point x might be the Cartesian product of the state sets of the cells at points $\{\eta(x)\} + K$ of $\mathcal{S}$. Since $h(x') = h(y')$ if and only if x' and y' belong to the same coset of $ker\, h$ the state set of a cell in $\mathcal{S}''$ is well defined. An application of the time reduction theorem 14 (replacing h by η and $\mathcal{S}'$ by $\mathcal{S}''$) concludes the proof. $\square$

References

[1] Amoroso, S. and Guilfoyle, R. *Some comments on neighbourhood size for tesselation automata*. Information and Control 21 (1972), 48–55.

[2] Butler, J. T. *A note on cellular automata simulations*. Information and Control 26 (1974), 286–295.

[3] Choffrut, C. and Čulik II, K. *On real-time cellular automata and trellis automata*. Acta Informatica 21 (1984), 393–407.

[4] Cole, S. N. *Real-time computation by n-dimensional iterative arrays of finite-state machines*. IEEE Transactions on Computers C-18 (1969), 349–365.

[5] Čulik II, K. and Yu, S. *Translation of systolic algorithms between systems of different topology*. IEEE 14^{th} International Conference on Parallel Processing, 1985, pp. 756–763.

[6] Durand-Lose, J. O. *Partitioning automata, cellular automata, simulation and reversibility*. Research Report 95-01, Ecole Normale Supérieure de Lyon, Lyon, 1995.

[7] Dyer, C. R. *One-way bounded cellular automata*. Information and Control 44 (1980), 261–281.

[8] Höllerer, W. O. and Vollmar, R. *On forgetful cellular automata*. Journal of Computer and System Sciences 11 (1975), 237–251.

[9] Ibarra, O. H., Kim, S. M., and Moran, S. *Sequential machine characterizations of trellis and cellular automata and applications*. SIAM Journal on Computing 14 (1985), 426–447.

[10] Ibarra, O. H., Palis, M. A., and Kim, S. M. *Fast parallel language recognition by cellular automata*. Theoretical Computer Science 41 (1985), 231–246.

[11] Jacobson, N. *Lectures in Abstract Algebra 2*. Van Nostrand Reinhold, New York, 1953.

[12] Kim, S. and McCloskey, R. *A characterization of constant-time cellular automata computation*. Physica D 45 (1990), 404–419.

[13] Kosaraju, S. R. *On some open problems in the theory of cellular automata*. IEEE Transactions on Computers C-23 (1974), 561–565.

[14] Kutrib, M. *Pushdown cellular automata*. Theoretical Computer Science (1998).

[15] Kutrib, M., Vollmar, R., and Worsch, Th. *Introduction to the special issue on cellular automata*. Parallel Computing 23 (1997), 1567–1576.

[16] Róka, Z. *Simulations between cellular automata on cayley graphs*. Research Report RR 94-40, Ecole Normale Supérieure de Lyon, Lyon, 1994.

[17] Smith III, A. R. *Cellular automata complexity trade-offs*. Information and Control 18 (1971), 466–482.

[18] Smith III, A. R. *Simple computation–universal cellular spaces*. Journal of the ACM 18 (1971), 339–353.

[19] Smith III, A. R. *Two-dimensional formal languages and pattern recognition by cellular automata*. IEEE Conference Record of 12^{th} Annual Symposium on Switching and Automata Theory, 1971, pp. 144–152.

[20] Smith III, A. R. *Real-time language recognition by one-dimensional cellular automata*. Journal of Computer and System Sciences 6 (1972), 233–253.

[21] Sommerhalder, R. and van Westrhenen, S. C. *Parallel language recognition in constant time by cellular automata*. Acta Informatica 19 (1983), 397–407.

[22] Toffoli, T. *Computation and construction universality of reversible cellular automata*. Journal of Computer and System Sciences 15 (1977), 213–231.

[23] Vollmar, R. *Cellular spaces and parallel algorithms*. In Feilmeier, M. (ed.), *Parallel Computers – Parallel Mathematics, IMACS*. North-Holland, 1977, pp. 49–58.

[24] von Neumann, J. *Theory of Self-Reproducing Automata*. edited and completed by Arthur W. Burks. University of Illinois Press, 1966.

[25] Yamada, H. and Amoroso, S. *Tesselation automata*. Information and Control 14 (1969), 299–317.

[26] Yamada, H. and Amoroso, S. *Structural and behavioral equivalences of tesselation automata*. Information and Control 18 (1971), 1–31.