ASYNCHRONOUS AUTOMATA NETWORKS CAN EMULATE ANY SYNCHRONOUS AUTOMATA NETWORK

CHRYSTOPHER L. NEHANIV

Abstract. We show that any locally finite automata network $\mathcal{A}$ with global synchronous updates can be emulated by another one $\widehat{\mathcal{A}}$, whose structure derives from that of $\mathcal{A}$ by a simple construction, but whose updates are made asynchronously at its various component automata (e.g., possibly randomly or sequentially, with or without possible simultaneous updates at different nodes). By "emulatation", we refer to the existence of a spatial-temporal covering ('local time'), allowing one to project the behavior of $\widehat{\mathcal{A}}$ continuously onto that of $\mathcal{A}$. We also show the existence of a spatial-temporal section of the asynchronous automata network's behavior which completely determines the synchronous global state of $\mathcal{A}$ at every time step.

We give the construction of the asynchronous automata network, establish its freedom from deadlocks, and construct local time functions and spatial-temporal sections relating any posssible behavior of $\widehat{\mathcal{A}}$ to the single corresponding behavior of $\mathcal{A}$ on a given input sequence starting from a given initial global state.

This establishes that the behavior of any (locally finite) synchronous automata network actually can be emulated without the restriction of synchronous update, freeing us from the need of a global clock signal. Local information is sufficient to guarantee that the synchronous behavior of $\mathcal{A}$ is completely determined by any asynchronous behavior of $\widehat{\mathcal{A}}$ starting from a corresponding global state and given the same input sequence as $\mathcal{A}$. Moreover, the relative passage of corresponding local time at any two nodes in $\widehat{\mathcal{A}}$ is bounded in a simple way by approximately one-third of the distance between them.

As corollaries, any synchronous generalized cellular automaton or synchronous cellular automaton can be emulated by an asynchronous one of the same type.

Implementation aspects of these asynchronous automata are also discussed, and open problems and research directions are indicated.

1. Introduction and Preliminaries

In this paper we derive a general result which shows how it is possible to emulate the behavior of a given synchronously updated automata network by a corresponding asynchrononous one. This allows one to transfer results concerning the usual (synchronous) automata networks to the asynchronous realm, including

Preprint submitted on 31 January 2003 to *International Journal of Algebra & Computation*. Minor revisions 20 December 2003.

for example cellular automata and their generalizations. Moreover, the result holds also for infinite automata networks over locally finite underlying graphs.

1.1. **Graphs.** A *directed graph* (or *digraph*) $\Gamma = (V, E)$ is a set V of *vertices* and a set of *directed edges* $E \subseteq V \times V$. Elements of V are sometimes called *nodes*. An edge $(w, v) \in E$ is said to have *source* w and *target* v, and to be an *outgoing edge* of w and *incoming edge* to v.

We say node w is a *neighbor* of v if there is an incoming edge from w to v, that is, $(w, v) \in E$. The *neighborhood of* v is the set $N(v) \subseteq V$ of all neighbors of node w.

The associated undirected graph to Γ is $\widehat{\Gamma}$ with the same set of vertices V and edges

$$\widehat{E} = \{(w, v) \in V \times V \mid (w, v) \in E \text{ or } (v, w) \in E \text{ or } v = w\}.$$

Thus $\widehat{\Gamma}$ has as edge set the symmetric and reflexive closure of the relation E.

A *path* of length $n \geq 0$ in Γ from node v to node w is a sequence of vertices $v_0, \ldots, v_n$ with $v = v_0$ and $w = v_n$ such that $(v_i, v_{i+1}) \in E$ for all $0 \leq i < n$.

The digraph Γ is *locally finite* if the associated undirected graph $\widehat{\Gamma}$ has no node with infinitely many neighbors.

The *distance* $d(v, w)$ from node v to node w in $\widehat{\Gamma}$ is the least n such that there is a path of length n from v to w (if such a path exists), or, otherwise ∞, if there is no path from v and w. (Since the edge relation in $\widehat{\Gamma}$ is symmetric, this gives a metric on $\widehat{\Gamma}$.)

1.2. **Synchronous Automata Networks.** We recall the concept of *(synchronous) automata network* $\mathcal{A}$, which is an automaton defined by giving a directed graph $\Gamma = (V, E)$, a V-indexed set of automata $\mathcal{A}^v$, an external input alphabet X, and a V-indexed set of feedback functions that are compatible with Γ:

To each $v \in V$, let an *automaton* $\mathcal{A}^v = (Q^v, X^v, \delta^v : Q^v \times X^v \to Q^v)$ be associated. We say Q^v is the set of local states, X^v is the set of local input letters, and δ^v is the local transition function at node v. If there is no danger of confusion we shall write $q^v \cdot x^v$ for the state $\delta^v(q^v, x^v)$ whenever $q^v \in Q^v$ and $x^v \in X^v$.

A *global state* of the automata network $\mathcal{A}$ is an element q of $Q = \prod_{v \in V} Q^v$. For a vertex $v \in V$, denote by $q^v \in Q^v$ the v-component of q.

Let $X \neq \emptyset$ be an external alphabet, and let $X^{\eth} = X \cup \{\eth\}$ where $\eth \notin X$ may be regarded as as a *Wait* symbol. For each node $v \in V$, let there be a *feedback function*

$$\varphi^v : \prod_{w \in N(v)} Q^w \times X^{\eth} \to X^v.$$

This determines a local input letter $x^v \in X^v$ to $\mathcal{A}^v$ as a function of the external input letter (or *Wait* symbol) $x \in X^{\eth}$ and the state nodes in the neighborhood of v. In the synchronous case, the *Wait* symbol $\eth$ is actually superfluous and X

rather than $X^{\eth}$ may be used throughout as we shall see. It is only required in the asynchronous generalization.

We may extend φ^v to $\varphi^v : Q \times X^{\eth} \to Q^v$ by letting $\varphi^v(q, x) = \varphi^v((q_w)_{w\in N(v)}, x)$. Here $(q_w)_{w\in N(v)} \in \prod_{w\in N(v)} Q^w$ is the assignment of states to all components in the neighborhood of v according to global state q. In this way, φ^v does not really depend on its w-component unless $w \in N(v)$.

Given the digraph $\Gamma = (V, E)$, automata $\{\mathcal{A}^v\}_{v\in V}$, feedback functions $\{\varphi^v\}_{v\in V}$, and external alphabet X as above, the *(synchronous) automata network* $\mathcal{A}$ is an automaton with states $Q = \prod_{v\in V} Q^v$, inputs X, and transition function $\delta : Q \times X \to Q$ defined for all $q \in Q$ and $x \in X$ by giving the new v-component of state $q \cdot x$ as

$$\delta^v(q^v, x) = \delta^v(q^v, \varphi^v(q, x)) = q^v \cdot \varphi^v(q, x),$$

here $q^v \in Q^v$ is the v-component of global state $q \in Q$.

We say $\mathcal{A}$ is the (synchronous) automata network (or a general product or Gluškov product) of local automata $\mathcal{A}^v$ over the digraph Γ according to the feedback functions φ^v.

For all natural numbers $n \in \mathbb{N}$, let x_n be a letter in X. If the sequence $x_1, x_2, x_3, \ldots$ is input to $\mathcal{A}$ in a synchronous network, starting from an initial global state $q_0 \in A$ with $v \mapsto q_0^v$, then the global state q_n of $\mathcal{A}$ at time n is given inductively by

$$q_n^v = q_{n-1}^v \cdot \varphi^v(q_{n-1}, x_n),$$

for all $n \geq 1$. Note that we are using a discrete model of time. Thus the successive states of the local automaton $\mathcal{A}^v$ at node v, $q_0^v, q_1^v, q_2^v, \ldots$ and the successive global states $q_0, q_1, q_2, \ldots$ of the entire network $\mathcal{A}$ depend in general on the particular values of external inputs, except in the case $|X| = 1$. The function $q : \mathbb{N} \to Q$ with q_n having v-component $q_n^v \in Q^v$ is called the *behavior* of the synchronous network $\mathcal{A}$ on the given input sequence $\{x_n\}_{n\in\mathbb{N}}$ and initial state q_0.

Note also that in this synchronous case, a letter is read at each update, so the $Wait$ symbol is never used in place of an input letter by any feedback function φ^v. Thus we could have equivalently used X rather than $X^{\eth}$ in the definition of the φ^v. This is what the classical definition does. $X^{\eth}$ is needed for the general asynchronous case below. In the synchronous case and, as we shall see, for generalized cellular automata, this is equivalent to the classical definition.

1.3. **Asynchronous Automata Networks.** Our concept of *asynchronous automata network* $\mathcal{A}$ requires again giving a directed graph $\Gamma = (V, E)$, a V-indexed set of automata A^v, an external input alphabet X, and a V-indexed set of feedback functions $\{\varphi^v\}_{v\in V}$ that are compatible with Γ. It also requires a V-indexed family of *read functions*

$$\rho^v : \prod_{w\in N(v)} Q^w \to \{Read, Wait\},$$

which are used to determine whether the feedback function for node v receives the *Wait* symbol $\eth$ or a letter of external input. These ingredients completely determine the asynchronous automata network $\mathcal{A}$.

We will allow "local update" at a node v without necessarily changing local state at any other node, and local automata will be allowed to read the global input sequence asynchronously and independently according to their update times and local state in their neighborhoods. In particular, local automata will be allowed to wait (as a function of the state of their local neighborhood) before reading the next letter of external input.

Update Patterns. To capture the notion of asynchronous local updates, embed $\mathbb{N}$ as a model of time arbitrarily into the non-negative real numbers $\mathbb{R}^+$ (or non-negative rationals $\mathbb{Q}^+$, or $\mathbb{N}$):

$$\tau : \mathbb{N} \to \mathbb{R}^+ \text{ (or } \mathbb{Q}^+ \text{ or } \mathbb{N}),$$

with $\tau(0) = 0$, and $i < j$ implying $\tau(i) < \tau(j)$. At time $\tau(n)$ with n positive, a set of local updates will occur simultaneously in the asynchronous automata network. During the half-open interval $[0, \tau(1))$, the state of the automata network is an initial global state q_0 as above. For each $n > 0$, during open interval $(\tau(n), \tau(n+1))$, the state of the network does not change at all. At time $\tau(n)$, let $U_{\tau(n)} \subseteq V$ denote the *nodes updated at time* $\tau(n)$. Formally, a *(local) update* is said to occur at node $v \in V$ at time $\tau(n) \in \mathbb{R}^+$ if and only if v lies in the update set $U_{\tau(n)}$. Thus subsets of the local automata of $\mathcal{A}$ will be updated instantaneously at time points $\tau(1), \tau(2), \ldots$, with all local automata having nodes in the update set $U_{\tau(n)}$ updated simultaneously as a function of the current states of their neighbors and possibly the input letters they are currently reading. We require that each node $v \in V$ is updated an unbounded number of times, i.e. $v \in U_{\tau(n)}$ for infinitely many $n \in \mathbb{N}^+$.

An *update pattern* (τ, U) of an asynchronous network is an order preserving function (as above) $\tau : \mathbb{N} \to \mathbb{R}^+$ together with a family of update sets $U_{\tau(n)} \subseteq V$, $n > 0$. (Sometimes we will suppress the update sets and refer to τ as an update pattern.) For $t \in \mathbb{R}^+ \setminus \tau(\mathbb{N})$ and also for $t = 0$, one may define $U_t = \emptyset$. Then for all moments in time $t \in \mathbb{R}^+$, an update occurs at node v at time t if and only if $v \in U_t$. A *run* of a network is a sequence of global states q_t of the network, and we will soon see how an update pattern together with a infinite input word $\{x_n\}_{n>0}$ (with x_n in X, for all positive natural numbers n) determine a well-defined run $q : \mathbb{R}^+ \to Q$, with component values $q_t^v = (q_t)^v \in Q^v$ at time $t \in \mathbb{R}^+$, called the *(continuous) behavior* of the asynchronous network $\mathcal{A}$ for this update pattern and input sequence, with initial global state q_0. The restriction $q : \tau(\mathbb{N}) \to Q$, of q to $\tau(\mathbb{N})$, is called the *(discrete) behavior* of $\mathcal{A}$, and clearly determines the continuous behavior q on $\mathbb{R}^+$, since nothing in the network may change at any $t \notin \mathbb{R}^+ \setminus \tau(\mathbb{N})$. An update pattern is not a part of the specification of the automata network, and

need not be given in advance. An update pattern and external input sequence are however required in to order determine a behavior of the network.

Local Reading and Waiting. For each node $v \in V$, we assume a next available letter $x_{n^*(v)} \in X$ — *which one will depend on how far* $\mathcal{A}^v$ *has read(!)* — is available at time $\tau(n)$ to be read from the sequence of global external inputs $x_1, x_2, x_3, \ldots$ which are read sequentially but not synchronously by the local automata in the asynchronous network. That is, the letters of external input $x_1, x_2, x_3, \ldots$ are read in sequence at each node v but node v is also permitted to $Wait$ and update itself before reading (or "consuming") a letter. This is why the feedback function must handle the case when the next letter is not to be read yet, i.e. $\varphi^v : Q \times X^{\eth} \to X^v$ where $\eth$, the $Wait$ symbol, is used as the second argument to φ^v when the next input letter is not read.

However, whether or not the next letter at node v is read may depend at most on the states of the local automata at the neighbors of node v, and may not depend on external input letter itself. Thus this is determined, for each node $v \in V$, by the read function $\rho^v : \prod_{w \in N(v)} Q^w \to \{Read, Wait\}$ for node v. Like the feedback functions φ^v, the ρ^v are given when specifying the asynchronous automata network, and may be extended to all global states $\rho^v : Q \to \{Read, Wait\}$ by defining

$$\rho^v(q) = \rho^v((q^w)_{w \in N(v)}).$$

Thus $\rho^v(q)$ does not really depend on the w-component q^w of $q \in Q$ unless $w \in N(v)$.

If $\rho^v(q)$ is $Read$ when $v \in U_{\tau(n)}$ then the next external input letter is passed to the feedback function φ^v; but if $\rho^v(q) = Wait$ when $v \in U_{\tau(n)}$, then $\eth$ is passed to the feedback function, and the external letter remains available. Letters of a copy of the infinite external input sequence are thus consumed in order at every node. Each local automaton gets to consume a copy of the same external input sequence, but the local automaton may wait before consuming the next letter depending on the state of its local neighborhood and the function ρ^v.

Thus at time $t = \tau(n)$ with $n > 0$, if $v \in U(\tau(n))$ and the next letter locally available for reading is $x \in X$, a local update of state at node v is given by:

$$q^v_{\tau(n)} = \begin{cases} q^v_{\tau(n-1)} \cdot \varphi^v(q_{\tau(n-1)}, x) \text{ if } \rho^v(q_{\tau(n-1)}) = Read \\ q^v_{\tau(n-1)} \cdot \varphi^v(q_{\tau(n-1)}, \eth) \text{ if } \rho^v(q_{\tau(n-1)}) = Wait. \end{cases}$$

Note that the values of φ^v and ρ^v here do not depend on the state of local automata other than the neighbors of $\mathcal{A}^v$ at time $\tau(n-1)$. Since ρ^v is a function, there is no non-determinism in deciding whether or not the next letter is to be read or not for a given state $q \in Q$.

Details of Asynchronous Behavior. To keep track, as external observers, of which letter of the external input sequence is currently available at time $\tau(n)$ at node v, we note that the *index* $n^*(v)$ *to the next letter for node* v *at time* $\tau(n)$ is given inductively by $1^*(v) = 1$ and

$$(n+1)^*(v) = \begin{cases} n^*(v)+1 & \text{if } v \in U_{\tau(n)} \text{ and } \rho^v(q_{\tau(n-1)}) = \mathit{Read} \\ n^*(v) & \text{otherwise.} \end{cases}$$

Thus, starting from the first letter of the global input sequence $\{x_n\}_{n\in\mathbb{N}}$, the index to the next input letter is advanced if and only if the input letter in position $n^*(v)$ has been read when $\mathcal{A}^v$ was last updated. Thus *the next external letter which may be read by the local automaton at node* $v \in V$ at time $\tau(n)$, with n a positive natural number, is denoted $x_{n^*(v)} \in X$.

Up to but not including time $\tau(n)$, the local automaton at v will have consumed $x_1, \ldots, x_{n^*(v)-1}$. Thus $x_{n^*(v)}$ indicates the next letter that the local automaton may consume at time $\tau(n)$. It is important to note that the local automata $\mathcal{A}^v$ do not themselves carry any information on what the next letter will be or where to find it, any more than do standard finite automata reading an input sequence. The notation $x_{n^*(v)}$ merely allows an external observer to describe which is the next letter of the external input sequence that is available to the local automaton.

The updates of state at node v for $n > 0$ are formally described by:

$$q^v_{\tau(n)} = \begin{cases} q^v_{\tau(n-1)} \cdot \varphi^v(q_{\tau(n-1)}, x_{n^*(v)}) & \text{if } v \in U_{\tau(n)} \text{ and } \rho^v(q_{\tau(n-1)}) = \mathit{Read} \\ q^v_{\tau(n-1)} \cdot \varphi^v(q_{\tau(n-1)}, \eth) & \text{if } v \in U_{\tau(n)} \text{ and } \rho^v(q_{\tau(n-1)}) \neq \mathit{Read} \\ q^v_{\tau(n-1)} & \text{otherwise,} \end{cases}$$

where $x_{n^*(v)} \in X$ is the letter in position $n^*(v)$ of the external input sequence.

Recall that no change of state occurs at time t unless $t = \tau(n)$, for some $n > 0$. Therefore at every node $v \in V$, for all times t in the half-open interval $[\tau(n-1), \tau(n))$, where $n > 0$, we have $q^v_t = q^v_{\tau(n-1)}$. Thus the state of node v during this interval, i.e. the state from time $\tau(n-1)$ up to and including any time 'just before' time $\tau(n)$ is exactly $q^v_{\tau(n-1)}$. Given an initial global state q_0, the above update rule determines the state q^v_t of $\mathcal{A}^v$ and hence the state q_t of the entire network for all $t \in \mathbb{R}^+$, so we have a well-defined run, the (continuous) behavior of $\mathcal{A}$.

Note that if external inputs are always read (i.e. $\rho^v(q) = \mathit{Read}$ for all $q \in Q, v \in V$) and $U_{\tau(n)} = V$ for every $n > 0$ then the sequence of global states $q_{\tau(0)}, q_{\tau(1)}, q_{\tau(2)}, \ldots$ is exactly the behavior of a uniquely determined corresponding synchronous automata network.

2. Asynchronous Emulation Theorem

Definition (Spatial-Temporal Covering). Let $\Gamma = (V, E)$ be a directed graph. Then a *spatial-temporal covering* for Γ is any function $\lambda : \mathbb{R}^+ \times V \to \mathbb{N}$ such that following conditions hold:
(1) the restriction $\lambda : \mathbb{R}^+ \times \{v\} \twoheadrightarrow \mathbb{N}$ of λ to every given vertex $v \in V$ is surjective,
(2) λ is locally monotonically increasing, i.e. for all $t, t' \in \mathbb{R}^+$ and $v \in V$,

$$t \geq t' \text{ implies } \lambda(t, v) \geq \lambda(t', v),$$

(3) for all $t, t' \in \mathbb{R}^+$ and $v \in V$,

$$|\lambda(t, v) - \lambda(t'v)| \leq d(v, v'),$$

where d denotes the distance metric in the associated undirected graph $\widehat{\Gamma}$ (the reflexive-symmetric closure of the relation E).

Definition (Emulation).[1] Let $\mathcal{A}$ be an synchronous automata network over a directed graph $\Gamma = (V, E)$ with global state set Q and $\widehat{\mathcal{A}}$ be an asynchronous automata network with the same input alphabet X, a directed graph $\Gamma' = (V, E')$ with the same set of nodes, and global state set $\widehat{Q}$. Let $\pi : \widehat{Q} \to Q$ be a function from global states of the asynchronous automata network to global states of the synchronous one, such that $\pi^v(\hat{q}) = (\pi(\hat{q}))^v$ depends only on $\hat{q}^v$ for all $\hat{q} \in \widehat{Q}$. Thus we can denote $(\pi(\hat{q}))^v$ by $\pi(\hat{q}^v)$.

We then say that the behavior $\hat{q} : \mathbb{R}^+ \to \widehat{Q}$ of $\widehat{\mathcal{A}}$ starting in state $\hat{q}_0$ for update pattern (τ, U) and input sequence $x_1, x_2, \ldots$ ($x_i \in X$ for $i \in \mathbb{N}$) *emulates* the behavior $q : \mathbb{N} \to Q$ of $\widehat{\mathcal{A}}$ starting in state q_0 with the same input sequence under the projection π if there exists a spatial-temporal covering $\lambda : \mathbb{R}^+ \times V \to \mathbb{N}$ such that the following diagram commutes for each $v \in V$:

$$\begin{array}{rcll} \mathbb{R}^+ & \xrightarrow{\hat{q}^v} & \widehat{Q}^v & \text{(asynchronous)} \\ \lambda(-, v) \downarrow & & \downarrow \pi & \\ \mathbb{N} & \xrightarrow{q^v} & Q^v & \text{(synchronous)} \end{array}$$

That is, $\pi(\hat{q}^v_t) = q^v_{\lambda(t,v)}$

with q^v_n = state in $\mathcal{A}$ of node v at time $n \in \mathbb{N}$
and $\hat{q}^v_t$ = state in $\widehat{\mathcal{A}}$ of node v at time $t \in \mathbb{R}^+$.

[1]More general definitions of emulation allowing differing sets of nodes and alphabets for $\mathcal{A}$ and $\widehat{\mathcal{A}}$, partial definition of π, etc., are possible (in analogy to the classical notion of emulation for synchronous automata networks), but for simplicity we shall use this one which suffices for purposes of this paper.

Theorem 1. (Asynchronous Emulation of Synchronous Automata Networks) *Let any synchronous automata network $\mathcal{A}$ over a locally finite digraph $\Gamma = (V, E)$ with local automata $\mathcal{A}^v = (Q^v, X^v, \delta^v)$ $(v \in V)$ and external input alphabet X be given.*

We construct an asynchronous automata network $\widehat{\mathcal{A}}$ (with the same input alphabet X) such that every possible behavior of $\widehat{\mathcal{A}}$ with input sequence $\{x_n\}_{n>0}$ emulates the (only possible) behavior of $\mathcal{A}$ with input sequence $\{x_n\}_{n>0}$, when $\widehat{\mathcal{A}}$ starts in an initial global state $\hat{q}_0$ depending only on the initial global state q_0 of $\mathcal{A}$.

Moreover, the following hold:

1. *The underlying digraph for $\widehat{\mathcal{A}}$ is the reflexive-symmetric closure of the digraph for $\mathcal{A}$.*
2. *For each vertex v, the local automaton $\widehat{\mathcal{A}}^v$ of $\widehat{\mathcal{A}}$ are "not much more complicated" than the local automaton $\mathcal{A}^v$ of $\mathcal{A}$. Moreover, $\widehat{\mathcal{A}}^v$ is a direct product of $\mathcal{A}^v$, an identity-reset automaton, and a modulo three counter. In fact, $\mathcal{A}^v$ has state set state set $\widehat{Q}^v = Q^v \times Q^v \times \{0, 1, 2\}$.*
3. *The projection $\pi : \widehat{Q} \to Q$ is given locally by $\pi^v(q^v, b^v, r) = q^v$ for $(q^v, b^v, r) \in \widehat{Q}^v$.*
4. *The starting state of $\widehat{\mathcal{A}}$ is given by $\hat{q}_0^v = (q_0^v, q_0^v, 0)$ for all $v \in V$.*
5. *Furthermore, the spatial-temporal covering of the emulation satisfies*

$$|\lambda(t, v) - \lambda(t, v')| \leq \lfloor \frac{d(v, v') + 2}{3} \rfloor.$$

We call $\lambda(t, v)$ the *local time* of the synchronous automaton $\mathcal{A}$ at vertex v for time t in the emulating asynchronous automaton $\widehat{\mathcal{A}}$. Of course, λ depends in general on the update pattern (τ, U) for the particular behavior of $\widehat{\mathcal{A}}$. Thus (5.) above says that the difference in local time at two nodes in the emulating asynchronous automata network is bounded above by approximately one third of the distance between them.

Proof: We give the construction of $\widehat{\mathcal{A}}$ and show by a series of lemmata that it has the required properites:
As before, let $\widehat{\Gamma}$ be the reflexive and symmetric closure of Γ. $\widehat{\Gamma} = (V, \widehat{E})$, where

$$\widehat{E} = \{(v, v') \times V \times V : v = v' \text{ or } (v, v') \in E \text{ or } (v', v) \in E\}.$$

Let $\hat{N}(v)$ denote the neighborhood of v in $\widehat{\Gamma}$. We construct $\widehat{\mathcal{A}}$ as an automata network over $\widehat{\Gamma}$. The local automaton $\widehat{\mathcal{A}}^v$ at node $v \in V$ has states $\widehat{Q}^v = Q^v \times Q^v \times \{0, 1, 2\}$, and its input alphabet is

$$\widehat{X}^v = (X^v \cup \{1\}) \times (\text{Constants on } Q^v \cup \{1\}) \times \{+0, +1\},$$

where 1 is a new symbol that acts as the identity on the corresponding component, and, for each $q^v \in Q^v$, the middle component includes input letter *constant* q^v

which acts as a constant resetting the middle component to q^v, whereas, in the third component, $+0$ acts as the identity and $+1$ increases that component by 1 modulo 3.

We write $\hat{q}^v = (q^v, b^v, r^v) \in \widehat{Q}^v = Q^v \times Q^v \times \{0,1,2\}$ for the local state $\hat{q}^v$ at node $v \in V$ of $\widehat{\mathcal{A}}$.
The read functions of $\widehat{\mathcal{A}}$ are $\hat{\rho}^v : \widehat{Q} \to \{Read, Wait\}$ with

$$\hat{\rho}^v(\hat{q}) = \begin{cases} Read & \text{if } r^w \neq r^v - 1 \mod 3 \text{ for all } w \in \widehat{N}(v) \text{ and } r^v = 0 \\ Wait & \text{otherwise.} \end{cases}$$

The feedback functions of $\widehat{\mathcal{A}}$ are $\widehat{\varphi}^v : Q \times X^{\mathfrak{d}} \to \widehat{X}^v$ with

$$\widehat{\varphi}^v(\hat{q}, x) = \begin{cases} (1,1,+0) & \text{if } r^w = r^v - 1 \mod 3 \text{ for some } w \in \widehat{N}(v) \\ (1,1,+1) & \text{if } r^w \neq r^v - 1 \mod 3 \text{ for all } w \in \widehat{N}(v) \\ & \text{and } r^v \neq 0 \\ (\varphi^v(c,x), constant\, q^v, +1) & \text{if } r^w \neq r^v - 1 \mod 3 \text{ for all } w \in \widehat{N}(v) \\ & \text{and } r^v = 0, \end{cases}$$

where q^v is the first component of $\hat{q}^v$ in state $\hat{q}$ and c is an arbitrary state of $\mathcal{A}$ such that for each $w \in N(v)$,

$$c^w = \begin{cases} q^w \text{ if } r^w = 0 \\ b^w \text{ if } r^w = 1. \end{cases}$$

Note also r^w must lie in $\{0,1\}$ in the determining the c^w of the third case, as necessarily $r^v = 0$ in third case and $w \in N(v) \subseteq \widehat{N}(v)$ implies $r^w \neq 2 \mod 3$.

Updates at node v in $\widehat{\mathcal{A}}$ are thus given by the local update function with

$$\widehat{\delta}^v((q^v, b^v, r^v), \widehat{\varphi}^v(\hat{q}, x)) = (q^v, b^v, r^v) \cdot \widehat{\varphi}^v(\hat{q}, x) =$$
$$\begin{cases} (q^v \cdot 1, b^v \cdot 1, r^v + 0) & \text{if } r^w = r^v - 1 \bmod 3 \\ & \text{for some } w \in \widehat{N}(v) \\ (q^v \cdot 1, b^v \cdot 1, r^v + 1 \bmod 3) & \text{if } r^w \neq r^v - 1 \bmod 3 \\ & \text{for all } w \in \widehat{N}(v), \text{ and } r^v \neq 0 \\ (q^v \cdot \varphi^v(c,x), b^v \cdot constant\, q^v, 0 + 1) & \text{otherwise,} \end{cases}$$

where c is as above.

That is,

$$\widehat{\delta}^v((q^v, b^v, r^v), \widehat{\varphi}^v(\hat{q}, x)) =$$
$$\begin{cases} (q^v, b^v, r^v) & \text{if } r^w = r^v - 1 \bmod 3 \text{ for some } w \in \widehat{N}(v) \\ (q^v, b^v, r^v + 1 \bmod 3) & \text{if } r^w \neq r^v - 1 \bmod 3 \text{ for all } w \in \widehat{N}(v) \\ & \text{and } r^v \neq 0 \\ (q^v \cdot \varphi^v(c,x), q^v, 1) & \text{otherwise,} \end{cases}$$

where c is as above.

Notice that the transition function of $\mathcal{A}^v$ and the feedback function φ^v from the synchronous network are used to give the input to $\mathcal{A}^v$ in the third case. Of course the value of $q^v \cdot \varphi^v(c, x)$ depends only on x, q^v and the c^w with w in $N(v)$, the neighborhood of v in the original digraph Γ.

In computing $\widehat{\delta}^v$, $x \in X^{\eth}$ is the letter currently available for possible reading by the local automaton at node v if $\hat{\rho}^v(\hat{q}) = Read$ but is $x = \eth$ in case $\hat{\rho}^v(\hat{q}) = Wait$ (see discussion of local reading and waiting above). By our choice of reading functions, the letter x is the $Wait$ symbol $\eth$ in the first two cases of the local update rule and lies in X if and only if the third case applies.[2] Thus, the third case applies if and only if the next available letter is consumed.

Suppose the initial state of $\mathcal{A}$ in a synchronous run is q_0 with each node $v \in V$ in state $q_0^v \in Q^v$. Let the initial state of $\widehat{\mathcal{A}}$ have the automaton at each node v in state $\hat{q}_0^v = (q_0^v, q_0^v, 0)$.

For a given behavior of $\widehat{\mathcal{A}}$, we say there is a $+1$-*update* at vertex v whenever the transition rule $\widehat{\delta}^v$ is applied to update the local state using either its second or third cases, i.e. exactly when the last component of state is incremented by $+1$ modulo 3. We say there is a *real update* at node v (corresponding to a synchronous update in $\mathcal{A}$ at that node) whenever transition rule $\widehat{\delta}^v$ is applied to update the local state using its third case, i.e. exactly when the the last component of state changes from 0 to 1. Let $p(t, v)$ be the number of $+1$-updates that have occurred at vertex v during a behavior with update pattern τ up to and including time $t \in \mathbb{R}^+$.

Lemma 2. *For each pair of neighboring nodes $v, v' \in V$ in $\widehat{\Gamma}$ and for all $t \in \mathbb{R}^+$,*

$$|p(t, v) - p(t, v')| \leq 1.$$

Proof: It suffices to consider the values of $p(t, v)$ for $t \in \{\tau(0), \tau(1), \tau(2), \tau(3), \ldots\}$ since no applications of local transition rules occur between them. At $t = 0 = \tau(0)$, $p(t, w) = 0$ for all $w \in V$, so $p(t, v) = p(t, v')$ holds. Now by induction, we suppose the inequality holds at time $\tau(k)$ for all $k \leq n \in \mathbb{N}$.

If $p(\tau(n), v) = p(\tau(n), v')$ then at time $\tau(n + 1)$, a $+1$-update will occur at none, one, or both of v and v', so, as a result, the inequality will always hold at time $\tau(n + 1)$.

Otherwise $|p(\tau(n), v) - p(\tau(n), v')| = 1$. Without loss of generality, we may assume

$$p(\tau(n), v') = p(\tau(n), v) - 1.$$

[2]We remark that if X is a singleton, the above definition of $\widehat{\delta}^v$ is also consistent with $\hat{\rho}^v(\hat{q}) =$ *Read* for all $\hat{q} \in \widehat{Q}$, since the current letter is not different from the next one in an infinite input word consisting of identical letters. This observation will be used when we specialize Theorem 1 in Corollaries 8 and 9 respectively to emulating generalized cellular automata and cellular automata by asynchronous automata networks which can be chosen to be asynchronous generalized cellular automata and asynchronous cellular automata, respectively.

Since the last component of state at a node is increased by one (modulo three) for each +1-update at that node and otherwise remains unchanged, obviously the last component r_t^w of state at any node w in V is $p(t,w) \mod 3$. Therefore

$$r^{v'}_{\tau(n)} = r^v_{\tau(n)} - 1 \mod 3.$$

It follows from the definition of local update $\widehat{\delta}^v$ that there will be no +1-update at node v at $\tau(n+1)$, so $p(\tau(n),v) = p(\tau(n+1),v)$. Now there are two possibilities: Either there is also no +1-update at node v' at this time, so $p(\tau(n),v') = p(\tau(n+1),v')$ and the inequality is preserved. Or, otherwise, there is a +1-update at node v', so then

$$p(\tau(n+1),v') = p(\tau(n),v') + 1 = p(\tau(n),v) = p(\tau(n+1),v),$$

and again the inequality holds.

It follows by induction that it holds for all n, hence for all $t \in \mathbb{R}^+$. □

Corollary 3. *Let v and v' be vertices at distance d in the graph $\widehat{\Gamma}$. Then for all $t \in \mathbb{R}^+$,*

$$|p(t,v) - p(t,v')| \leq d.$$

Proof: This follows immediately by considering a path of minimal length from v to v' and applying the above lemma. □

Define $\lambda(t,v)$ to be the number of real updates that have occured at node v in $\widehat{\mathcal{A}}$ up to and including time $t \in \mathbb{R}^+$.

Corollary 4. *Let v and v' be vertices at distance d in the graph $\widehat{\Gamma}$. Then for all $t \in \mathbb{R}^+$,*

$$|\lambda(t,v) - \lambda(t,v')| \leq \lfloor \frac{d+2}{3} \rfloor.$$

Proof: By definition we have

$$\lambda(t,v) = \lceil \frac{p(t,v)}{3} \rceil.$$

We may assume $\lambda(t,v) \geq \lambda(t,v')$, whence

$$\begin{aligned} |\lambda(t,v) - \lambda(t,v')| &= \lambda(t,v) - \lambda(t,v') \\ &= \lceil \frac{p(t,v)}{3} \rceil - \lceil \frac{p(t,v')}{3} \rceil \\ &\leq \frac{p(t,v)+2}{3} - \frac{p(t,v')}{3} \\ &\leq \frac{d+2}{3} \text{ by the corollary above.} \end{aligned}$$

Since $|\lambda(t,v) - \lambda(t,v')|$ is an integer, it can be no more than $\lfloor \frac{d+2}{3} \rfloor$. □

Lemma 5 (Freedom from Deadlocks)**.** *With asynchronous automata network $\widehat{\mathcal{A}}$ in the initial configuration with $\hat{q}_0^v = (q_0^v, q_0^v, 0)$ corresponding to an initial configuration of $\mathcal{A}$ with node $v \in V$ in q_0^v, for any update pattern (τ, U) and any input sequence $\{x_n\}_{n>0}$ of letters in X, the number of real updates at each node is unbounded. That is, for each fixed $v \in V$, always*

$$\lim_{n\to\infty} \lambda(\tau(n), v) = \infty.$$

It follows that $\lambda : \mathbb{R}^+ \times \{v\} \to \mathbb{N}$ is surjective for each $v \in V$.

Proof: It suffices to show, for each fixed $v = v_0 \in V$, $p(\tau(n), v_0)$ increases without bound. Hence it is enough to show, that if $p(\tau(n), v_0) = R$ then there is an $r > n$ with $p(\tau(r), v_0) > R$.

Since $\widehat{\Gamma}$ is locally finite, there are finitely many nodes $v_0, v_1, \ldots v_k$ with distance $d(v_0, v_i) \leq R$. By our hypotheses on update patterns each of these $v_i \in U_{\tau(m)}$ for infinitely many $m > n$.

Suppose, for a contradiction, that no node amongst these can receive a $+1$-update. That is, for all $m > n$, $p(\tau(n), v_i) = p(\tau(m), v_i)$ for all $0 \leq i \leq k$. Let $w(0) = v_0$. Inductively, starting with $i = 0$, since $w(i)$ cannot get a $+1$-update, by definition of $\hat{\delta}^v$, $w(i)$ must have a neighbor $w(i+1)$ with $r_{w(i+1)} = r_{w(i)} - 1 \mod 3$, hence by Lemma 2, $p(\tau(n), w(i+1)) = p(\tau(n), w(i)) - 1$. Thus we can find distinct nodes, $w(0), w(1), w(2), \ldots, w(R)$ within distance R of node $v = w(0)$ such that $p(\tau(n), w(i)) = p(\tau(n), v) - i$ for $0 \leq i \leq R$. In particular the node $w(R)$ has $p(\tau(n), w(R)) = 0$. By Lemma 2, the neighboring nodes w' to $w(R)$ have $p(\tau(n), w') \in \{0, 1\}$, and $0 \leq p(\tau(m'), w') \leq 1$ for all $m' > n$ as long as $w(R)$ has not received a $+1$-update. Let $m > n$ be the least integer, such that $w(R) \in U_{\tau(m)}$. Then by the local update rule, $w(R)$ gets a $+1$-update at time $\tau(m)$ but lies with distance R of v_0, a contradiction.

Therefore, within the finitely many nodes within distance R of v, some node must indeed be $+1$-updated at some time $\tau(m)$ with $m > n$. Repeating this argument, for any time $\tau(n')$, we can find always a time $\tau(m')$ with $m' > n'$ such that some node within distance R of v_0 gets a $+1$-update. Since there are only finitely many such nodes, eventually (for some $r > n$) some node $w = v_i$ $(0 \leq i \leq k)$ among them will have $p(\tau(r), w) > 2R$. But then by Corollary 3,

$$|p(\tau(r), w) - p(\tau(r), v_0)| \leq R,$$

implying that

$$p(\tau(r), v_0) > R,$$

i.e. node v_0 must get a $+1$-update as well. □

The local time function λ is clearly locally monotonically increasing, so Lemma 5 and Corollory 4 (together with the fact that $\lfloor \frac{d+2}{3} \rfloor \leq d$) show that λ is a spatial-temporal covering as required.

Proposition 6 (Emulation using Local Time)**.** *Let the initial states, inputs and update pattern be as in Lemma 5. Then the first component of state at node v in the asynchronous automata network $\widehat{\mathcal{A}}$ at time t equals the state of node v in the synchronous automata network $\mathcal{A}$ at time $\lambda(t, v)$. That is,*

$$q^v_{\lambda(t,v)} = \pi(\hat{q}^v_t).$$

Proof: It suffices to prove the assertion for all $t \in \tau(\mathbb{N})$.

If $\hat{q}^v_t = (x, y, r) \in \widehat{\mathcal{A}}^v$, then let $\pi(\hat{q}^v_t) = x$, its first component as before; let $\pi_2(\hat{q}^v_t) = y$, its second component, and let $r^v_t = r$, its third component.

We proceed by induction on m to show that

1. $q^v_{\lambda(\tau(m),v)} = \pi(\hat{q}^v_{\tau(m)})$,
2. $\lambda(\tau(m), v) \geq 1$ implies that $\pi_2(\hat{q}^v_{\tau(m)}) = q^v_{\lambda(\tau(m),v)-1}$,
3. $m > 0$ implies that $m^*(v) = \lambda(\tau(m-1), v) + 1$.

We first carry out the induction for (3.). If $m = 1$, then by definition $1^*(v) = 1$, but since at time $\tau(0)$ at v there have been no real updates, we have $\lambda(\tau(0), v) + 1 = 1$. For $m > 1$, by definition $m^*(v) = (m-1)^*(v) + 1$ if and only if $v \in U_{\tau(m)}$ and $\rho^v(\hat{q}_{\tau(m-1)}) = \mathit{Read}$, i.e. if and only if the third case in the definition of $\hat{\delta}^v$ is applied. Thus $m^*(v)$ increases if and only if there is a real update at node v, i.e. every time $\lambda(\tau(m-1), v)$ increases by 1 so does $m^*(v)$. Thus,

$$\lambda(\tau(m), v) - \lambda(\tau(m-1), v) = (m+1)^*(v) - m^*(v).$$

This and the induction hypotheis yields (3).

For $m = 0$, we have $\tau(0) = 0$, $\lambda(\tau(0), v) = 0$ and $\hat{q}^v_0 = (q^v_0, q^v_0, 0)$, so (l.) is immediate; (2.) holds vacuously.

Suppose by induction hypothesis that (1.) and (2.) hold for all m with $0 \leq m < n$. We show these assertions follow also for $m = n$:

(case 1): If $v \in U(\tau(n))$ but there is no real update at node v, it follows from the definition of λ that $\lambda(\tau(n), v) = \lambda(\tau(n-1), v)$, and, using the definition of $\hat{\delta}^v$ that the first and second coordinates of $\hat{q}^v$ are unchanged. Thus, $\pi(\hat{q}^v_{\tau(n)}) = \pi(\hat{q}^v_{\tau(n-1)})$ and therefore,

$$q^v_{\lambda(\tau(n),v)} = q^v_{\lambda(\tau(n-1),v)} = \pi(\hat{q}^v_{\tau(n-1)}) = \pi(\hat{q}^v_{\tau(n)}),$$

as required for (1.). For the implication (2.), if $\lambda(\tau(n), v) > 1$ then

$$\begin{aligned} \pi_2(\hat{q}^v_{\tau(n)}) &= \pi_2(\hat{q}^v_{\tau(n-1)}) \text{ as 2nd component is unchanged} \\ &= q^v_{\lambda(\tau(n-1),v)-1} \text{ by induction hypothesis} \\ &= q^v_{\lambda(\tau(n),v)-1} \text{ since } \lambda(\tau(n), v) = \lambda(\tau(n-1), v), \end{aligned}$$

as required. While if $\lambda(\tau(n), v) = 1$ then it is clear from the definition of $\hat{\delta}^v$, since the value of the second coordinate can only be changed in case 3 below, that $\pi_2(\hat{q}^v_{\tau(n)}) = q^v_0$, which is just $q^v_{\lambda(\tau(n),v)-1}$, as required.

(case 2): If $v \notin U(\tau(n))$, then $\hat{q}^v$ is unchanged and everything follows as above.

(case 3): Finally, if $v \in U(\tau(n))$ and there is a real update at v at time $\tau(n)$, then by definition of $\hat{\delta}^v$ have $r^v_{\tau(n-1)} = 0$ and $r^w_{\tau(n-1)} \neq r^v_{\tau(n-1)} - 1 \mod 3$ for all $w \in \widehat{N}(v)$, where r^w_t denotes the third component of $\hat{q}^w_t$. So each $r^w_{\tau(n-1)} \in \{0, 1\}$. It is also clear by induction that $r^w_t = p(t, w) \mod 3$ always holds. By Lemma 2, $|p(t, w) - p(t, v)| \leq 1$ for the neighboring nodes w and v.

We consider the c^w's that are used in the third case of the local update rule in updating node v:

If $r^w_{\tau(n-1)} = 0$ then $p(\tau(n-1), w) = p(\tau(n-1), v)$ follows and hence $\lambda(\tau(n-1), w) = \lambda(\tau(n-1), v)$. In this case, by induction hypothesis for node w, $\pi(\hat{q}^w_{\tau(n-1)}) = q^w_{\lambda(\tau(n-1),w)} = q^w_{\lambda(\tau(n-1),v)}$. It follows that c^w in third case of the local update rule equals $q^w_{\lambda(\tau(n-1),v)}$.

Otherwise, $r^w_{\tau(n-1)} = 1$, and since $r^v_{\tau(n-1)} = 0$, we have $p(\tau(n-1), w) = p(\tau(n-1), v) + 1$ and hence $\lambda(\tau(n-1), w) = \lambda(\tau(n-1), v) + 1$.[3]

Thus $\lambda(\tau(n-1), w) \geq 1$, and so then

$$\begin{aligned} c^w &= \pi_2(\hat{q}^w_{\tau(n-1)}) \text{ since } r^w_{\tau(n-1)} = 1 \\ &= q^w_{\lambda(\tau(n-1),w)-1} \text{ by induction hypothesis at node } w \\ &= q^w_{\lambda(\tau(n-1),v)}. \end{aligned}$$

It follows that, for each $w \in \widehat{N}(v)$, the c^w in the third case of the local update rule applied at node v equals $q^w_{\lambda(\tau(n-1),v)}$.

By the induction hypothesis that $\pi(\hat{q}^v_{\tau(n-1)}) = q^v_{\lambda(\tau(n-1),v)}$ and by (3.), $n^*(v) = \lambda(\tau(n-1), v) + 1$. Thus, the first component of $\hat{q}^v_{\tau(n)}$ is

$$\begin{aligned} \pi(\hat{q}^v_{\tau(n)}) &= \pi(\hat{q}^v_{\tau(n-1)}) \cdot \varphi^v(c, x_{n^*(v)}) \text{ by definition of } \widehat{\delta^v} \\ &= q^v_{\lambda(\tau(n-1),v)} \cdot \varphi^v(c, x_{n^*(v)}) \text{ by induction hypothesis} \\ &= q^v_{\lambda(\tau(n-1),v)} \cdot \varphi^v(q_{\lambda(\tau(n-1),v)}, x_{n^*(v)}) \\ &\quad \text{since for all neighbors } w \in N(v), \text{ we have} \\ &\quad\quad c^w = q^w_{\lambda(\tau(n-1),v)} \\ &= q^v_{\lambda(\tau(n-1),v)} \cdot \varphi^v(q_{\lambda(\tau(n-1),v)}, x_{\lambda(\tau(n-1),v)+1}) \text{ by (3.)} \\ &= q^v_{\lambda(\tau(n-1),v)+1} \text{ by definition of local update in } \mathcal{A} \\ &= q^v_{\lambda(\tau(n),v)} \text{ since at time } \tau(n) \text{ there is a real update at } v. \end{aligned}$$

[3]Notice that the most recent +1-update at node w must have been a real update, since $r^w_{\tau(n-1)} = 1$. Since $r^v_{\tau(n-1)} = 0$ and node v and w differ by at most one +1-update, local time at node v is behind local time at node w by exactly 1.

This shows (1.)

Also by definition of $\widehat{\delta}^v$, we have that the second component of $\hat{q}^v_{\tau(n)}$ equals the first component of $\hat{q}^v_{\tau(n-1)}$, so

$$\begin{aligned}\pi_2(\hat{q}^v_{\tau(n)}) &= \pi(\hat{q}^v_{\tau(n-1)}) \\ &= q^v_{\lambda(\tau(n-1),v)} \text{ by induction hypothesis} \\ &= q^v_{\lambda(\tau(n),v)-1} \text{ again since at time } \tau(n) \text{ there is a real} \\ &\quad \text{update at } v.\end{aligned}$$

This shows (2.) and completes the induction. The lemma is proved. □

Remark. By Lemma 5, for each node $v_0 \in V$, $\lambda(\tau(n), v_0)$ takes all non-negative integer values, so the Proposition 6 shows how to recover the entire history of node v in $\mathcal{A}$: Just record the first component of the state at node v in $\widehat{\mathcal{A}}$ every time there is a real update at v. Thus the sequence of global states $q_0, q_1, \ldots$ of $\mathcal{A}$ under a given external input sequence $x_1, x_2, \ldots$ can be recovered from any behavior of $\widehat{\mathcal{A}}$ started in state $\hat{q}^v_0 = (q^v_0, q^v_0, 0)$ for all $v \in V$ with the same input sequence.

The results above establish all the properties asserted for $\widehat{\mathcal{A}}$ in the statement of Theorem 1, whose proof is now complete. □

Definition. For a given external input sequence $\{x_n\}_{n>0}$ and update pattern (τ, U), any function $\eta : \mathbb{N} \times V \to \mathbb{R}^+$ satisfying, for all $v \in V$, $n \in \mathbb{N}$,

$$q^v_n = \pi(\hat{q}^v_{\eta(n,v)})$$

is called a *spatial-temporal section* of the behavior of the asynchronous automata network $\widehat{\mathcal{A}}$ mapping onto the behavior of the synchronous automata network $\mathcal{A}$.

Corollary 7 (Existence of Spatial-Temporal Sections)**.** *Let $\eta_\tau : \mathbb{N} \times V \to \mathbb{R}^+$ be defined by*

$$\eta_\tau(n, v) = \textit{the least } \tau(m) \textit{ with } m \in \mathbb{N} \textit{ such that } \lambda(\tau(m), v) \textit{ is } n.$$

Then η_τ is a spatial-temporal section of the behavior of the asynchronous automata network $\widehat{\mathcal{A}}$ mapping onto the behavior of the synchronous automata network $\mathcal{A}$.

Proof: As noted in the above remark, for every node v, local time $\lambda(\tau(m), v)$ takes all values $n \in \mathbb{N}$, so η_τ is well-defined. By Proposition 6, this function is a spatial-temporal section. □

The function η_τ of Corollary 7 is called the *natural spatial-temporal section* for fixed external input sequence $\{x_n\}_{n>0}$ and update pattern (τ, U) of the behavior of $\widehat{\mathcal{A}}$.

3. Generalized Cellular Automata and Cellular Automata

A locally finite synchronous or asynchronous automata network is said to be a *generalized cellular automaton* if its external input alphabet X is a singleton. In the synchronous case, the external input serves, in effect, only as a global synchronous update signal for $\mathcal{A}$ – a 'clock tick', but does not otherwise effect the state of $\mathcal{A}$.[4] Similarly, in the asynchronous case, the external input letter serves as a local update signal for $\mathcal{A}^v$ whenever $v \in U(\tau(n))$ and $\rho^v(\tau(n-1)) = \mathit{Read}$.

A generalized cellular automaton $\mathcal{A}$ is a *cellular automaton* (CA) if it satisfies:
(1) The edge relation E is a symmetric relation on V,
(2) For every $v, w \in V$, $\mathcal{A}^w = \mathcal{A}^v$. That is, the same local automaton occurs at each node.
(3) For every $v, w \in V$, there is a corresponding graph automorphism $\pi : \Gamma \to \Gamma$ with $\pi(v) = w$ such that for all $q \in Q$,

$$\varphi^v((q_u)_{u \in N(v)}, x) = \varphi^w((q_{\pi(u)})_{\pi(u) \in N(w)}, x).$$

Conditions (2) and (3) imply that the automata network is highly homogeneous in that every vertex in the network has an isomorphic local neigbhorhood and, also, the component automaton at each vertex computes the same transition function of the states of component automata in its local neighborhood as computed by any other component automaton in the network.

For cellular automata (and sometimes for automata networks in general), the local automata $\mathcal{A}^v$ are sometimes referred to as the "*cells*" of $\mathcal{A}$.

Corollary 8 (Asynchronous Emulation of Generalized CAs)**.** *If $\mathcal{A}$ is a synchronous generalized cellular automaton then there is an emulating asynchronous automata network satisfying the same conclusions as in Theorem 1, but also $\widehat{\mathcal{A}}$ is an asynchronous generalized cellular automaton.*

Proof: This is of course just a special case of Theorem 1, with the additional observation that $\widehat{\mathcal{A}}$ is an asynchronous generalized cellular automaton since it has the same singleton alphabet as $\mathcal{A}$.

But it worth remarking that in this case a further simplification is possible. Since the external input alphabet $X = \{x\}$, all input letters in the infinite external input sequence are identical. So it is possible to completely ignore the external input letter in the feedback functions φ^v and moreover it is unnecessary to keep track locally of which letter in the infinite word is available for reading. Therefore the definition of $\hat{\rho}^v(\hat{q})$ can be simplified to be constant $\hat{\rho}^v(\hat{q}) = \mathit{Read}$ for all $v \in V$ and $\hat{q} \in \widehat{Q}$ without affecting the global run. The resulting asynchronous generalized cellular automata never waits before reading a letter, node

[4]The reader familiar with cellular automata may find it helpful to think of a $\mathcal{A}$ as cellular automaton, except that the interconnection graph Γ is not required to be regular, the local automata $\mathcal{A}^v$ are not required to be isomorphic, and neighborhoods are not required to be symmetric ($v \in N(v')$ does not necessarily imply $v' \in N(v)$ for vertices v, v'.)

v is updated exactly when $v \in U_t$, and no $Wait$ symbol cases are needed for the the feedback functions, i.e. we restrict φ^v to $\widehat{Q} \times X \to X^v$ using X rather than $X^{\eth}$. Since X is a singleton we may as well simplify feedback functions to have form $\varphi^v : \widehat{Q} \to X^v$. Thus, the local transition functions then simplify to

$$\widehat{\delta}^v((q^v, b^v, r^v), \widehat{\varphi}^v(\hat{q})) = \begin{cases} (q^v, b^v, r^v) & \text{if } r^w = r^v - 1 \bmod 3 \text{ for some } w \in \widehat{N}(v) \\ (q^v, b^v, r^v + 1 \bmod 3) & \text{if } r^w \neq r^v - 1 \bmod 3 \text{ for all } w \in \widehat{N}(v) \\ & \quad \text{and } r^v \neq 0 \\ (q^v \cdot \varphi^v(c), q^v, 1) & \text{otherwise,} \end{cases}$$

where c is an arbitrary state of $\mathcal{A}$ such that for each $w \in N(v)$,

$$c^w = \begin{cases} q^w \text{ if } r^w = 0 \\ b^w \text{ if } r^w = 1. \end{cases}$$

□

Corollary 9 (Asynchronous Emulation of Cellular Automata)**.** *If $\mathcal{A}$ is a synchronous cellular automaton then there is an emulating asynchronous generalized cellular automaton satisfying the same conclusions as in Corollary 8, but also $\widehat{\mathcal{A}}$ is an asynchronous cellular automaton.*

Proof: This is clear since the directed graph $\widehat{\Gamma}$ of $\widehat{\mathcal{A}}$ is identical to the directed graph Γ of $\mathcal{A}$ except possibly that all nodes become neighbors of themselves in $\widehat{\Gamma}$, and so $\widehat{\mathcal{A}}$ inherits the conditions in the definition of cellular automata satisfied by $\mathcal{A}$. □

4. Remarks and Open Problems

Remarks on Implementations and State Number. (1) The need for a global clock that is required by synchronous automata networks and (generalized) cellular automata is eliminated by the Asynchronous Emulation Theorem. It constructively shows how an asynchronous emulation can be implemented, e.g. on parallel, distributed and/or asynchronous computational devices, without global clocks, and how the synchronous behavior can be recovered.
(2) In computational implementations of synchronous cellular automata on present-day sequential computers it is usual to keep two copies of the state space, one for current state of the entire space and one for the next state into which updated local states are written as they are computed. Before the next global time step the two global copies are exchanged, and then the process repeated. Thus in practice for each cell $\mathcal{A}^v$ in the space, one keeps two copies of local states. So if there are $|Q^v| = n$ possible states in each cell, this corresponds to n^2 possible states for each cell in an implementation.

For asynchronous cellular automata, our construction of $\widehat{\mathcal{A}}$ for Corollary 9 (and for asynchronous automata networks more generally in Theorem 1) uses local automata that for each of the corresponding synchronous local automaton keep a copy of current local state (their first component), which is 'current' according to local time $\lambda(t, v)$, and a copy of the previous local state (in their second component), and in addition a modulo 3 counter value. There are thus $3n^2 = |\widehat{Q}^v| = |Q^v| \times |Q^v| \times 3$ possible local states. But if $v, v' \in U_t$ implies $d(v, v') \neq 1$ (e.g. if only a single random node is updated at a time), then it unnecessary to keep auxillary copies of the entire state space (or even of the portion to be updated) in a sequential implementation. The only essential increase in memory usage is then the addition of local modulo 3 counters at each node.

Remark on Local Synchronization with Modulo n Counters. It is straightforward to modify the proof of Theorem 1 to obtain a variant result using modulo n counters, for $n \geq 3$, rather than modulo 3 counters for local synchronization. This of course results in corresponding variants of Corollaries 8 and 9 for asynchronous emulation in the realms of generalized cellular and cellular automata.

Open Problems. The Asynchronous Emulation Theorem (Theorem 1) allows the update sets $U_{\tau(n)}$ for $n > 0$ to be arbitrary subject only to having $v \in U_{\tau(n)}$ for infinitely many n. Thus, for example, determinisitic or nondeterministic, sequential, uniformly random or Poisson-distributed, locally synchronous and other choices of update patterns are permitted.
(1) For particular types of update patterns and network topologies, derive precise bounds on the rate of local real update: Given $v \in V$ study the relative passage of local time in the asynchronous model at node v as compared to the synchronous one, i.e. for synchronous global time $t \in \mathbb{N}$ determine bounds on the ratio

$$\lambda(\tau(t), v)/t$$

for $t > 0$, and study its behavior as $t \to \infty$.
Under what circumstances does the ratio remain bounded away from zero?
(2) Also determine the precise (or expected) number $E(t_0)$ of asynchronous updates for local time to exceed t_0, i.e. determine $E(t_0) \in \mathbb{R}^+$ with

$$n \geq E(t_0) \Rightarrow \lambda(v, \tau(n)) \geq t_0.$$

Under what circumstances is $E(t_0)$ independent of v?
(3) Extend the Asynchronous Emulation Theorem and its Corollaries to networks in which the underlying graph is permitted to change over time, i.e. with addition or deletion of new edges and nodes.
(4) Extend the results to the case when state changes are not instantaneous, and a node may receive delayed information concerning the states of its neighbors.
(5) Develop methods for synchronous and asynchronous automata networks to cope with defective local automata and errors in transmission of local state to neighbors.

(6) Is it possible to obtain analogous results to those of this paper if sometimes letters of external input have not yet arrived at nodes reading them? This would represented a strengthening of the Asynchronous Emulation Theorem (but not for the generalized cellular automata analogues), since it would then not need to be assumed that the next letter of global input were always available for local reading, thus allowing for delays in the external input reaching local nodes of the network.

Bibliographic Remarks

Cellular automata were introduced by J. von Neumann and S. Ulam; an important early study is by J. von Neumann [1966], see also E. F. Codd [1968], A. W. Burks [1970]. Automata networks are studied in the books of F. Gécseg and I. Péak [1972], F. Gécseg [1986], C. Choffrut [1986], F. Fogelman Soulié et al. [1987], P. Dömösi and C. L. Nehaniv [in press]. Research articles on general and restricted classes of network digraphs include for example V. M. Gluškov [1961], A. A. Letichevsky [1961], K. Krohn and J. L. Rhodes [1962, 1965, 1968 (with B. Tilson)], Z. Ésik and collaborators [1983, 1985, 1986a, 1986], M. Tchuente [1979a, 1979b, 1983, 1985, 1986, 1988], and P. Dömösi and C. L. Nehaniv [1998, 1999, 2000].

Studies of asynchronous automata networks with applications to computer and electrical engineering include, e.g. V. Varshavsky [1965 (with B. L. Osievich), 1968, 1969, 1990] and his collaborators M. Kishinevsky et al. [1994a, 1994b], and also J. A. Brzozowski and C.-J. Seger [1995], J. A. Brzozowski [2000], among many others.

A weaker version of Corollary 9 for asynchronous cellular automata, using a local transition function equivalent to the one in Corollary 8, was found independently by K. Nakamura [1974], who sketches a proof of freedom from deadlocks (but not emulation) in this case, and also again independently in a variant form using modulo four counters but without complete proofs by T. Toffoli [1978] and T. Toffoli and N. Margolus [1987]. The proofs in this paper thus fill gaps in the literature, and apply to more general settings. The Asynchronous Emulation Theorem, its corrolaries (Corollaries 8 and 9, which extend these previously known constructions but incompletely proved results), and all other results shown here are due independently to the author and were announced in C. L. Nehaniv [2002a]. Applications exhibiting the details of the construction in the simplified case of emulating synchronous cellular automata by asynchronous cellular automata, and the first examples of self-replication and of evolution in implemented asynchronous cellular automata and as well as remarks on universal computation in asynchronous cellular automata are given by C. L. Nehaniv [2002b, 2002c].

References

[1] J. A. Brzozowski [2000], Delay-insensitivity and tenary timulation. *Theoretical Computer Science*, **245**, **245** (2000), 3–25.

[2] J. A. Brzozowski and C.-J. Seger [1995], *Asynchronous Circuits.* Springer Verlag, New York, 1995.

[3] A. W. Burks [1970] (ed.), *Essays on Cellular Automata.* University of Illionis Press, Urbana, 1970.

[4] C. Choffrut (ed.) [1986], *Automata Networks.* Lecture Notes in Computer Science, vol. 316, Springer,1986.

[5] E. F. Codd [1968], *Cellular Automata.* Academic Press, New York, 1968.

[6] P. Dömösi and C. L. Nehaniv [1998], Algebraic theory of finite automata networks. *Math. Japon.*, **48** (1998), 481–509.

[7] P. Dömösi and C. L. Nehaniv [1999], Complete finite automata network graphs with minimal number of edges. *Acta Cybern.,* **14** (1999), 37–50.

[8] P. Dömösi and C. L. Nehaniv [2000], On complete systems of automata. *Theor. Comput. Sci.*, **245**, No. 1, (2000), 27–54.

[9] P. Dömösi and C. L. Nehaniv [in press], *Algebraic Theory of Automata Networks: An Introduction*, SIAM Monographs on Discrete Mathematics and Applications (in press).

[10] Z. Ésik [1985], Homomorphically complete classes of automata with respect to the α_2-product. *Acta Sci. Math.*, **48** (1985), 135–141.

[11] Z. Ésik [1986a], Complete classes of automata for the α_1-product. *Found. Control Engr.* **11** (1986), 95–107.

[12] Z. Ésik and P. Dömösi [1986], Complete classes of automata for the α_0-product. *Theor. Comput. Sci.*, **47** (1986), 1–14.

[13] Z. Ésik and Gy. Horváth [1983], The α_2-product is homomorphically general. *Papers on Automata Theory V*, No. DM83-3, Dep. Math., Karl Marx University of Economics, Budapest (1983), 49–62.

[14] F. Gécseg [1986], *Products of Automata.* EATCS Monographs on Theor. Comput. Sci., Vol. **7**, Springer-Verlag, Berlin - Heidelberg - New York - Tokyo, 1986.

[15] F. Gécseg and and I. Peák [1972], *Algebraic Theory of Automata.* Disquisitiones Mathematicae Hungaricae **2**, Akadémiai Kiadó, Budapest, 1972.

[16] V. M. Gluškov [1961], Abstract theory of automata (in Russian). *Uspehi matematičevskih nauk* 16:5 (101) (1961), pp. 3–62. Correction: ibid., 17:2 (104) (1962), p. 270.

[17] M. Kishinevsky, A. Kondratyev, A. Taubin, V. Varshavsky [1994a], *Concurrent Hardware: The Theory and Practice of Self-Timed Design.* Wiley Professional Computing, Chichester, (1994).

[18] M. Kishinevsky, A. Kondratyev, A. Taubin, V. Varshavsky [1994b], Analysis and identification of speed-independent circuits on an event model. *Form. Methods Syst. Des.* **4**, No.1, 33-75 (1994).

[19] K. B. Krohn and J. L. Rhodes [1962], Algebraic theory of machines. In: J. Fox, ed., *Proc. Symp. Math. Theory of Automata,* (Brooklyn 1962), 341-384.

[20] K. B. Krohn and J. L. Rhodes [1965], Algebraic theory of machines, I. Prime decomposition theorem for finite semi-groups and machines. *Trans. Amer. Math. Soc.* **116** (1965), 450–464.

[21] K. B. Krohn, J. L. Rhodes and B. R. Tilson [1968], The prime decomposition theorem of the algebraic theory of machines. In: M. Arbib, ed., *Algebraic Theory of Machines, Languages and Semigroups*, Academic Press, New York, 1968.

[22] A. A. Letichevsky [1961], Conditions of completeness for finite automata (in Russian). *Žurn. Mat. i Mat. Fiz.*, 1 (1961), 702–710.

[23] K. Nakamura [1974], Asynchronous cellular automata and their computational ability. *Systems, Computers, Controls*, **5**, No. 5, pp. 58-66, 1974, [translated from Japanese, *Denshi Tsushin Gakkai Ronbunshi*, **57-D**, No. 10, pp. 573–580, October 1974.]

[24] C. L. Nehaniv [2002a], Asynchronous Automata Networks Can Emulate Any Synchronous Automata Network, presented at *International Workshop on Semigroups, Automata, and Formal Languages (June 2002 -Crema, Italy)*, 2002 [published abstract].

[25] C. L. Nehaniv [2002b], Self-Reproduction in Asynchronous Cellular Automata, *Proc. 2002 NASA/DoD Conference on Evolvable Hardware (15-18 July 2002 – Alexandria, Virginia)*, IEEE Computer Society Press, pp. 201–209, 2002.

[26] C. L. Nehaniv [2002c], Evolution in Asynchronous Cellular Automata, *Artificial Life VIII: Proc. 8th Intl. Conf. on Artificial Life* (eds: R. K. Standish, M. A. Bedau, and H. A. Abbass), MIT Press, pp. 65–73, 2002.

[27] J. von Neumann [1966], *Theory of Self Reproducing Automata.* Edited and completed by A. W. Burks, University of Illionis Press, Urbana, 1966.

[28] F. Fogelman Soulié, Y. Robert and M. Thcuente (eds.) [1987], *Automata Networks in Computer Science: Theory and Applications*, Princeton, 1987.

[29] M. Tchuente [1979], Parallel calculation of a linear mapping on a computer network. *Linear Algebra Appl.,* 28 (1979), 223–247.

[30] M. Tchuente [1979], Parallel realization of permutations over trees. *Discrete Math.,***39** (1982), 211–214.

[31] M. Tchuente [1983], Computation of Boolean functions on networks of binary automata. *J. Comput. Syst. Sci.* **26** (1983), 269–277.

[32] M. Tchuente [1985], Permutation factorization on star-connected networks of finite automata. *SIAM J. Algebraic Discrete Methods* **6** (1985), 537–540.

[33] M. Tchuente [1986], Computation on binary tree network. *Discrete Appl. Math.,* **14** (1986), 295–310.

[34] M. Tchuente [1988], Computation on finite networks of automata. In: C. Choffrut, ed., *Automata Networks (Argelès-Village, France, 1986)*, Lecture Notes in Computer Science, Vol. **316**, Springer Verlag, 53–67, 1988.

[35] T. Toffoli [1978], Integration of Phase-Difference Relations in Asynchronous Sequential Networks, In: G. Ausiello and C. Bohm, eds., *Automata, Languages, and Programming (Fifth Colloquium, Udine, July 1978)*, Lecture Notes in Computer Science **62**, Springer Verlag, pp. 457–463, 1978

[36] T. Toffoli and N. Margolus, *Cellular Automata Machines*, MIT Press, 1987.

[37] V. I. Varshavsky [1968], Collective behaviour and control problems. *Machine Intell.* **3**, 217–242 (1968).

[38] V. I. Varshavsky [1969], The organization of interaction in collectives of automata. *Machine Intell.* **4**, 285–311 (1969).

[39] V. I. Varshavsky (ed.), [1990], Self-timed control of concurrent processes. The design of aperiodic logical circuits in computers and discrete systems. Mathematics and its applications (Soviet Series), Dordrecht: Kluwer Academic Publishers, 1990.

[40] V. I. Varshavsky and B. L. Osievich [1965], Networks composed of ternary majority elements. *IEEE Trans. Electron. Comput.* **14**, 730-733 (1965).

Faculty of Engineering & Information Sciences, University of Hertfordshire, Hatfield Hertfordshire AL10 9AB, United Kingdom

E-mail address: C.L.Nehaniv@herts.ac.uk