Algebraic and informational aspects of Zielonka's Theorem

A. Bertoni – G. Mauri – G. Pighizzini – N. Sabadini
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
via Comelico, 39
20135 Milano – ITALY

Abstract

Zielonka's Theorem characterizes recognizable subsets of free partially commutative monoids in terms of *asynchronous automata*.

Here, we prove two results closely related to Zielonka's Theorem: first, we give a new representation theorem for finite monoids; next, we characterize the class of recognizable subsets of free partially commutative monoids in terms of a new model of distributed system with bounds on space and communication complexity.

1 Introduction

Asynchronous automata, as introduced by W. Zielonka [Zie87] in order to give a mathematical model of concurrent systems, are essentially nets of finite state automata (processors), with a distributed control structure. Each processor has its own local alphabet and state space. An action symbol belonging to more than one local alphabet is executed synchronizing the corresponding processors, while actions that require different processors could be executed concurrently. The resulting structure can be easily interpreted as an automaton over a free partially commutative monoid (fpcm): as a consequence, given a fpcm, the subsets accepted by asynchronous automata are recognizable subsets of the fpcm [BBMS81].

Surprisingly, despite their distributed structure, asynchronous automata exactly characterize the class of recognizable subsets, as witnessed by a deep result obtained by Zielonka [Zie87]: given an automaton on a fpcm, it is always possible to construct an asynchronous automaton, which accepts the same subset.

In this paper, in Section 3 we give a new representation theorem for finite monoids, obtained as a simple consequence of some methods introduced in the proof of Zielonka's theorem. In particular, we prove that every finite monoid $(M, \cdot, 1)$ is isomorphic to a quotient of a monoid $\mathcal{F}$ of transformations on a suitable finite set $S = S_1 \times \ldots \times S_n$, such that:

(i) for every $m \in M$ there is an element $f_m \in \mathcal{F}$, a subset $Dom(m)$ of $\{1, \ldots, n\}$ and a transformation $\tilde{f}_m : \prod_{i \in Dom(m)} S_i \to \prod_{i \in Dom(m)} S_i$ such that $f_m = \tilde{f}_m \times 1_{\overline{Dom(m)}}$, where $1_{\overline{Dom(m)}}$ is the identity on $\prod_{i \notin Dom(m)} S_i$;

(ii) given two transformations f_m and $f_{m'}$, associated with two distinct elements $m, m' \in M$, the product of m and m' is commutative, i.e., $mm' = m'm$, if and only if $Dom(m) \cap Dom(m') = \emptyset$, that is the transformations $\tilde{f}$ and $\tilde{f}'$ act on disjoint components of S.

In Section 4 we give a different interpretation of Zielonka's theorem in the area of distributed systems with bounds on space and communications.

We introduce a model of distributed system consisting of a net of off–line Turing Machines that exchange information using a simple communication protocol. The notion of *communication complexity* we introduce can be view as a natural extension to arbitrary fpcm of the analogous concept introduced in [BFS86] for products of free monoids. The main result states that the class of recognizable subsets of a finitely generated fpcm coincides with the behaviors of nets working in constant space and with a constant number of communications.

2 Preliminary notions and definitions

In this section we briefly recall basic notions and facts concerning trace languages and the algebraic structures supporting them, i.e., free partially commutative monoids.

Definition 2.1 *A* independence alphabet *is a pair* (Σ, I)*, where:*

- $\Sigma = \{a_1, \ldots, a_m\}$ *is a finite alphabet;*
- $I \subseteq \Sigma \times \Sigma$ *is a symmetric and irreflexive relation, the* independence *or* concurrence relation.

The complement of the independence relation I is called *dependence relation* and, in the following, will be denoted by $\overline{I}$.

As usual, the relations I and $\overline{I}$ can be represented using undirected graphs. More precisely, the *independence graph* associated with the independence alphabet (Σ, I) is the undirected graph with vertex set Σ and with edge set I, while the *dependence graph* of (Σ, I) is the undirected graph with vertex set Σ and with edge set $\overline{I} - \{(a, a) \mid a \in \Sigma\}$. Every clique of the dependence graph, i.e., every subset $\alpha \subseteq \Sigma$ such that $\alpha \neq \emptyset$ and $\forall a, b \in \alpha \; (a, b) \in \overline{I}$, will be called *dependence clique*.

Definition 2.2 *The* free partially commutative monoid $F(\Sigma, I)$ *generated by an independence alphabet* (Σ, I) *is the initial object of the category of monoids generated by the elements of* Σ *and satisfying, besides the monoid equations, the set of* *"commutativity laws":*

$$\{ab = ba \mid a, b \in \Sigma \text{ and } (a, b) \in I\}.$$

Definition 2.3 *A* trace *on an independence alphabet* (Σ, I) *is any element* $t \in F(\Sigma, I)$*; a trace language is any subset* $T \subseteq F(\Sigma, I)$.

In the usual representation, $F(\Sigma, I)$ is the quotient structure $F(\Sigma, I) = \Sigma^*/\equiv_I$ where $\equiv_I$ is the least congruence over Σ^* which extends the set of commutativity laws, and a trace is a congruence class of words. The congruence class containing the string $w \in \Sigma^*$ will be denoted by $[w]_I$, or simply by $[w]$ when I is understood; the composition on $F(\Sigma, I)$ is defined by $[w] \cdot [v] = [w \cdot v]$; the identity 1 is the class $[\epsilon]$, where ϵ denotes the empty string.

The *linearization* $lin(t)$ of a trace t is the set of all strings belonging to the equivalence class t, i.e., $lin(t) = \{x \in \Sigma^* \mid t = [x]\}$. The linearization of a trace language T is the union of linearizations of its traces, i.e., $lin(T) = \bigcup_{t \in T} lin(t)$.

We recall that every trace t can be represented as a partial order $ord(t)$. This order is formally defined in the following way:

Definition 2.4 *Given a trace* $t \in F(\Sigma, I)$*, let* $x = x_1 \ldots x_n$ $(x_i \in \Sigma,\ i = 1, \ldots, n)$ *be a representative string of* t*, i.e.,* $[x]_{\equiv_I} = t$*; the* partial order $ord(t)$ *associated to* t *is the pair* $ord(t) = (O_t, \leq_t)$*, such that:*

1. $O_t = \{(x_1, k_1), \ldots, (x_n, k_n)\}$*, where* k_s *denotes the number of symbols equal to* x_s *in the string* $x_1 \ldots x_s$*;*
2. $\leq_t$ *is the transitive closure of the relation* L *defined by:*

$$((x_i, k_i), (x_j, k_j)) \in L \text{ iff } (i \leq j \text{ and } (x_i, x_j) \notin I).$$

We recall that an *order ideal* of a partial order $ord = (O, \leq)$ is a subset O' of O such that $o \leq o'$ and $o' \in O'$ implies $o \in O'$. It is easy to see that the set of order ideals of ord with the inclusion relation is a lattice. Moreover, it is not difficult to prove that the set of prefixes of a trace t equipped with the prefix relation is isomorphic to the lattice of the order ideals of $ord(t)$ [BMS89]. Of course, given a prefix t' of t, the order ideal corresponding to t' is the set $O_{t'}$, with ordering relation $\leq_{t'} = \leq_t \cap (O_{t'} \times O_{t'})$.

Now we are able to define α–prefixes of traces.

Definition 2.5 *Let* (Σ, I) *be an independence alphabet,* t *a trace belonging to* $F(\Sigma, I)$ *and* α *a subset of the alphabet* Σ*. We denote by* $Pref_\alpha(t)$ *the order ideal generated by the set of occurrences in* t *of symbols belonging to* α*, i.e.,*

$$Pref_\alpha(t) = \{(x, k) \in O_t \mid \exists (x', k') \in O_t : \ (x, k) \leq_t (x', k') \text{ and } x' \in \alpha\}.$$

The α–prefix $\partial_\alpha(t)$ *is defined as the prefix of* t *corresponding to* $Pref_\alpha(t)$.

Properties of α–prefixes are extensively studied in [Zie87, CM88]. In this paper we are interested in the following property, useful to compute the α–prefix of a trace:

Lemma 2.1 *For $t \in F(\Sigma, I)$, $\alpha \subseteq \Sigma$, $a \in \Sigma$, we have:*

$$\partial_\alpha(ta) = \begin{cases} \partial_{\alpha \cup \overline{I}(a)} t \cdot a & \textit{if } a \in \alpha \\ \partial_\alpha t & \textit{otherwise,} \end{cases}$$

where $\overline{I}(a) = \{b \in \Sigma \mid (a, b) \notin I\}$.

In this paper we will consider the class of *recognizable trace languages.* In order to define this class it is useful to recall the standard notion of automata over a monoid.

Definition 2.6 *Let $(M, \cdot, 1)$ be a monoid. An* automaton *over M, or M–automaton, is a triple $\mathcal{A} = (Q, q_0, \delta)$ where:*

- *Q is a finite* set of states*;*
- *$q_0 \in Q$ is the* initial state*;*
- *$\delta : Q \times M \to Q$ is a* transition function *such that $\delta(q, 1) = q$ and $\delta(q, m \cdot m') = \delta(\delta(q, m), m')$ for $q \in Q$, $m, m' \in M$.*

Definition 2.7 *Given a monoid $(M, \cdot, 1)$ and two M–automata $\mathcal{A} = (Q, q_0, \delta)$, $\mathcal{A}' = (Q', q_0', \delta')$, a* morphism *$\phi$ from $\mathcal{A}$ to $\mathcal{A}'$ is a function $\phi : Q \to Q'$ such that $\phi(q_0) = q_0'$ (i.e., ϕ preserves the initial state) and $\phi(\delta(q, m)) = \delta'(\phi(q), m)$ for every $q \in Q$, $m \in M$ (i.e., ϕ preserves transitions).*

From Definition 2.6, it turns out that if M is the free partially commutative monoid $F(\Sigma, I)$ then $\delta(q, ab) = \delta(q, ba)$ for $(a, b) \in I$, $q \in Q$, i.e., the action of two independent letters a, b on a state q produces the same result independently on the order of execution. Moreover, the transition function δ is completely defined giving the values $\delta(q, a)$, for $q \in Q$ and $a \in \Sigma$. In fact, the only extension of δ to traces is obtained setting $\delta(q, 1) = q$ and $\delta(\delta(q, t), a) = \delta(q, ta)$, for $q \in Q$, $t \in F(\Sigma, I)$, $a \in \Sigma$.

Specifying also a set of final states, every M–automaton can be seen as an acceptor of subsets of M. A trace language $T \subseteq F(\Sigma, I)$ is said to be *recognizable* if and only if there is a $F(\Sigma, I)$–automaton accepting it. It is well known that T is recognizable if and only if its linearization $lin(T)$ is a regular subset of the free monoid Σ^*.

A different model of automata over free partially commutative monoids was introduced in [Zie87], defining *asynchronous automata.* In this model, the state set is the product of a family of sets of *local states* and independent symbols "locally" act on disjoint subfamilies of local states. Formally, we give the following definition:

Definition 2.8 *An* asynchronous automaton *over an independence alphabet* (Σ, I), *with* n *processes, is a tuple* $\mathcal{A} = (S_1, \ldots, S_n, s_{10}, \ldots, s_{n0}, Dom, \{\tilde{f}_\sigma\}_{\sigma\in\Sigma})$*, where:*

- *for* $i = 1, \ldots, n$*,* S_i *is the* ith *finite set of* local states *and* $s_{i0} \in S_i$ *is the* ith initial state*;*
- $Dom : \Sigma \to 2^{\{1,\ldots,n\}}$ *is a function associating to every symbol* $\sigma \in \Sigma$ *its "domain", such that* $(a, b) \in I$ *if and only if* $Dom(a) \cap Dom(b) = \emptyset$*;*
- *for every* $\sigma \in \Sigma$*,* $\tilde{f}_\sigma : \prod_{i\in Dom(\sigma)} S_i \to \prod_{i\in Dom(\sigma)} S_i$ *is the* local transition function *associated with the symbol* σ*.*

We observe that defining, for $i = 1, \ldots, n$, Σ_i as the set $\{\sigma \mid i \in Dom(\sigma)\}$, the family $(\Sigma_1, \ldots, \Sigma_n)$ is a clique cover of the dependence graph of (Σ, I).

It can be useful to see asynchronous automata as particular monoid automata. In fact, with every asynchronous automaton $\mathcal{A} = (S_1, \ldots, S_n, s_{10}, \ldots, s_{n0}, Dom, \{\tilde{f}_\sigma\}_{\sigma\in\Sigma})$ we associate the $F(\Sigma, I)$–automaton (S, q_0, δ) where $S = \prod_{i=1}^n S_i$ is the set of *global states*, $q_0 = (s_{10}, \ldots, s_{n0})$ is the *initial global state* and $\delta : S \times M \to S$ is the *global transition function* defined by $\delta(q, \sigma) = p$, for $q, p \in S$, $\sigma \in \Sigma$ if and only if $q = (s_1, \ldots, s_n)$, $p = (r_1, \ldots, r_n)$, $s_i = r_i$ for $i \notin Dom(\sigma)$ and $(r_{i_1}, \ldots, r_{i_k}) = \tilde{f}_\sigma(s_{i_1}, \ldots, s_{i_k})$, where $\{i_1, \ldots, i_k\} = Dom(\sigma)$. Intuitively, the transition function δ is obtained extending local transition functions to global states, in such a way that an action $\sigma \in \Sigma$ modifies, according to the local transition function $\tilde{f}_\sigma$, only the components of a local state corresponding to its domain. Observe that the ith component of the global state $\delta(q, t)$ depends only on q and $\partial_{\Sigma_i}(t)$, i.e., it coincides with the ith components of $\delta(q, \partial_{\Sigma_i}(t))$.

A morphism from an asynchronous automaton $\mathcal{A}$ to a $F(\Sigma, I)$–automaton $\mathcal{A}'$ is a morphism from the $F(\Sigma, I)$–automaton associated with $\mathcal{A}$

The most important result in the theory of asynchronous automata was obtained by W. Zielonka in [Zie87] and it can be formulated in the following way:

Theorem 2.1 *Given a* $F(\Sigma, I)$*–automaton* $\mathcal{A}$*, there exists an asynchronous automaton* $\mathcal{A}_s$ *with independence alphabet* (Σ, I) *and a morphism* ϕ *from* $\mathcal{A}_s$ *to* $\mathcal{A}$*.*

Actually, asynchronous automata were defined as acceptors of trace languages, specifying also a set of final global states; in the original formulation, Zielonka's Theorem states the equality between the classes of languages accepted by asynchronous automata over (Σ, I) and by $F(\Sigma, I)$–automata. So, asynchronous automata characterize recognizable subsets of free partially commutative monoids. Our formulation given in Theorem 2.1 is an immediate consequence of that result.

3 A representation theorem for finite monoids

It is well known that every finite monoid M of N elements is isomorphic to a transformation monoid $\widehat{M} \subseteq N^N$. The following theorem improves that result considering pairs of elements of M whose product is commutative.

Theorem 3.1 *Every finite monoid $(M, \cdot, 1)$ is isomorphic to a quotient of a monoid $\mathcal{F}$ of transformations on a set S, such that:*

(i) *S is the product of n finite sets $S_1, \ldots, S_n$, $n \geq 1$;*

(ii) *for every $m \in M$ there is an element $f_m \in \mathcal{F}$, a subset $Dom(m)$ of $\{1, \ldots, n\}$ and a "local" transformation $\tilde{f}_m : \prod_{i \in Dom(m)} S_i \to \prod_{i \in Dom(m)} S_i$ such that $f_m = \tilde{f}_m \times 1_{\overline{Dom(m)}}$, where $1_{\overline{Dom(m)}}$ is the identity on $\prod_{i \notin Dom(m)} S_i$;*

(iii) *given two transformations f_m and $f_{m'}$, associated with two distinct elements $m, m' \in M$, the product of m and m' is commutative, i.e., $mm' = m'm$, if and only if the "local" transformations $\tilde{f}$ and $\tilde{f}'$ act on disjoint components of S, i.e., $Dom(m) \cap Dom(m') = \emptyset$.*

Proof. Given the monoid $(M, \cdot, 1)$, we consider the independence alphabet (Σ, I) where $\Sigma = M$, $I = \{(m, m') \in M \times M \mid m, m' \in M\}$, and the $F(\Sigma, I)$–automaton $\mathcal{A} = (Q, \delta, q_0)$ with $Q = M$, $q_0 = 1$ and $\delta(m, \sigma) = m \cdot \sigma$, for $m \in Q$, $\sigma \in \Sigma$.

It is not difficult to verify that $\delta(m, t) = m \cdot \delta(1, t)$, for $m \in M$ and $t \in F(\Sigma, I)$.

From Theorem 2.1, it turns out that there exists an asynchronous automaton $\mathcal{A}_s = (S_1, \ldots, S_n, s_{10}, \ldots, s_{n0}, Dom, \{\tilde{f}_\sigma\}_{\sigma \in \Sigma})$ and a morphism ϕ from $\mathcal{A}_s$ to $\mathcal{A}$.

For $\sigma \in \Sigma$, let f_σ be the transformation $\tilde{f}_\sigma \times 1_{\overline{Dom(m)}}$, obtained extending $\tilde{f}_\sigma$ to global states.

We consider the monoid $(\mathcal{F}, \circ, I)$, where $\mathcal{F} = \{f_\sigma \mid \sigma \in M\}^*$, $\circ$ denotes the composition of transformations, and I is the identity on S. We observe that $(\sigma, \gamma) \in I$ implies $f_\sigma \circ f_\gamma = f_\gamma \circ f_\sigma$. Thus, we can associate with every trace $t \in F(\Sigma, I)$ an element $f_t \in \mathcal{F}$, setting: $f_\epsilon = I$ and $f_{t\sigma} = f_t \circ f_\sigma$, for $t \in F(\Sigma, I)$, $\sigma \in \Sigma$. It is not difficult to see that $f_t(s_{10}, \ldots, s_{n0})$ is the global state reached by the automaton $\mathcal{A}_s$ executing the trace t from the initial global state, i.e., $f_t(s_{10}, \ldots, s_{n0}) = \delta_s((s_{10}, \ldots, s_{n0}), t)$, where δ_s is the global transition function of $\mathcal{A}_s$.

Given $t', t'' \in F(\Sigma, I)$, $f_{t'} = f_{t''}$ implies $\delta(1, t') = \delta(1, t'')$. In fact, $f_{t'}(s_{10}, \ldots, s_{n0}) = \delta_s((s_{10}, \ldots, s_{n0}), t')$ and $f_{t''}(s_{10}, \ldots, s_{n0}) = \delta_s((s_{10}, \ldots, s_{n0}), t'')$; since there is a morphism from $\mathcal{A}_s$ to $\mathcal{A}$, $\delta_s((s_{10}, \ldots, s_{n0}), t') = \delta_s((s_{10}, \ldots, s_{n0}), t'')$ implies $\delta(1, t') = \delta(1, t'')$.

Now, we consider the function $\psi : \mathcal{F} \to M$ defined by:

$$\psi(f) = m \text{ if and only if } \exists t \in F(\Sigma, I)(f = f_t \ \wedge \ \delta(1, t) = m).$$

From the last observation, it turns out that this definition of ψ is sound. Now, we show that ψ is an homomorphism.

First, we observe that $\psi(I) = 1$, in fact $I = f_\epsilon$ and $\delta(1, \epsilon) = 1$. Given $f', f'' \in \mathcal{F}$, let $t', t'' \in F(\Sigma, I)$, $m', m'' \in M$ such that $f' = f_{t'}$ and $f'' = f_{t''}$, $m' = \psi(f')$ and $m'' = \psi(f'')$. We have:

$$\begin{aligned} \psi(f' \circ f'') &= \psi(f_{t'} \circ f_{t''}) = \psi(f_{t' \cdot f''}) = \delta(1, t't'') = \delta(\delta(1, t'), t'') \\ &= \delta(m', t'') = m' \cdot \delta(1, t'') = m' \cdot m'' = \psi(f') \cdot \psi(f''). \end{aligned}$$

Hence, ψ is an homomorphism. At this point, it is not difficult to conclude that M is isomorphic to the quotient of $\mathcal{F}$ with respect to the congruence induced by the morphism ψ. ∎

4 Recognizable Trace Languages and Communication Complexity

In this section we relate the main Theorem of Zielonka to *communication complexity*. In fact we present a model of distributed system constituted by off–line Turing Machines exchanging information using a simple communication protocol, and we show that with constant bounds on the space of computation and on communication, this model characterizes the class of recognizable trace languages.

Let (Σ, I) be an independence alphabet and $\Sigma_1, \ldots, \Sigma_m$ a clique cover of its dependence graph. We consider a net $N = (M_1, \ldots, M_m)$, where any M_i is a Turing Machine with a read only input tape and a work tape; the input alphabet of M_i is Σ_i. Machine M_i can send messages to machine M_j through an unidirectional communication channel, with a FIFO strategy. When the machine M_i enters in a particular state $q_!^{(ij)}$, it put a bit in the queue of messages from M_i to M_j; when M_i enters in a particular state $q_?^{(ji)}$, it get a bit from the queue of messages from M_j to M_i or it waits if the queue is empty.

Now we explain how a net of this kind can recognize trace languages. Given a trace $t \in F(\Sigma, I)$, the machine M_i receives on its input tape a whatever representative string x_i of the prefix $\partial_{\Sigma_i}(t)$, i.e., $x_i \in lin(\partial_{\Sigma_i}(t))$. With this input the machine starts its computation. During the computation the machine can send or receive messages from other machines as explained above. The machine M_1 has an accepting state q_Y and a rejecting state q_N. We say that the net is *sound* whether, for every trace t, every machine M_i ends its computation in a finite number of steps and the machine M_1 halts in a state $q \in \{q_Y, q_N\}$, independently on the choice of representative strings $x_1, \ldots, x_m$; in other words, this means that the acceptation or the rejection does not depend on the choice of representative strings.

We say that the *space complexity* of a net N on a trace t is the maximum number of cells on work tapes scanned during the computation, while the *communication complexity* is the maximum number of bits exchanged.

A sound net N recognizes a trace language $T \subseteq F(\Sigma, I)$ if and only if for every $t \in T$ the machine M_1 halts in q_Y and for every $t \notin T$ the machine M_1 halts in q_N. We denote by $\mathcal{C}_{(\Sigma,I)}(f(n), g(n))$ the class containing trace languages $T \subseteq F(\Sigma, I)$ which are recognizable by nets N within space and communication complexity bounded respectively, on traces of length n, by $cf(n)$ and $cg(n)$, where c is a suitable constant depending only on N.

Theorem 4.1 *The class of recognizable subsets of a free partially commutative monoid $F(\Sigma, I)$ with a finite number of generators coincides with the class $\mathcal{C}_{(\Sigma,I)}(1, 1)$.*

Proof. $\subseteq$) If $T \subseteq F(\Sigma, I)$ is a recognizable trace language, then there exists an asynchronous automaton $\mathcal{A}$ with m processes recognizing it. We construct a net N of m machines $M_1, \ldots, M_m$ that work in this way. Given an input trace $t \in F(\Sigma, I)$, the machine M_i simulates the computation of the asynchronous automaton $\mathcal{A}$ on the trace $\partial_{\Sigma_i}(t)$. When the machine M_i, $i > 1$, ends its simulation, it sends the global state reached to the machine M_1. Collecting these states, M_1 can reconstruct the global state reached by the asynchronous automaton $\mathcal{A}$ on input t and so, it can decide whether or not t belongs to T. Since the space and the amount of bits exchanged during the simulation do not depend on the input trace t, we can conclude that the language T belongs to the class $\mathcal{C}_{(\Sigma,I)}(1,1)$.

$\supseteq$) To prove the inclusion of the class $\mathcal{C}_{(\Sigma,I)}(1,1)$ in the class of recognizable trace languages, we sketch an algorithm for simulating a net $N = (M_1, \ldots, M_m)$ recognizing a trace language T, using a one tape Turing Machine M working in linear time. Since, as shown in [Hen65], every language accepted by a machine with these constraints is regular, we can conclude that T is a recognizable trace language.

Given a net N we consider strings on the alphabet $\{q_?^{ij}, q_!^{ij}0, q_!^{ij}1\}$. Such a string can represent a sequence of operations "send" or "receive" among machines of the net. However, some strings do not correspond to safe computations of the machine; for instance, the string $q_?^{ij}q_?^{ji}$ represents a deadlock (M_i waits a bit from M_j and vice versa). With a precomputation, we recover the set D of strings representing "safe" sequences of communications in the net N. Note that this set is finite since N has communication complexity $O(1)$ and it does not depend on the input trace.

Now we explain how the machine M can simulate the computation of the net N on a trace $t \in F(\Sigma, I)$. The machine starts its computation with a representative string $w \in lin(t)$ on its tape; for every "safe" sequence d of communications, i.e., $d \in D$, M simulates the computations of $M_1, M_2, \ldots, M_m$ using the string d for the communications. The machine accepts if and only if M_1 can reach the accepting state q_Y in a simulation involving a string $d \in D$. More precisely, we can describe the computation of M with the following program:

```
f := 0
for all d ∈ D do
begin
    i:=0
    while i ≤ m and compatible(M_i, d) do i := i + 1
    if i = m + 1 and the last state reached by M_1 is q_Y then f := 1
end
if f = 1 then accepts
        else reject
```

The test $compatible(M_i, d)$ can be performed in the following way:

- M scans the string w on its tape from right to left marking the symbols belonging to the prefix of $\partial_{\Sigma_i}(t)$ (see Lemma 2.1);
- M simulates M_i (skipping all not marked symbols), using the string d in order to obtain in the states $q_?^{ij}$ the bits from other processes;

- the answer to the test is TRUE if and only if the sequence of symbols $q_!^{ij}x$, $x \in \{0,1\}$, and $q_?^{ij}$ matches with the sequence of messages sent and of interrogation states of M_i during the simulated computation;

- after the simulation of M_i, the machine unmarks the input symbols of $\partial_{\Sigma_i}(t)$; so, at the end of the simulation of M_i, the content of the tape is the string w.

At this point, it is not difficult to verify that the string w is accepted by M if and only if the trace t is accepted by the given net N.

Now we show that M works in linear time. The machine M_i uses space $O(1)$ and its computation is finite; so, M can be seen as a two way finite automaton whose head inverts the direction a constant number of times [HU79]. Thus, the test $compatible(M_i, d)$ can be done in time $O(n)$, where $n = |w|$. This test is repeated for every machine M_i and for every $d \in D$. But the number of machines M_i and the set D do not depend on the input. This permits us of concluding that the running time of M is $O(n)$.

Since the machine M has only one tape and it works in linear time, as shown in [Hen65], the language L recognized by it is a regular subset of Σ^*. On the other hand, the given net N accepts the trace language T. From the above given simulation, we can easily conclude that $L = lin(T)$. Thus T is a recognizable trace language. ∎

5 Conclusions and open problems

The main results obtained in this paper are:

(1) The Theorem of Zielonka (Theorem 2.1) is related to a representation result for finite monoids (Theorem 3.1).

(2) It is given an interpretation of Zielonka's Theorem in terms of communication complexity.

With respect to (1), we observe that Theorem 3.1 easily implies Zielonka's Theorem. Since the direct proof of this theorem is very complicated, and it seems very difficult to obtain some meaningful simplification, despite all efforts (see, e.g., [CM88, CMZ89]), an interesting open problem is then to study a direct proof of our result, using more algebraic oriented techniques.

With respect to (2), we leave as an open problem to characterize other classes of languages recognized by nets of machines with given non constant bounds in space and communication complexity.

References

[BBMS81] A. Bertoni, M. Brambilla, G. Mauri, and N. Sabadini. An application of the theory of free partially commutative monoids: asymptotic densities of trace languages. In *Proc. 10th MFCS*, Lecture Notes in Computer Science 118, pages 205–215, 1981.

[BFS86] L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory. In *Proc. 27th Symposium on Foundations of Computer Science*, pages 337–347, 1986.

[BMS89] A. Bertoni, G. Mauri, and N. Sabadini. Membership problem for regular and context–free trace languages. *Information and Computation*, 82:135–150, 1989.

[CM88] R. Cori and Y. Métivier. Approximation of a trace, asynchronous automata and the ordering of events in a distributed system. In *Proc. 15th ICALP*, Lecture Notes in Computer Science 317, pages 147–161, 1988.

[CMZ89] R. Cori, Y. Métivier, and W. Zielonka. Asynchronous mappings and asynchronous cellular automata. Technical Report 89–97, LaBRI, Université de Bordeaux I, 1989.

[Hen65] F. Hennie. One–tape, off–line turing machine computations. *Information and Control*, 8:553–578, 1965.

[HU79] J. Hopcroft and J. Ullman. *Introduction to automata theory, languages and computations*. Addison–Wesley, 1979.

[Zie87] W. Zielonka. Notes on finite asynchronous automata. *RAIRO Inf. Theor.*, 21:99–135, 1987.