Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 21, n° 2, 1987, p. 99 à 135)

NOTES ON FINITE ASYNCHRONOUS AUTOMATA (*)

by Wieslaw ZIELONKA ([1])

Communicated by A. ARNOLD

Abstract. – *We introduce the notion of finite asynchronous automata. Having ability of simultaneous execution of independent actions, these automata are used in a natural way as recognizing devices for subsets of free partially commutative monoids. We prove that a subset of a f.p.c. monoid is recognizable by a finite asynchronous automaton iff it is recognizable by a finite automaton. As a corollary we obtain a new characterization of the recognizable subsets of the f.p.c. monoids by means of a parallel composition and certain homomorphisms.*

Résumé. – *On introduit la notion d'automate fini asynchrone. Ayant la possibilité d'effectuer simultanément des actions indépendantes, ces automates sont utilisés d'une façon naturelle pour reconnaître des sous-ensembles d'un monoïde partiellement commutatif libre. On prouve qu'un sous-ensemble de ce monoïde est reconnaissable par un de ces automates si et seulement s'il est reconnaissable par un automate fini. Comme conclusion on obtient une caractérisation nouvelle des sous-ensembles reconnaissables des monoïdes partiellement commutatifs libres à l'aide d'une composition parallèle et de certains homomorphismes.*

1. INTRODUCTION

Let Σ be a finite alphabet on which a symmetric and irreflexive relation $I \subset \Sigma \times \Sigma$ is defined. Intuitively, I is a concurrency relation and $(a, b) \in I$ indicates that the actions a and b can be executed simultaneously. With the concurrent alphabet (Σ, I) there is associated the congruence relation $\sim$ over Σ^* generated by $\{ ab \equiv ba: (a, b) \in I \}$. The free partially commutative monoid over (Σ, I), denoted by $E(\Sigma, I)$, is the quotient of Σ^* by the congruence $\sim$ and traces are elements of this monoid.

The study of the free partially commutative monoids was initiated by Cartier and Foata [3] in 1969, but only in 1977 traces were used by Mazurkiewicz [12] as a tool for describing the behaviour of concurrent systems. Since then a number of papers has been devoted to various aspects of the theory

(*) Received August 1986, Revised October 1986.
([1]) Institute of Mathematics, Warsaw Technical University, Pl. Jedności Robotniczej 1, 00-661 Warsaw, Poland.

Informatique théorique et Applications/Theoretical Informatics and Applications
0296-1598 87/02 99 36/$5.60/© Gauthier-Villars

of traces [1, 2, 4 to 7, 13]. On the other hand very little is known about parallel devices accepting traces. In fact, after the pioneering paper of Mazurkiewicz only a few papers, e. g. [10, 17], dealt to some extent with this problem. This situation is even more surprising if we compare it with the development of the theory of formal languages, which has been inspired in great part by the automata theory.

The aim of this paper is to study parallel finite state devices recognizing traces. The paper is organized as follows. After some preliminary results in Sections 2 and 3, we introduce in Section 4 a class of finite asynchronous automata and we prove our main result that they recognize exactly all regular trace languages. In Section 5 we present another class of parallel automata, with a simpler synchronization mechanism. In Section 6 we use results of the two previous sections to obtain a new characterization of regular trace languages.

Throughtout the paper we shall use the following notation. card (X) will stand for the cardinality of a set X, $\mathscr{P}(X)$ for the family of all subsets of X, the empty word will be represented by ε, $\#_a u$ is the number of occurrences of a letter $a\in\Sigma$ in a word $u\in\Sigma^*$, whereas $\#u$ is the length of u. For a positive integer n by $\bar{n}$ we denote the set $\{1,\ldots,n\}$. The shuffle operation is defined on Σ^* as follows.

$$\forall u, v\in\Sigma^*,\quad \mathrm{sh}(u, v)=\{u_1 v_1\ldots u_n v_n : \forall i\in\bar{n},\ u_i, v_i\in\Sigma^*,\ u=u_1\ldots u_n,\ v=v_1\ldots v_n\}.$$

If $L_1, L_2\subset\Sigma^*$ then

$$\mathrm{sh}(L_1, L_2)=\bigcup_{u\in L_1,\, v\in L_2}\mathrm{sh}(u, v).$$

If R is a binary relation over a finite set X then by Cliques (R) we denote the family of all cliques of R, $A\in\mathrm{Cliques}(R)$ if $\forall a, b\in A$, $(a, b)\in R$ and $\forall c\in X-A$, $\exists a\in A$, $(a, c)\notin R$, while $R\mid Y$ will stand for the restriction of R to a subset Y of X.

2. TRACES AND TRACE LANGUAGES

In this section we describe elementary properties of traces. None of the presented here results seems to be original, in fact, most of them are folklore. Proofs are for the most part very simple, so we decided to sketch them only in a few cases.

A pair (Σ, I) is a concurrent alphabet if Σ is a finite and nonempty set of actions and $I\subset\Sigma\times\Sigma$ is a symmetric and irreflexive relation over Σ (the independency relation).

Two words u and v are congruent, $u\sim_I v$, or $u\sim v$ if I fixed, if there exist words $w_1,\ldots, w_{k+1}$ such that $u=w_1$, $v=w_{k+1}$ and $\forall i\in k$, $\exists x, y\in\Sigma^*$, $\exists (a, b)\in I$, $w_i=xaby$ and $w_{i+1}=xbay$.

PROPOSITION 2.1 [6]: *Let $h_{a,b}$ be the projection of Σ^* onto $\{a, b\}^*$. Then two words u, v are equivalent, $u\sim v$, iff*

(i) $\forall a\in\Sigma$, $\#_a u=\#_a v$ *and*

(ii) $\forall (a, b)\notin I$, $h_{a,b}(u)=h_{a,b}(v)$. □

By definition, the quotient $E(\Sigma, I)=\Sigma^*/\sim$ of the free monoid Σ^* by $\sim$ is the free partially commutative monoid over (Σ, I). Its elements are called traces. In the sequel $[u]_I$, or $[u]$ if I fixed, will stand for the trace represented by the word $u\in\Sigma^*$, if $u=\varepsilon$ or $u=a$, $a\in\Sigma$, then we shall write ε and a to denote the traces $[\varepsilon]=\{\varepsilon\}$ and $[a]=\{a\}$. t, p, r with or without subscripts will denote traces. By definition, trace languages are subsets of $E(\Sigma, I)$. For $T\subset E(\Sigma, I)$ by lin T we denote the language.

$$\bigcup T=\{u\in\Sigma^* : \exists t\in T,\ u\in t\}.$$

The following property of traces proved to be very useful.

PROPOSITION 2.2 [11]: *The monoid $E(\Sigma, I)$ is cancellative, i. e.*

$$\forall t, t_1, t_2\in E(\Sigma, I),\quad tt_1=tt_2\ \Rightarrow\ t_1=t_2$$

and

$$t_1 t=t_2 t\ \Rightarrow\ t_1=t_2.\ \square$$

The number of occurrences of an action $a\in\Sigma$ in a trace $t\in E(\Sigma, I)$ and the length of t are defined as follows $\#_a t=\#_a u$ and $\#t=\#u$ for $u\in t$.

Every word $u\in\Sigma^*$ generates a linear order $\leqq_u$ over the set

$$O(u)=\{a^i : a\in\Sigma,\ 1\leqq i\leqq\#_a u\}$$

of action occurrences. For instance, if $u=abbacc$ then

$$a^1\leqq_u b^1\leqq_u b^2\leqq_u a^2\leqq_u c^1\leqq_u c^2.$$

Formally, $a^i\leqq_u b^j$ if either $a=b$ and $1\leqq i\leqq j\leqq\#_a u$ or $a\neq b$ and $u=vawbz$, where $\#_a v=i-1$, $\#_b vaw=j-1$. On the other hand every trace $t\in E(\Sigma, I)$

generates the canonical partial order $\leqq_t$ over the set

$$O(t)=\{a^i: a\in\Sigma,\quad 1\leqq i\leqq \#_a t\}$$

of action occurrences:

$$a^i\leqq_t b^j \iff \forall u\in t,\quad a^i\leqq_u b^j,\quad \text{i.e. } \leqq_t=\bigcap_{u\in t}\leqq_u.$$

This partial order has the following properties.

PROPOSITION 2.3 [5]: *Let $v\in\Sigma^*$ and $t\in E(\Sigma, I)$. Then $v\in t$ iff $O(t)=O(v)$ and $\leqq_v$ is an extension of $\leqq_t$ to a linear order, i.e. $\leqq_t\subset\leqq_v$.* □

COROLLARY 2.4: *Traces t and r are equal iff $\leqq_t=\leqq_r$.* □

As usual, $a^i<_t b^j$ will denote that $a^i\leqq_t b^j$ and $a^i\neq b^j$. If the trace t is fixed then we shall simply write $\leqq$ and $<$.

The representation of traces by partial orders will be extensively used in the next sections. For this reason we introduce further notational conventions.

$$O(\Sigma)=\{a^i: a\in\Sigma,\ i\in N\}$$

will be the set of all action occurrences. Evidently, $O(t)\subset O(\Sigma)$ for any trace t.

name: $O(\Sigma)\to\Sigma$ is a projection of $O(\Sigma)$ onto Σ defined as follows

$$\forall a\in\Sigma,\quad \forall i\in N,\qquad \text{name}(a^i)=a.$$

In the sequel x, y, z will stand for elements of $O(\Sigma)$. We now give properties that completely characterize the partial order $\leqq_t$.

Fact 2.5: Let $t\in E(\Sigma, I)$, $x,\ y\in O(t)$ such that $x<_t y$ and $\neg\exists z\in O(t)$, $x<_t z<_t y$. Then $(\text{name}(x), \text{name}(y))\in D$. On the other hand if $(\text{name}(x), \text{name}(y))\in D$ then $x\leqq_t y$ or $y\leqq_t x$. □

A trace r is a subtrace of t if there exist traces t_1, t_2 such that $t=t_1 r t_2$. If $t_1=\varepsilon$ then r is a prefix of t, whereas if $t_2=\varepsilon$ then r is a suffix of t.

The following notions will be extensively used in the next sections.

Let $t\in E(\Sigma, I)$ and $H\subset O(t)$. Then H is

(1) initial in t if

$$\forall x\in H,\quad \forall z\in O(t),\qquad z\leqq_t x\ \Rightarrow\ z\in H$$

(2) final in t if

$$\forall x\in H,\qquad \forall z\in O(t),\qquad x\leqq_t z\ \Rightarrow\ z\in H.$$

Fact 2.6: H is initial in t iff there exists a prefix r of t such that $H=O(r)$ and then $\leqq_r=\leqq_t|H$. H is final in t iff there exists a suffix r in t and an isomorphism $\varphi: H\to O(r)$ of partial orders $\leqq_t|H$ and $\leqq_r$ such that

$$\forall x\in H,\quad \text{name}(\varphi(x))=\text{name}(x).$$ □

Clearly, if H is empty then $r=\varepsilon$.

If H_1, H_2 are initial (final) in t then $H_1\cup H_2$ and $H_1\cap H_2$ are initial (final).

The representation of traces by partial orders was already noticed by Mazurkiewicz [12]. Detailed analysis of these connections was made by Shields [15].

We now introduce a special kind of trace homomorphisms. Let (Σ_1, I_1) and (Σ_2, I_2) be concurrent alphabets. A homomorphism $f: \Sigma_1^*\to\Sigma_2^*$ is consistent with I_1 and I_2 if

(i) f is strictly alphabetic, i.e. $f(\Sigma_1)\subset\Sigma_2$ and

(ii) $\forall a, b\in\Sigma_1$, $(a, b)\in I_1\Leftrightarrow(f(a), f(b))\in I_2$.

Thus f preserves both dependency and independency between actions.

LEMMA 2.7: *Let f be consistent with I_1 and I_2. Then, for all*

$$t_1\in E(\Sigma_1, I_1),\quad f(t_1)=\{f(u): u\in t_1\}\in E(\Sigma_2, I_2).$$

Proof: Immediate from Proposition 2.1. □

COROLLARY 2.8: *Let f be consistent with I_1 and I_2. Then*

(i) *if $t=[u]_{I_1}\in E(\Sigma_1, I_1)$ then $f(t)=[f(u)]_{I_2}$*

(ii) *the mapping $f: E(\Sigma_1, I_1)\to E(\Sigma_2, I_2)$ defined by $f(t)=\{f(u): u\in t\}$, $t\in E(\Sigma_1, I_1)$, is a homomorphism of free partially commutative monoids.*

Proof: (i) Obvious by Lemma 2.7.

(ii) Let $t_1, t_2\in E(\Sigma_1, I_1)$, $u_1\in t_1$, $u_2\in t_2$. Then

$$f(t_1)f(t_2)=[f(u_1)]_{I_2}[f(u_2)]_{I_2}=[f(u_1)f(u_2)]_{I_2}=[f(u_1u_2)]_{I_2}=f(t_1t_2).$$

The last equality holds because $u_1u_2\in t_1t_2$. □

A trace homomorphism generated by a consistent homomorphism of words will be called elementary. Let us describe the elementary homomorphisms in terms of partial orders.

COROLLARY 2.9: *Let f be an elementary homomorphism of traces. Then for all $t_1\in E(\Sigma_1, I_1)$, $t_2=f(t_1)$ there exists an isomorphism $i: O(t_1)\to O(t_2)$ of*

partial orders $\leqq_{t_1}$ *and* $\leqq_{t_2}$ *such that*

$$\forall x \in O(t_1), \quad f(\text{name}(x)) = \text{name}(i(x)).$$

Proof: Obvious. □

Corollary 2.8 implies that an elementary homomorphism changes only names of actions but maintains the partial order between them. A similar concept was previously introduced by Tarlecki [17].

The next operation we shall present, a parallel composition of trace languages, is of great importance. It enables us to construct parallel systems from sequential components.

Let (Σ_i, I_i), $D_i = \Sigma_i \times \Sigma_i - I_i$, $i \in \bar{n}$, be concurrent alphabets, and their dependency relations, where Σ_i are not necessarily disjoint. Let $\Sigma = \bigcup_{i=1}^{n} \Sigma_i$, $D = \bigcup_{i=1}^{n} D_i$, $I = \Sigma \times \Sigma - D$.

The concurrent alphabet (Σ, I) is said to be the parallel composition of the alphabets (Σ_i, I_i), $i \in \bar{n}$, and is denoted by $\| \, |_{i=1}^{n} (\Sigma_i, I_i)$. Obviously, the parallel composition of alphabets is commutative and associative.

PROPOSITION 2.10: *Let*

$$\forall i \in \bar{n}, \quad t_i \in E(\Sigma_i, I_i), \qquad (\Sigma, I) = \| \, |_{i=1}^{n} (\Sigma_i, I_i)$$

and let h_i *be projections of* Σ^* *onto* Σ_i^*. *Then the set*

$$R = \{ u \in \Sigma^* : \forall i \in \bar{n}, \ h_i(u) \in t_i \}$$

either is a trace over (Σ, I) *or is empty.*

Proof: Suppose that $R \neq \varnothing$. Let $u \in R$ and $v \sim_I u$. Then it is easy to observe that $\forall i \in n$, $h_i(v) \sim_{I_i} h_i(u)$ and hence $v \in R$. On the other hand, let $u, v \in R$. It is obvious that $\forall a \in \Sigma$, $\#_a u = \#_a v$. Let $(a, b) \in D$. Then there exists $i \in \bar{n}$ such that $(a, b) \in D_i$. Moreover

$$\forall w \in t_i, \quad h_{a,b}(u) = h_{a,b}(w) = h_{a,b}(v),$$

where $h_{a,b}$ is the projection onto $\{a, b\}^*$. Thus by Proposition 2.1 $u \sim_I v$. □

The set R from Proposition 2.10, henceforth denoted by $\| \, |_{i=1}^{n} t_i$, will be called the parallel composition of traces.

COROLLARY 2.11: *If* $t_i \in E(\Sigma_i, I_i)$, $i = 1, 2$, *then*

$$t_1 \| t_2 = \text{sh}((\Sigma_2 - \Sigma_1)^*, t_1) \cap \text{sh}((\Sigma_1 - \Sigma_2)^*, t_2). \quad \square$$

PROPOSITION 2.12: *Let* $t \in E(\Sigma, I)$, *Cliques* $(D) = \{\Sigma_1, \ldots, \Sigma_n\}$ *and let for all* $i \in \bar{n}$ h_i *be the projection of* Σ^* *onto* Σ_i^*. *Then* $t = \| \, |_{i=1}^{n} h_i(t)$.

Proof: First observe that $\forall u, v \in t$, $h_i(u) = h_i(v)$ thus $h_i(t)$ is a one element set. It may be viewed as a trace over the concurrent alphabet $(\Sigma_i, \varnothing)$ with the empty independency relation. The thesis follows then from Propositions 2.1 and 2.10. □

Let $T_i \subset E(\Sigma_i, I_i)$ for $i \in \bar{n}$, $(\Sigma, I) = \| \, |_{i=1}^{n} (\Sigma_i, I_i)$.

We define the parallel composition of trace languages to be the set

$$\| \, |_{i=1}^{n} T_i = \{ t \in E(\Sigma, I) : \forall i \in \bar{n}, \ \exists t_i \in T_i, \ t = \| \, |_{i=1}^{n} t_i \}$$

The parallel composition is associative and commutative.

COROLLARY 2.13: *Let* $T_i \subset E(\Sigma_i, I_i)$, $i = 1, 2$. *Then*

$$\text{lin } T_1 \| \, | \, T_2 = \text{sh}((\Sigma_2 - \Sigma_1)^*, \text{lin } T_1) \cap \text{sh}((\Sigma_1 - \Sigma_2)^*, \text{lin } T_2). \quad \square$$

The parallel composition of traces can be described in terms of partial orders in the following way.

PROPOSITION 2.14: *Let* $\forall i \in \bar{n}$, $t_i \in E(\Sigma_i, I_i)$. *Then* $t = \| \, |_{i=1}^{n} t_i \neq \varnothing$ *if the following conditions hold*

(i) $\forall i, j \in \bar{n}$, $\forall a \in \Sigma_i \cap \Sigma_j$, $\#_a t_i = \#_a t_j$

(ii) $\left(\bigcup_{i=1}^{n} \leqq_{t_i} \right)^*$ *is a partial order, where* * *denotes the transitive closure of a binary relation.*

Moreover, the partial order computed in (ii) is equal to $\leqq_t$.

Proof: Elementary. □

The presented here parallel composition relates closely to the operation of restriction examined by Starke [16] in connection with Petri net languages.

3. REGULAR TRACE LANGUAGES

Let $T \subset E(\Sigma, I)$ be a trace language. The syntactic congruence $\sim_T$ of T is defined by $\forall t, r \in E(\Sigma, I)$, $t \sim_T r$ iff

$$\forall t_1, t_2 \in E(\Sigma, I), \quad t_1 t t_2 \in T \iff t_1 r t_2 \in T.$$

The fact that $\sim_T$ is really a congruence can be deduced in exactly the same way as in case of the syntactic congruence of string languages, *see* e. g. [11].

LEMMA 3.1: *Let* $u, v \in \Sigma^*$, $T \subset E(\Sigma, I)$, $L = \text{lin } T$. *Then* $[u] \sim_T [v]$ *iff* $u \sim_L v$, *where* $\sim_L$ *is the syntactic congruence of the language* L.

Proof:

$$u \sim_L v \quad \text{iff } \forall x, y \in \Sigma^*,$$
$$xuy \in L \Leftrightarrow xvy \in L \quad \text{iff } \forall x, y \in \Sigma^*,$$
$$[xuy] \in T \Leftrightarrow [xvy] \in T \quad \text{iff } \forall x, y \in \Sigma^*,$$
$$[x][u][y] \in T \Leftrightarrow [x][v][y] \in T \quad \text{iff } [u] \sim_T [v]. \quad \square$$

A trace language T is said to be regular iff the syntactic congruence $\sim_T$ is of finite index. The family of regular trace languages over (Σ, I) will be denoted by Reg (Σ, I). The preceding Lemma shows that the syntactic congruences of T and lin T are isomorphic. This implies

COROLLARY 3.2: $T \in \text{Reg}(\Sigma, I)$ *iff* lin T *is a regular language over* Σ. $\square$

PROPOSITION 3.3 [6]: *If* $T_1, T_2 \subset \text{Reg}(\Sigma, I)$ *then*

$$T_1 \cup T_2, \quad T_1 \cap T_2, \quad T_1 . T_2 = \{ t_1 t_2 : t_1 \in T_1, t_2 \in T_2 \},$$
$$E(\Sigma, I) - T_1 \in \text{Reg}(\Sigma, I). \quad \square$$

As it is well-known, in general, the family $\text{Reg}(\Sigma, I)$ is not closed under star operation and therefore, contrary to free monoids, in partially commutative monoids regularity does not coincide with rationality.

LEMMA 3.4: *If* $T_i \in \text{Reg}(\Sigma_i, I_i)$, $i = 1, 2$, *then*

$$T_1 \| T_2 \in \text{Reg}(\Sigma, I), \qquad \text{where} \quad (\Sigma, I) = (\Sigma_1, I_1) \| (\Sigma_2, I_2).$$

Proof: The family of regular languages is closed under the shuffle operation thus by Corollary 2.13 lin $T_1 \| T_2$ is regular. $\square$

LEMMA 3.5: *If* $f : E(\Sigma_1, I_1) \to E(\Sigma_2, I_2)$ *is an elementary homomorphism and* $T \in \text{Reg}(\Sigma_1, I_1)$ *then* $f(T) \in \text{Reg}(\Sigma_2, I_2)$.

Proof: By Lemma 2.7 lin $f(T) = f(\text{lin } T)$ but regular languages are closed under homomorphism. $\square$

At the end let us observe that if we have any parallel device with a finite number of global configurations recognizing a trace language T then it can

be simulated sequentially by a finite state acceptor of lin T. Therefore trace languages outside $\text{Reg}(\Sigma, I)$ could not be recognized by such devices.

According to Eilenberg [9] regular trace languages should be called recognizable and, in fact, this name is usually used in literature, but in our paper recognizable, while applied to traces, will always mean "recognizable by a parallel device".

4. FINITE ASYNCHRONOUS AUTOMATA

A finite asynchronous automaton ASYN with n processes is a tuple $\mathbb{A} = (\mathbb{P}_1, \ldots, \mathbb{P}_n, \Delta, \mathbb{F})$. For every $i \in \bar{n}$ $\mathbb{P}_i = (\Sigma_i, S_i, s_i^0)$ is an i-th process, Σ_i is a finite and nonempty alphabet of $\mathbb{P}_i$, S_i is a finite and nonempty set of states, $s_i^0 \in S_i$ is an initial state of $\mathbb{P}_i$.

$S = \underset{i \in \bar{n}}{\times} S_i$ is the set of global states of $\mathbb{A}$ and $\mathbb{F} \subset S$ is the set of final states.

$\Sigma = \bigcup_{i \in \bar{n}} \Sigma_i$ is the alphabet of $\mathbb{A}$. In the following we denote by Proc the set $\{1, \ldots, n\}$ and we shall often identify every process $\mathbb{P}_i$ with its index $i \in \text{Proc}$. Moreover, $s^0 = (s_1^0, \ldots, s_n^0) \in S$ will denote the initial state of $\mathbb{A}$. The set $\text{Dom}(a) = \{ i \in \text{Proc} : a \in \Sigma_i \}$ of processes synchronously executing an action $a \in \Sigma$ will be called the domain of a. Intuitively, every action a acts only on the domain Dom (a) during its execution and this execution can be interpreted as a "hand shaking" communication between the processes from Dom (a).

Formally this is described by the next-state functions from the set $\Delta = \{ \delta_a : a \in \Sigma \}$. For all $a \in \Sigma$

$$\delta_a : \underset{i \in \text{Dom}(a)}{\times} S_i \to \mathscr{P}\Big(\underset{i \in \text{Dom}(a)}{\times} S_i \Big)$$

$\mathbb{A}$ is deterministic if

$$\forall a \in \Sigma, \quad \forall s \in \underset{i \in \text{Dom}(a)}{\times} S_i, \qquad \text{card}(\delta_a(s)) \leqq 1.$$

We now define the transition relation between global states of $\mathbb{A}$:
let

$$(s_1', \ldots, s_n'), (s_1'', \ldots, s_n'') \in S, \qquad a \in \Sigma,$$

then

$$(s_1', \ldots, s_n') \overset{a}{\Rightarrow} (s_1'', \ldots, s_n'')$$

iff

(i) $s'_i = s''_i$ for $i \notin \mathrm{Dom}(a)$, and

(ii) $(s''_{i_1}, \ldots, s''_{i_k}) \in \delta_a(s'_{i_1}, \ldots, s'_{i_k})$, where

$$\{i_1, \ldots, i_k\} = \mathrm{Dom}(a).$$

We extend this relation to words in the standard way. Let $s', s'' \in S$. Then $s' \overset{u}{\Rightarrow} s''$ if either

(i) $u = \varepsilon$ and $s' = s''$, or

(ii) $u = a_1 \ldots a_m$, $\forall i \in \bar{m}$, $a_i \in \Sigma$ and there exists a sequence $s^1, \ldots, s^{m+1} \in S$ such that

$$s' = s^1, \quad s'' = s^{m+1}, \quad \forall j \in \bar{m}, \qquad s^j \overset{a_j}{\Rightarrow} s^{j+1}.$$

We define the language recognized by $\mathbb{A}$ as

$$L(\mathbb{A}) = \{u \in \Sigma^* : \exists s \in \mathbb{F},\ s^0 \overset{u}{\Rightarrow} s\}.$$

Now we shall show how $\mathbb{A}$ can recognize traces. It is clear that if actions a, b operate on disjoint sets of processes then they may be executed concurrently, thus

$$I_{\mathbb{A}} = \{(a, b) \in \Sigma \times \Sigma : \mathrm{Dom}(a) \cap \mathrm{Dom}(b) = \varnothing\}$$

is the independency relation for $\mathbb{A}$.

LEMMA 4.1: *If* $(a, b) \in I_{\mathbb{A}}$, $s', s'' \in S$ *then*

$$s' \overset{ab}{\Rightarrow} s'' \quad \Leftrightarrow \quad s' \overset{ba}{\Rightarrow} s''$$

Let $u, v \in \Sigma^*$. *Then*

$$u \sim_{I_{\mathbb{A}}} v \quad \Rightarrow \quad (s' \overset{u}{\Rightarrow} s'' \Leftrightarrow s' \overset{v}{\Rightarrow} s'').$$

Proof: Obvious. □

We define the trace language recognized by $\mathbb{A}$ as

$$T(\mathbb{A}) = \{t \in E(\Sigma, I_{\mathbb{A}}) : \forall u \in t,\ \exists s \in \mathbb{F},\ s^0 \overset{u}{\Rightarrow} s\}.$$

From Lemma 4.1 it follows that

$$T(\mathbb{A}) = \{t \in E(\Sigma, I_{\mathbb{A}}) : \exists u \in t,\ \exists s \in \mathbb{F},\ s^0 \overset{u}{\Rightarrow} s\}.$$

Example 4.2: Let $\mathbb{A} = (\mathbb{P}_1, \mathbb{P}_2, \Delta, \mathbb{F})$, where

$$\mathbb{P}_i = (\Sigma_i, \{s_i^0, s_i^1, s_i^2\}, s_i^0), \qquad i = 1, 2,$$

$$\Sigma_1 = \{a, c\}, \qquad \Sigma_2 = \{b, c\}, \qquad \mathbb{F} = \{(s_1^0, s_2^0)\}.$$

The next-state functions are defined as follows

$$\delta_a(s_1^0) = s_1^1, \qquad \delta_a(s_1^1) = s_1^2, \qquad \delta_b(s_2^0) = s_2^1,$$

$$\delta_b(s_2^1) = s_2^2, \qquad \delta_c(s_1^1, s_2^1) = (s_1^0, s_2^0), \qquad \delta_c(s_1^2, s_2^2) = (s_1^0, s_2^0).$$

We assume that for other possible arguments δ_a, δ_b, δ_c produce empty sets. This automaton is deterministic and it is easy to establish that

$$T(\mathbb{A}) = [((ab \cup a^2 b^2) c)^*]_{I_{\mathbb{A}}}, \quad I_{\mathbb{A}} = \{(a, b), (b, a)\}. \quad \square$$

Every $\mathbb{A} \in \mathrm{ASYN}$ has a nice graphical representation as a labelled Petri net (*see* [14]). The set of all places of the net is equal to the disjoint union of S_i, $i \in \mathrm{Proc}$. For a given action a and every pair

$$s' = (s'_{i_1}, \ldots, s'_{i_k}), \qquad s'' = (s''_{i_1}, \ldots, s''_{i_k}) \in \underset{i \in \mathrm{Dom}(a)}{\times} S_i$$

such that $s'' \in \delta_a(s')$ we create a transition labelled by a, with the input places $\{s'_{i_1}, \ldots, s'_{i_k}\}$ and the output places $\{s''_{i_1}, \ldots, s''_{i_k}\}$. As the initial marking we take $\{s_1^0, \ldots, s_n^0\}$. Figure 1 presents the net representation of the ASYN automaton from Example 4.2.

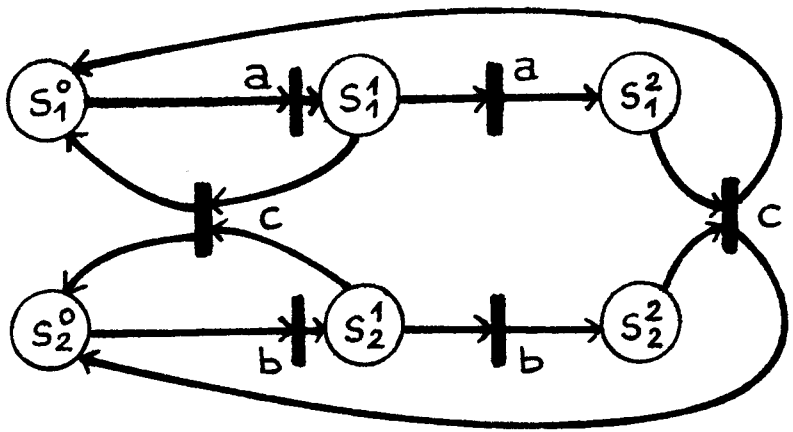

Figure 1.

Petri nets play here the same role as transition graphs for finite state acceptors. They are useful as pictures but rather clumsy in formal constructions.

Let

$$D_{\mathbb{A}} = \Sigma \times \Sigma - I_{\mathbb{A}} = \{ (a, b) \in \Sigma \times \Sigma : \mathrm{Dom}(a) \cap \mathrm{Dom}(b) \neq \varnothing \}$$

be the dependency relation of an $\mathbb{A} \in \mathrm{ASYN}$.

We say that $\mathbb{A}$ is in the normal form if

$$\forall i \in \mathrm{Proc}, \quad \Sigma_i \in \mathrm{Cliques}(D_{\mathbb{A}}) \quad \text{and} \quad \forall i, j \in \mathrm{Proc},$$
$$i \neq j \Rightarrow \Sigma_i \neq \Sigma_j.$$

PROPOSITION 4.3: *For every* $\mathbb{A} \in \mathrm{ASYN}$ *there exists* $\tilde{\mathbb{A}} \in \mathrm{ASYN}$ *in the normal form such that* $I_{\mathbb{A}} = I_{\tilde{\mathbb{A}}}$ *and* $T(\mathbb{A}) = T(\tilde{\mathbb{A}})$.

Proof: Let Cliques $(D_{\mathbb{A}}) = \{ \tilde{\Sigma}_1, \ldots, \tilde{\Sigma}_k \}$. Then $\forall i \in \mathrm{Proc}$, $\exists j \in \bar{k}$, $\Sigma_i \subset \tilde{\Sigma}_j$. For all $j \in \bar{k}$ we create a new process $\tilde{\mathbb{P}}_j = (\tilde{\Sigma}_j, \tilde{S}_j, \tilde{s}_j^0)$ with the following set of states $\tilde{S}_j = S_{j_1} \times \ldots \times S_{j_m}$ and the initial state

$$\tilde{s}_j^0 = (s_{j_1}^0, \ldots, s_{j_m}^0),$$

where

$$\{ j_1, \ldots, j_m \} = \{ i \in \mathrm{Proc} : \Sigma_i \subset \tilde{\Sigma}_j \}.$$

$\tilde{\delta}_a$ changes these components of $\tilde{S}_j$ that would be changed by δ_a during the execution of a in $\mathbb{A}$, while the rest remains unaltered. According to this

$(\tilde{s}_1, \ldots, \tilde{s}_k) \in \tilde{\mathbb{F}}$ if there exists $s \in \mathbb{F}$ such that every $\tilde{s}_i$, $i \in \bar{k}$, is a projection of s onto adequate coordinates. We leave details to the reader. □

Now we are ready to formulate the following two main theorems.

THEOREM 4.4: *For every finite asynchronous automaton* $\mathbb{A} \in \mathrm{ASYN}$ *the trace language* $T(\mathbb{A})$ *is regular,* $T(\mathbb{A}) \in \mathrm{Reg}(\Sigma, I_{\mathbb{A}})$.

Proof: It suffices to prove that $L(\mathbb{A})$ is regular. Let $\mathbb{A}$ be as in the preceding definition. We build the finite state acceptor $B = (\Sigma, S, s^0, \delta, \mathbb{F})$, where the next-state function is defined as follows

$$\delta : \quad S \times \Sigma \to \mathscr{P}(S), \qquad \forall s \in S, \qquad \forall a \in \Sigma,$$

$$\delta(s, a) = \{ s' \in S : s \overset{a}{\Rightarrow} s' \}.$$

clearly $L(\mathbb{A}) = L(B)$. □

THEOREM 4.5: *For every* $T \in \mathrm{Reg}(\Sigma, I)$ *there exists a deterministic finite asynchronous automaton* $\mathbb{A}$ *such that* $I_{\mathbb{A}} = I$ *and* $T(\mathbb{A}) = T$. □

Let us consider what really Theorem 4.5 states. It is clear that every regular trace language T can be implemented provided that we neglect concurrency. In this case we may build the minimal finite state acceptor A of the language lin T. In a way, A recognizes T because it recognizes all possible sequential executions of traces from T and independent actions can be executed in any order but only sequentially. This contrasts sharply with the behaviour of the finite asynchronous atomata. As in Petri nets, they have real ability of simultaneous execution of the independent actions. Thus Theorem 4.5 states that every $T \in \mathrm{Reg}(\Sigma, I)$ can be implemented by a finite state system, which is trivial, entirely preserving concurrency between independent actions, which is not so trivial. A similar problem was previously examined by Tarlecki [17]. The remainder of this section is devoted to the construction of an ASYN automaton recognizing a given regular trace language T. All proofs are shifted to Appendix at the end of the section.

Let Cliques $(D) = \{ \Sigma_1, \ldots, \Sigma_n \}$. We shall build an ASYN automaton $\mathbb{A}$ in the normal form, thus each Σ_i will stand for the alphabet of a process $\mathbb{P}_i$. We set $\mathrm{Proc} = \{ 1, \ldots, n \}$. As previously, for

$$a \in \Sigma, \quad x \in O(\Sigma), \qquad \mathrm{Dom}(a) = \{ i \in \mathrm{Proc} : a \in \Sigma_i \};$$
$$\mathrm{Dom}(x) = \mathrm{Dom}(\mathrm{name}(x)), \quad \text{for any } t \in E(\Sigma, I);$$
$$\mathrm{Dom}(t) = \{ i \in \mathrm{Proc} : \exists a \in \Sigma, \ \#_a t > 0, \ i \in \mathrm{Dom}(a) \}.$$

Let $x \in O(t)$. Consider the set $\mathrm{Pref}_x(t) = \{ y \in O(t) : y \leqq_t x \}$. It is obvious that this set is initial in $O(t)$ and therefore it determines a prefix of t which we shall denote by $P_x(t)$.

Let $t \in E(\Sigma, I)$, $i \in \mathrm{Proc}$. Then $\mathrm{last}_i^i(t)$ will be the last action occurrence executed in t by the process $i \in \mathrm{Proc}$, i. e. the maximal element of the set $\{ y \in O(t) : i \in \mathrm{Dom}(y) \}$.

If $x = \mathrm{last}_i^i(t)$ then we shall write $P_i(t)$ and $\mathrm{Pref}_i(t)$ to denote the prefix $P_x(t)$ and its set of actions $\mathrm{Pref}_x(t)$. Finally, for $\alpha \subset \mathrm{Proc}$, $P_\alpha(t)$ will be a prefix of t determined by the initial subset $\mathrm{Pref}_\alpha(t) = \bigcup_{i \in \alpha} \mathrm{Pref}_i(t)$ of $O(t)$. The following characterizarion of $\mathrm{Pref}_\alpha(t)$ will be sometimes useful.

Fact 4.6: Let $t \in E(\Sigma, I)$, $\alpha \subset \mathrm{Proc}$. Then

$$\mathrm{Pref}_\alpha(t) = \{ y \in O(t) : \exists x \in O(t),\ y \leqq_t x \wedge \alpha \cap \mathrm{Dom}(x) \neq \varnothing \}. \quad \square$$

Let $i, j \in \mathrm{Proc}$. Then $\mathrm{last}_j^i(t) = \mathrm{last}_j^j(P_i(t))$, i. e.

$$\mathrm{last}_j^i(t) = \max \{ y \in O(t) : y \leqq \mathrm{last}_i^i(t) \wedge j \in \mathrm{Dom}(y) \}$$
$$= \max \{ y \in O(t) : j \in \mathrm{Dom}(y) \wedge \exists x \in O(t),\ y \leqq_t x \wedge i \in \mathrm{Dom}(x) \}.$$

Intuitively, $\mathrm{last}_j^i(t)$ is the last action occurrence executed in t by the process j, $j \in \mathrm{Dom}(\mathrm{last}_j^i(t))$, and which can be "observed" by the process i, $\mathrm{last}_j^i(t) \leqq \mathrm{last}_i^i(t)$.

We set $\mathrm{LAST}(t) := \{ \mathrm{last}_j^i(t) : i, j \in \mathrm{Proc} \}$.

Note that the value $\mathrm{last}_j^i(t)$ may be sometimes undefined, e. g. $\mathrm{last}_j^i(\varepsilon)$ is undefined for all $i, j \in \mathrm{Proc}$.

Example 4.7: Let

$$\Sigma = \{ a, b, c, d, e, f \}, \qquad \mathrm{Cliques}(D) = \{ \Sigma_1, \Sigma_2, \Sigma_3 \},$$
$$\Sigma_1 = \{ a, b, d \}, \qquad \Sigma_2 = \{ a, c, e \}, \qquad \Sigma_3 = \{ f, e \}$$

Figure 2 presents a trace t over this alphabet and its prefix $P_1(t)$.
We have

$$\mathrm{last}_1^1(t) = b^2, \qquad \mathrm{last}_2^2(t) = c^2, \qquad \mathrm{last}_3^3(t) = f^4,$$
$$\mathrm{last}_2^1(t) = a^2, \qquad \mathrm{last}_3^1(t) = e^1,$$
$$\mathrm{last}_3^2(t) = \mathrm{last}_2^3(t) = e^2, \qquad \mathrm{last}_1^2(t) = \mathrm{last}_1^3(t) = a^2. \quad \square$$

In the sequel we shall frequently use equalities of the form $\mathrm{last}_j^i(t_1) = \mathrm{last}_l^k(t_2)$, where t_1, t_2 are prefixes of a trace t. This will mean that

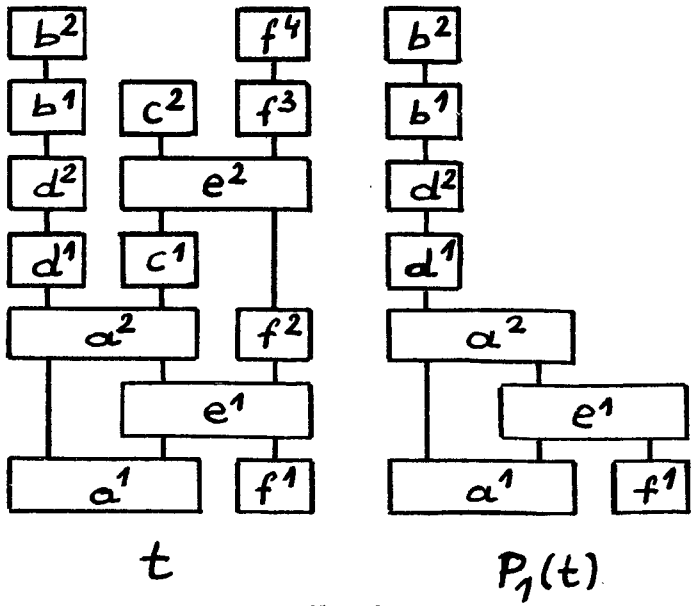

Figure 2.

$\mathrm{last}_j^i(t_1)$ is well-defined iff $\mathrm{last}_l^k(t_2)$ is well-defined and if that is the case then they denote the same elements of $O(t)$. Now we shall present two elementary properties of $P_\gamma(t)$.

Fact 4.8: (i) Let $t_1 \in E(\Sigma, I)$, $a \in \Sigma$, $t_2 = t_1 a$, $\alpha = \mathrm{Dom}(a)$, $m = \#_a t_2$, $\gamma \subset \mathrm{Proc}$. Then

$$\mathrm{Pref}_\gamma(t_2) = \begin{cases} \mathrm{Pref}_\gamma(t_1) & \text{if } \gamma \cap \alpha = \varnothing \\ \mathrm{Pref}_{\alpha \cup \gamma}(t_1) \cup \{ a^m \} & \text{if } \gamma \cap \alpha \neq \varnothing \end{cases}$$

$$P_\gamma(t_2) = \begin{cases} P_\gamma(t_1) & \text{if } \gamma \cap \alpha = \varnothing \\ P_{\alpha \cup \gamma}(t_1).a & \text{if } \gamma \cap \alpha \neq \varnothing \end{cases}$$

(ii) If $i \in \gamma \subset \mathrm{Proc}$, $t \in E(\Sigma, I)$ then

$$\mathrm{Pref}_i(P_\gamma(t)) = \mathrm{Pref}_i(t) \qquad \text{and} \qquad P_i(P_\gamma(t)) = P_i(t).$$

Proof: Immediate by Fact 2.5 and the definition of $P_\gamma(t)$. $\square$

We set $\mathrm{LAB} = \Sigma \times \mathrm{Proc}$ to be a set of labels. We shall present an algorithm that for every trace t constructs a labelling of t, i. e. a mapping

$$\mathrm{label}_t : O(t) \to \mathrm{LAB}.$$

ALGORITHM 4.9: (i) For the first occurrence a^1 of any action a in $O(t)$ we set $\text{label}_t(a^1)=(a, 1)$.

(ii) Suppose that the successive occurrences $a^1, \ldots, a^{k-1}$ of the action a have already been labelled. Let $x=a^k \in O(t)$ and let

$$G_x=\text{LAST}(P_x(t)) \cap \{a^1, \ldots, a^k\}.$$

Let $m \in N$ be the least positive integer such that

$$\forall y \in G_x - \{x\}, \quad \text{label}_t(y) \neq (a, m).$$

Then we set

$$\text{label}_t(x)=(a, m). \quad \square$$

The next lemma summarizes properties of label_t. Recall that name: $O(\Sigma) \to \Sigma$ is a projection of $O(\Sigma)$ onto Σ.

PROPOSITION 4.10: *The function* label_t *constructed by Algorithm* 4.9 *satisfies the following conditions.*

(L 1) $\forall x \in O(t)$, $\text{name}(x)=a \Rightarrow \exists i \in \text{Proc}$, $\text{label}_t(x)=(a, i)$

(L 2) *if* p *is a prefix of* t *then* $\text{label}_p=\text{label}_t \mid O(p)$

(L 3) *the mapping* label_t *is injective on the set* $\text{LAST}(t)$. $\square$

Let $\alpha \subset \text{Proc}$. We set

$$\text{Suff}_\alpha(t)=\{x \in O(t) : \text{Dom}(x) \subset \alpha \wedge \forall y \in O(t), \; x \leqq_t y \Rightarrow \text{Dom}(y) \subset \alpha\}$$

It is obvious that $\text{Suff}_\alpha(t)$ is final in t and it determines a suffix of t denoted by $S_\alpha(t)$.

Note that $S_\alpha(t)$ is the greatest suffix of t such that $\text{Dom}(S_\alpha(t)) \subset \alpha$.

In the sequel, for $\alpha \subset \text{Proc}$, $\bar{\alpha}$ will denote the complement of α, i. e. $\bar{\alpha}=\text{Proc}-\alpha$. Observe that Fact 4.6 and the definition of $\text{Suff}_\alpha(t)$ imply that $\forall \alpha \subset \text{Proc}$,

$$\text{Pref}_\alpha(t) \cap \text{Suff}_{\bar{\alpha}}(t)=\varnothing \qquad \text{and} \qquad \text{Pref}_\alpha(t) \cup \text{Suff}_{\bar{\alpha}}(t)=O(t).$$

This yields

Fact 4.11: $\forall \alpha \subset \text{Proc}$, $\forall t \in E(\Sigma, I)$, $P_\alpha(t).S_{\bar{\alpha}}(t)=t$. $\square$

We shall now define equivalence relations over $E(\Sigma, I)$. Their properties are crucial in our construction.

DEFINITION 4.12: Let $t, r \in E(\Sigma, I)$. Then $t \approx_E r$ if the following conditions hold

(i) The mapping C : LAST $(t) \to$ LAST (r) such that

$$\forall i, j \in \text{Proc}, \quad C(\text{last}_j^i(t))=\text{last}_j^i(r)$$

is an isomorphism of the partial orders $\leqq_t \mid \text{LAST}(t)$ and $\leqq_r \mid \text{LAST}(r)$. This isomorphism will be called canonical. Note that the above condition implies that $\text{last}_j^i(t)$ is well-defined iff $\text{last}_j^i(r)$ is well-defined.

(ii) C preserves the labellings, i. e.

$$\forall i, j \in \text{Proc}, \; \text{label}_t(\text{last}_j^i(t))=\text{label}_r(\text{last}_j^i(r)). \quad \square$$

Recall that $\sim_T$ denotes the syntactical congruence of T.

DEFINITION 4.13: Let $T \subset E(\Sigma, I)$. Then

$$t \approx_T r \quad \text{if } \forall \alpha \subset \text{Proc}, \qquad S_\alpha(t) \sim_T S_\alpha(r). \quad \square$$

DEFINITION 4.14: Let $t, r \in E(\Sigma, I)$. Then $t \approx r$ if

$$t \approx_E r \quad \text{and} \quad t \approx_T r, \qquad \text{i. e.} \quad \approx = \approx_E \cap \approx_T. \quad \square$$

It is obvious that $\approx_E$ is always of finite index, whereas $\approx_T$, and consequently $\approx$, are of finite index iff T is regular.

Henceforth $\langle t \rangle$ will stand for an equivalence class of t under $\approx$.

We now give two theorems that constitute a key to our construction.

THEOREM 4.15: *Let* $t, r \in E(\Sigma, I)$ *and* $\alpha, \beta \subset \text{Proc}$. *If* $P_\alpha(t) \approx P_\alpha(r)$ *and* $P_\beta(t) \approx P_\beta(r)$ *then* $P_{\alpha \cup \beta}(t) \approx P_{\alpha \cup \beta}(r)$. $\square$

THEOREM 4.16: *Let* $t_1, r_1 \in E(\Sigma, I)$, $a \in \Sigma$, $t_2=t_1 a$, $r_2=r_1 a$ *and* $\forall i \in \text{Dom}(a)$, $P_i(t_1) \approx P_i(r_1)$. *Then* $\forall i \in \text{Dom}(a)$, $P_i(t_2) \approx P_i(r_2)$. $\square$

Now we are able to present the construction of an ASYN automaton recognizing a given regular trace language T. The equivalence classes of $\approx$ will serve as states of our automaton $\mathbb{A}$ and $\mathbb{A}$ will behave in such a way that, having a trace t executed, an i-th process $\mathbb{P}_i$ reaches the state $\langle P_i(t) \rangle$.

Formally, $\mathbb{A}=(\mathbb{P}_1, \ldots, \mathbb{P}_n, \Delta, \mathbb{F})$. For all $i \in \text{Proc}=\{1, \ldots, n\}$ we set $\mathbb{P}_i=(\Sigma_i, S_i, s_i^0)$, $S_i=\{\langle P_i(t) \rangle : t \in E(\Sigma, I)\}$, $s_i^0=\langle \varepsilon \rangle$.

Let $a \in \Sigma$, $\text{Dom}(a)=\{i_1, \ldots, i_k\}$, $t_2=t_1 a$. Then we set

$$\delta_a(\langle P_{i_1}(t_1) \rangle, \ldots, \langle P_{i_k}(t_1) \rangle)=(\langle P_{i_1}(t_2) \rangle, \ldots, \langle P_{i_k}(t_2) \rangle)$$

Theorem 4.16 ensures that this definition is sound. Moreover, Fact 4.8 (i) implies that for $i \notin \mathrm{Dom}(a)$ $P_i(t_2) = P_i(t_1)$. Using this fact and Theorem 4.16, one can verify by trivial induction on the number of action occurrences in t that the following condition holds

PROPOSITION 4.17: $\forall t \in E(\Sigma, I)$,

$$(\langle \varepsilon \rangle, \ldots, \langle \varepsilon \rangle) \stackrel{t}{\Rightarrow} (\langle P_1(t) \rangle, \ldots, \langle P_n(t) \rangle) \quad \textit{in } \mathbb{A}. \quad \square$$

In order to accomplish the construction we have to define the set of final states:

$$\mathbb{F} = \{(\langle P_1(t) \rangle, \ldots, \langle P_n(t) \rangle) : t \in T\}.$$

By definition, for all $r \in E(\Sigma, I)$, if $r \approx t$ then

$$r = S_{\mathrm{Proc}}(r) \sim_T S_{\mathrm{Proc}}(t) = t,$$

hence $r \in T$ iff $t \in T$. This fact and Proposition 4.17 prove that $T(\mathbb{A}) = T$. Finally observe that the construction presented here can be carried out effectively. For any two traces t_1, t_2 we can establish effectively if $t_1 \approx t_2$. Thus by simple inspection, starting with the empty trace ε, we can find an oriented graph $G = (V, E)$ such that every vertex $v \in V$ is labelled by a trace $t \in E(\Sigma, I)$ and every edge $e \in E$ is labelled by an action $a \in \Sigma$ and fulfilling the following conditions

(1) if two different vertices v_1, v_2 are labelled by t_1 *and* t_2 respectively then $t_1 \not\approx t_2$;

(2) $\forall a \in \Sigma$, $\forall v \in V$ there is an edge outgoing from v and labelled by a;

(3) an edge labelled by $a \in \Sigma$ joins vertices labelled by t_1 and t_2 iff $t_2 \approx t_1 a$.

It is obvious that the graph G is isomorphic with the transition graph of $\mathbb{A}$. G enables us to define δ_a effectively for all $a \in \Sigma$, namely for every two vertices v_1, v_2 labelled by t_1 and t_2 and connected by an edge labelled by $a \in \Sigma$ we set

$$\delta_a(\langle P_{i_1}(t_1) \rangle, \ldots, \langle P_{i_k}(t_1) \rangle) = (\langle P_{i_1}(t_2) \rangle, \ldots, \langle P_{i_k}(t_2) \rangle)$$

where $\{i_1, \ldots, i_k\} = \mathrm{Dom}(a)$.

APPENDIX:

The following lemma gives a list of elementary properties of $\mathrm{last}_j^i(t)$.

LEMMA 4.18: *Let* $t_1 \in E(\Sigma, I)$, $\alpha \subset \mathrm{Proc}$. *Then*

(i) *If* $i \in \alpha$ *then*

$$\forall \beta \subset \mathrm{Proc}, \quad \mathrm{last}_j^i(P_\alpha(t_1)) = \mathrm{last}_j^i(P_{\alpha \cup \beta}(t_1)) = \mathrm{last}_j^i(t_1)$$

(ii) $\forall j \in \mathrm{Proc}$, $\mathrm{last}_j^j(P_\alpha(t_1)) = \max\{\mathrm{last}_j^i(t_1) : i \in \alpha\}$.

Let moreover $t_2 = t_1 a$, $a \in \Sigma$, $\mathrm{Dom}(a) = \alpha$, $m = \#_a t_2$. *Then*

(iii) $$\mathrm{last}_i^i(t_2) = \begin{cases} \mathrm{last}_i^i(t_1) & \textit{if } i \notin \alpha \\ a^m & \textit{if } i \in \alpha \end{cases}$$

(iv) $$\mathrm{last}_j^i(t_2) = \begin{cases} \mathrm{last}_j^i(t_1) & \textit{if } i \notin \alpha, \quad j \in \mathrm{Proc} \\ a^m & \textit{if } i, j \in \alpha \\ \max\{\mathrm{last}_j^k(t_1) : k \in \alpha\} & \textit{if } i \in \alpha, j \notin \alpha \end{cases}$$

Proof: (i) By Fact 4.8 (ii) $i \in \gamma \subset \mathrm{Proc}$ implies

$$\mathrm{last}_j^i(P_\gamma(t_1)) = \mathrm{last}_j^j(P_i(P_\gamma(t_1))) = \mathrm{last}_j^j(P_i(t_1)) = \mathrm{last}_j^i(t_1).$$

Now it suffices to take $\gamma = \alpha$ and $\gamma = \alpha \cup \beta$.

(ii) Since $\mathrm{Pref}_\alpha(t_1) = \bigcup_{i \in \alpha} \mathrm{Pref}_i(t_1)$, we have

$$\mathrm{last}_j^j(P_\alpha(t_1)) = \max\{\mathrm{last}_j^j(P_i(t_1)) : i \in \alpha\} = \max\{\mathrm{last}_j^i(t_1) : i \in \alpha\}$$

(iii) This is just another formulation of the property described by Fact 4.8 (i).

(iv) In case $i \in \alpha$, by Fact 4.8 (i), we get

$$\mathrm{last}_j^i(t_1 a) = \mathrm{last}_j^j(P_i(t_1 a)) = \mathrm{last}_j^j(P_\alpha(t_1) a)$$

and now it suffices to apply (iii) and (ii) to obtain a^m for $j \in \alpha$ and

$$\mathrm{last}_j^j(P_\alpha(t_1)) = \max\{\mathrm{last}_j^k(t_1) : k \in \alpha\} \qquad \text{for} \quad j \notin \alpha.$$

In case $i \notin \alpha$ we argue analogously. $\square$

LEMMA 4.19: *Let* $t \in E(\Sigma, I)$ *and* $i \in \mathrm{Proc}$. *Then the mapping name:* $O(\Sigma) \to \Sigma$ *is injective on the set*

$$\{\mathrm{last}_j^i(t) : j \in \mathrm{Proc}\} \subset \mathrm{LAST}(t).$$

Proof: Let $\mathrm{last}_j^i(t) = x$, $\mathrm{last}_k^i(t) = z$ and $\mathrm{name}(x) = \mathrm{name}(z) = a$. Then $j, k \in \mathrm{Dom}(a)$. But $j \in \mathrm{Dom}(z)$ implies $z \leqq_t \mathrm{last}_j^i(t)$, whereas $k \in \mathrm{Dom}(x)$ implies $x \leqq_t \mathrm{last}_k^i(t)$. Thus $x = z$. $\square$

LEMMA 4.20: *Let t be a trace and t_1 its prefix. If $\text{last}_j^i(t) \in O(t_1)$ for some $i, j \in \text{Proc}$ then there exists $k \in \text{Proc}$ such that $\text{last}_j^k(t_1) = \text{last}_j^i(t)$. In particular,* $\text{LAST}(t) \cap O(t_1) \subset \text{LAST}(t_1)$.

Proof: Let $t = t_1 t_2$. If $\# t_2 = 1$ then the thesis follows immediately from Lemma 4.18 (iv). In general, it suffices to apply an elementary induction on $\# t_2$. □

Now we are able to prove the properties (L1)–(L3) of label_t.

Proof of Proposition 4.10: First note that if p is a prefix of t and $x \in O(p)$ then $P_x(t) = P_x(p)$, which proves L2.

In order to prove L1 it suffices to show that the number m chosen in the step (ii) of Algorithm 4.9 belongs to Proc. If $i \in \text{Dom}(a)$, $x = a^k$, then either

$$\text{last}_j^i(P_x(t)) = x \qquad \text{for} \quad j \in \text{Dom}(a)$$

or

$$\text{name}(\text{last}_j^i(P_x(t))) \neq a \qquad \text{for} \quad j \in \text{Proc} - \text{Dom}(a).$$

Thus

$$G_x - \{x\} = \text{LAST}(P_x(t)) \cap \{a^1, \ldots, a^{k-1}\}$$
$$= \{\text{last}_j^i(P_x(t)) : i \in \text{Proc} - \text{Dom}(a), j \in \text{Proc}\}$$
$$\cap \{a^1, \ldots, a^{k-1}\} = \bigcup_{i \in \text{Proc} - \text{Dom}(a)} \{\text{last}_j^i(P_x(t)) : j \in \text{Proc}\}$$
$$\cap \{a^1, \ldots, a^{k-1}\}.$$

But from Lemma 4.19 it follows that

$$\text{card}(\{\text{last}_j^i(P_x(t)) : j \in \text{Proc}\} \cap \{a^1, \ldots, a^{k-1}\}) \leqq 1$$

for all $i \in \text{Proc}$. Therefore

$$\text{card}(G_x - \{x\}) \leqq \text{card}(\text{Proc} - \text{Dom}(a)) < \text{card}(\text{Proc})$$

and m really belongs to Proc.

We prove L3 by a contradiction. Let us suppose that for a trace t there exist x, $y \in \text{LAST}(t)$, $x \neq y$, such that $\text{label}_t(x) = \text{label}_t(y)$. Then by L1 $\text{name}(x) = \text{name}(y)$. We may assume that $y \leqq_t x$. Then x, $y \in \text{Pref}_x(t)$ and by Lemma 4.20 x, $y \in \text{LAST}(P_x(t))$. But then Algorithm 4.9 would have ensured that $\text{label}_t(x) \neq \text{label}_t(y)$. □

Henceforth $\text{Max}(t) = \{x \in O(t) : \neg \exists z \in O(t),\ x <_t z\}$ will stand for the set of maximal elements of a trace t.

In the next two lemmas we complete a list of useful properties of $\text{last}_j^i(t)$.

LEMMA 4.21: *Let $\alpha \subset \text{Proc}$, $t \in E(\Sigma, I)$. Then*

(i) *for every $j \in \text{Proc}$ there exists $i \in \alpha$ such that*

$$\text{last}_j^i(P_\alpha(t)) = \text{last}_j^i(t) = \text{last}_j^i(P_\alpha(t))$$

(ii) *if $x \in \text{Max}(P_\alpha(t))$ then there exists $i \in \alpha$ such that*

$$x = \text{last}_i^i(t) = \text{last}_i^i(P_\alpha(t)).$$

Proof: (i) To obtain (i) it suffices to take $i \in \alpha$ for which the maximum in Lemma 4.18 (ii) is reached and since $i \in \alpha$, again by Lemma 4.18 (i), $\text{last}_j^i(t) = \text{last}_j^i(P_\alpha(t))$.

(ii) This statement follows from Lemma 4.18 (i) and from the obvious inclusion $\text{Max}(P_\alpha(t)) \subset \{\text{last}_i^i(t) : i \in \alpha\}$. □

LEMMA 4.22: *Let $\gamma, \eta \subset \text{Proc}$, $t \in E(\Sigma, I)$. If $x \in \text{Max}(P_\gamma(P_\eta(t)))$ then there exist $i \in \eta$, $j \in \gamma$ such that*

$$x = \text{last}_j^j(P_\gamma(P_\eta(t))) = \text{last}_j^j(P_\eta(t)) = \text{last}_j^i(P_\eta(t)).$$

Proof: If $x \in \text{Max}(P_\gamma(P_\eta(t)))$ then by Lemma 4.21 (ii)

$$x = \text{last}_j^j(P_\gamma(P_\eta(t))) = \text{last}_j^j(P_\eta(t)) \quad \text{for some } j \in \gamma,$$

and again by Lemma 4.21 (i)

$$\text{last}_j^j(P_\eta(t)) = \text{last}_j^i(P_\eta(t)) \quad \text{for some } i \in \eta. \quad \square$$

The next two lemmas contain basic properties of $P_\alpha(t)$ and $S_\alpha(t)$.

LEMMA 4.23: *Let $t \in E(\Sigma, I)$, $\alpha, \beta, \gamma, \delta \subset \text{Proc}$. Then*

(i) $S_\alpha(S_\beta(t)) = S_{\alpha \cap \beta}(t)$;

(ii) *If $\gamma \subset \delta \subset \text{Proc}$ then*

$$P_\gamma(P_\delta(t)) = P_\gamma(t) \qquad \textit{and} \qquad P_\delta(P_\gamma(t)) = P_\gamma(t).$$

Later on the first condition will be used in the form

$$P_\alpha(P_{\alpha \cup \beta}(t)) = P_\alpha(t).$$

Proof: (i) follows immediately from the definition of $S_\alpha(t)$.

(ii) Since $\gamma \subset \delta$ by Fact 4.8 (ii) we get

$$\mathrm{Pref}_\gamma(P(t)) = \bigcup_{i\in\gamma} \mathrm{Pref}_i(P_\delta(t)) = \bigcup_{i\in\gamma} \mathrm{Pref}_i(t) = \mathrm{Pref}_\gamma(t), \text{ thus } P_\gamma(P_\delta(t)) = P_\gamma(t).$$

$P_\delta(P_\gamma(t))$ is a prefix of $P_\gamma(t)$ thus $\mathrm{Pref}_\delta(P_\gamma(t)) \subset \mathrm{Pref}_\gamma(t)$. On the other hand by Fact 4.8 (ii)

$$\mathrm{Pref}_\delta(P_\gamma(t)) = \bigcup_{i\in\delta} \mathrm{Pref}_i(P_\gamma(t)) \supset \bigcup_{i\in\gamma} \mathrm{Pref}_i(P_\gamma(t)) = \bigcup_{i\in\gamma} \mathrm{Pref}_i(t) = \mathrm{Pref}_\gamma(t),$$

which concludes the proof. □

LEMMA 4.24: *Let* $\alpha \subset \mathrm{Proc}$, $t_1, t_2 \in E(\Sigma, I)$. *Then*

$$P_\alpha(t_1 t_2) = P_{\alpha\cup\gamma}(t_1).P_\alpha(t_2)$$

and

$$S_\beta(t_1 t_2) = S_\delta(t_1).S_\beta(t_2),$$

where

$$\gamma = \mathrm{Dom}(P_\alpha(t_2)), \qquad \delta = \beta - \gamma, \qquad \alpha - \bar{\beta}.$$

Proof: Let $t_1 t_2$. We shall argue by induction on $\# t_2$. If $t_2 = c$, $c \in \Sigma$ then the formula for $P_\alpha(t_1 t_2)$ follows from Fact 4.8 (i) since $P_\alpha(c) = c$ if $\alpha \cap \mathrm{Dom}(c) \neq \varnothing$, and $P_\alpha(c) = \varepsilon$ otherwise.

Assume that the formula for $P_\alpha(t_1 t_2)$ is valid for all factorizations $t = t_1 t_2$ with $\# t_2 < m$.

Let $t = t_1 t_2$ and $\# t_2 = m > 1$. Then t_2 can be factorized in such a way that $t_2 = t_2' t_2''$ and $\# t_2' < m$, $\# t_2'' < m$. Let

$$\gamma_2 = \mathrm{Dom}(P_\alpha(t_2'')), \qquad \gamma_1 = \mathrm{Dom}(P_{\alpha\cup\gamma_2}(t_2')).$$

Then using three times the inductive hypothesis, for $(t_1 t_2')t_2''$, $t_1 t_2'$ and $t_2' t_2''$, we get

$$\begin{aligned} P_\alpha(t_1 t_2) = P_\alpha(t_1 t_2' t_2'') &= P_{\alpha\cup\gamma_2}(t_1 t_2').P_\alpha(t_2'') \\ &= P_{\alpha\cup\gamma_1\cup\gamma_2}(t_1).P_{\alpha\cup\gamma_2}(t_2').P_\alpha(t_2'') \\ &= P_{\alpha\cup\gamma_1\cup\gamma_2}(t_1).P_\alpha(t_2' t_2'') = P_{\alpha\cup\gamma_1\cup\gamma_2}(t_1).P_\alpha(t_2). \end{aligned}$$

But

$$\gamma_1 \cup \gamma_2 = \mathrm{Dom}(P_{\alpha\cup\gamma_2}(t_2')) \cup \mathrm{Dom}\, P_\alpha(t_2'')$$

$$= \mathrm{Dom}(P_{\alpha\cup\gamma_2}(t_2').P_\alpha(t_2'')) = \mathrm{Dom}(P_\alpha(t_2' t_2'')) = \mathrm{Dom}(P_\alpha(t_2)) = \gamma.$$

We have used above the obvious property

$$\mathrm{Dom}(t_1 t_2) = \mathrm{Dom}(t_1) \cup \mathrm{Dom}(t_2), \quad \text{for all } t_1, t_2 \in E(\Sigma, I).$$

Now we shall compute $S_\beta(t_1 t_2)$. First note that $\mathrm{Dom}(P_\alpha(t_2)) = \gamma$ and $\mathrm{Dom}(S_\delta(t_1)) \subset \delta = \beta - \gamma$, whence

$$\mathrm{Dom}(P_\alpha(t_2)) \cap \mathrm{Dom}(S_\delta(t_1)) = \varnothing \qquad \text{and} \qquad P_\alpha(t_2).S_\delta(t_1) = S_\delta(t_1).P_\alpha(t_2).$$

Therefore using the formula for $P_\alpha(t_1 t_2)$ and Fact 4.11 we obtain

$$\begin{aligned} P_\alpha(t_1 t_2).S_\delta(t_1).S_\beta(t_2) &= P_{\alpha\cup\gamma}(t_1).P_\alpha(t_2).S_\delta(t_1).S_\beta(t_2) \\ &= P_{\alpha\cup\gamma}(t_1).S_\delta(t_1).P_\alpha(t_2).S_\beta(t_2) = P_{\alpha\cup\gamma}(t_1).S_\delta(t_1).t_2 \\ &= P_{\alpha\cup\gamma}(t_1).S_{\bar{\alpha}\cap\bar{\gamma}}(t_1).t_2 = t_1 t_2. \end{aligned}$$

Since by Fact 4.11 $P_\alpha(t_1 t_2).S_\beta(t_1 t_2) = t_1 t_2$, we have

$$P_\alpha(t_1 t_2).S_\beta(t_1 t_2) = P_\alpha(t_1 t_2).S_\delta(t_1).S_\beta(t_2),$$

which implies by the cancellation property $S_\beta(t_1 t_2) = S_\delta(t_1).S_\beta(t_2)$. □

The next lemma describes a factorization of $P_{\alpha\cup\beta}(t)$.

LEMMA 4.25: *Let* $\alpha, \beta \subset \mathrm{Proc}$, $t \in E(\Sigma, I)$. *Then there exist* t_0, t', t'', $\gamma = \mathrm{Dom}(t')$, $\delta = \mathrm{Dom}(t'')$ *such that*

(i) $\gamma \cap \delta = \varnothing$;

(ii) $P_{\alpha\cup\beta}(t) = t_0 t' t'' = t_0 t'' t'$;

(iii) $P_\alpha(t) = t_0 t'$, $P_\beta(t) = t_0 t''$;

(iv) $\begin{cases} t_0 = P_{\bar{\gamma}}(P_\alpha(t)) = P_{\bar{\delta}}(P_\beta(t)), \\ t' = S_\gamma(P_\alpha(t)),\ t'' = S_\delta(P_\beta(t)) \end{cases}$

(v) $O(P_\alpha(t)) \cap O(P_\beta(t)) = O(t_0)$.

Proof: Using Lemma 4.23 (ii) and Fact 4.11 we obtain the following factorization

$$P_{\alpha\cup\beta}(t) = P_\alpha(P_{\alpha\cup\beta}(t)).S_{\bar{\alpha}}(P_{\alpha\cup\beta}(t)) = P_\alpha(t).S_{\bar{\alpha}}(P_{\alpha\cup\beta}(t))$$

and similarly

$$P_{\alpha\cup\beta}(t) = P_\beta(t).S_{\bar{\beta}}(P_{\alpha\cup\beta}(t)).$$

Let

$$\left.\begin{aligned} t' = S_{\bar{\beta}}(P_{\alpha\cup\beta}(t)), \qquad t'' = S_{\bar{\alpha}}(P_{\alpha\cup\beta}(t)), \\ \gamma = \mathrm{Dom}\,(t'), \qquad \delta = \mathrm{Dom}\,(t''). \end{aligned}\right\} \tag{1}$$

Note that

$$\gamma \subset \bar{\beta} \quad \text{and} \quad \delta \subset \bar{\alpha}. \tag{2}$$

We shall show that $\gamma \cap \delta = \varnothing$. Suppose the contrary, $\gamma \cap \delta \neq \varnothing$. Then there exist

$$x \in \mathrm{Pref}_{\alpha\cup\beta}(t) - \mathrm{Pref}_{\alpha}(t) = (\mathrm{Pref}_{\alpha}(t) \cup \mathrm{Pref}_{\beta}(t)) - \mathrm{Pref}_{\alpha}(t) = \mathrm{Pref}_{\beta}(t) - \mathrm{Pref}_{\alpha}(t)$$

and

$$y \in \mathrm{Pref}_{\alpha\cup\beta}(t) - \mathrm{Pref}_{\beta}(t) = \mathrm{Pref}_{\alpha}(t) - \mathrm{Pref}_{\beta}(t)$$

such that $\varnothing \neq \mathrm{Dom}\,(x) \cap \mathrm{Dom}\,(y) \subset \gamma \cap \delta$. Thus either $x \leqq_t y \in \mathrm{Pref}_{\alpha}(t)$ or $y \leqq_t x \in \mathrm{Pref}_{\beta}(t)$, which implies $x \in \mathrm{Pref}_{\alpha}(t)$ or $y \in \mathrm{Pref}_{\beta}(t)$ and both these cases yield a contradiction, which concludes the proof of (i).

Using Lemma 4.24 we compute

$$S_{\delta}(P_{\alpha\cup\beta}(t)) = S_{\delta}(P_{\beta}(t).t') = S_{\delta-\xi}(P_{\beta}(t)).S_{\delta}(t'),$$

with $\xi = \mathrm{Dom}\,(P_{\bar{\delta}}(t'))$. But $\mathrm{Dom}\,(t') \cap \delta = \varnothing$ implies $S_{\delta}(t') = \varepsilon$ and $P_{\bar{\delta}}(t') = t'$, whence $\xi = \mathrm{Dom}\,(t') = \gamma$. Since by (i) $\delta \cap \gamma = \varnothing$, $\delta - \xi = \delta - \gamma = \delta$. Therefore

$$S_{\delta}(P_{\alpha\cup\beta}(t)) = S_{\delta}(P_{\beta}(t)). \tag{3}$$

On the other hand

$$S_{\delta}(P_{\alpha\cup\beta}(t)) = S_{\delta}(P_{\alpha}(t).t'') = S_{\delta-\xi}(P_{\alpha}(t)).S_{\delta}(t''),$$

with $\xi = \mathrm{Dom}\,(P_{\bar{\delta}}(t''))$. But $\mathrm{Dom}\,(t'') = \delta$ implies $S_{\delta}(t'') = t''$ and $P_{\bar{\delta}}(t'') = \varepsilon$, thus $\xi = \varnothing$. Therefore

$$S_{\delta}(P_{\alpha\cup\beta}(t)) = S_{\delta}(P_{\alpha}(t)).t''.$$

We claim that $S_{\delta}(P_{\alpha}(t)) = \varepsilon$. Indeed, we have the factorization

$$P_{\alpha}(t) = P_{\bar{\delta}}(P_{\alpha}(t)).S_{\delta}(P_{\alpha}(t)),$$

but by (2) $\bar{\delta} \supset \alpha$, thus by Lemma 4.23 (ii) $P_{\bar{\delta}}(P_{\alpha}(t)) = P_{\alpha}(t)$, whence $P_{\alpha}(t) = P_{\alpha}(t).S_{\delta}(P_{\alpha}(t))$ and by Proposition 2.2 $S_{\delta}(P_{\alpha}(t)) = \varepsilon$. Thus finally

$S_{\delta}(P_{\alpha\cup\beta}(t)) = t''$. The last formula combined with (1) and (3) gives

$$t'' = S_{\delta}(P_{\alpha\cup\beta}(t)) = S_{\delta}(P_{\beta}(t)) = S_{\bar{\alpha}}(P_{\alpha\cup\beta}(t)). \tag{4}$$

In the same way we obtain

$$t' = S_{\gamma}(P_{\alpha\cup\beta}(t)) = S_{\gamma}(P_{\alpha}(t)) = S_{\bar{\beta}}(P_{\alpha\cup\beta}(t)).$$

Let $t'_0 = P_{\bar{\gamma}}(P_{\alpha}(t))$, $t''_0 = P_{\bar{\delta}}(P_{\beta}(t))$.

To complete the proof we must show (ii) and $t'_0 = t''_0$.

Using Fact 4.11, Lemma 4.23 (ii) and (4), we obtain

$$\begin{aligned} t'_0 t' t'' &= P_{\bar{\gamma}}(P_{\alpha}(t)).S_{\gamma}(P_{\alpha}(t)).S_{\bar{\alpha}}(P_{\alpha\cup\beta}(t)) \\ &= P_{\alpha}(t).S_{\bar{\alpha}}(P_{\alpha\cup\beta}(t)) = P_{\alpha}(P_{\alpha\cup\beta}(t)).S_{\bar{\alpha}}(P_{\alpha\cup\beta}(t)) = P_{\alpha\cup\beta}(t)). \end{aligned}$$

Analogously we get $t''_0 t'' t' = P_{\alpha\cup\beta}(t)$.

But by (i) $\mathrm{Dom}\,(t') \cap \mathrm{Dom}\,(t'') = \varnothing$, which implies $t' t'' = t' t''$, thus $P_{\alpha\cup\beta}(t) = t'_0 t' t'' = t''_0 t'' t' = t''_0 t' t''$ and by the cancellation property $t'_0 = t''_0$.

Finally note that (v) is an immediate consequence of (i) and (iii). □

LEMMA 4.26: *Let* $t,\ r \in E(\Sigma, I)$, $P_{\alpha}(t) \approx_E P_{\alpha}(r)$, $P_{\beta}(t) \approx_E P_{\beta}(r)$ *for* $\alpha, \beta \subset \mathrm{Proc}$. *Moreover, let* $P_{\alpha\cup\beta}(t) = t_0 t' t''$, $P_{\alpha\cup\beta}(r) = r_0 r' r''$ *be the factorizations of* $P_{\alpha\cup\beta}(t)$ *and* $P_{\alpha\cup\beta}(r)$ *defined in Lemma* 4.25. *And finally let* C_{α}, C_{β} *be the canonical isomorphisms of* $\mathrm{LAST}(P_{\alpha}(t))$, $\mathrm{LAST}(P_{\alpha}(r))$ *and* $\mathrm{LAST}(P_{\beta}(t))$, - $\mathrm{LAST}(P_{\beta}(r))$. *Then*

(i) $\mathrm{Max}\,(t_0) \subset \mathrm{LAST}(P_{\alpha}(t)) \cap \mathrm{LAST}(P_{\beta}(t)) \cap \mathrm{LAST}(P_{\alpha\cup\beta}(t))$;

(ii) $\forall x \in \mathrm{Max}\,(t_0)$, $C_{\alpha}(x) = C_{\beta}(x) \in \mathrm{Max}\,(r_0)$;

(iii) $\mathrm{Dom}\,(t') = \mathrm{Dom}\,(r')$, $\mathrm{Dom}\,(t'') = \mathrm{Dom}\,(r'')$.

Proof: Let $\gamma = \mathrm{Dom}\,(t')$, $\delta = \mathrm{Dom}\,(t'')$. By Lemma 4.25 (iv)

$$t_0 = P_{\bar{\gamma}}(P_{\alpha}(t)) = P_{\bar{\delta}}(P_{\beta}(t))$$

and therefore by Lemma 4.22 if $x \in \mathrm{Max}\,(t_0)$ then

$$\exists i \in \alpha, \quad \exists j \in \bar{\gamma}, \qquad x = \mathrm{last}^j_j(P_{\alpha}(t)) = \mathrm{last}^i_j(P_{\alpha}(t)).$$

Now, since $i \in \alpha$, we can apply Lemma 4.18 (i) to obtain $\mathrm{last}^i_j(P_{\alpha}(t)) = \mathrm{last}^i_j(P_{\alpha\cup\beta}(t))$. In the same way we obtain that

$$\exists k \in \beta, \quad \exists l \in \bar{\delta}, \qquad x = \mathrm{last}^l_l(P_{\beta}(t)) = \mathrm{last}^k_l(P_{\beta}(t)) = \mathrm{last}^k_l(P_{\alpha\cup\beta}(t)),$$

which proves (i). We shall now prove that

$$\text{if} \quad x \in \mathrm{Max}\,(t_0) \quad \text{then} \quad C_{\alpha}(x) = C_{\beta}(x). \tag{1}$$

Let i, j, k, l be as in the preceding part of the proof. Then, since $P_\alpha(t) \approx_E P_\alpha(r)$ and $P_\beta(t) \approx_E P_\beta(r)$, we get

$$\text{label}_r(\text{last}_j^i(P_\alpha(r))) = \text{label}_t(\text{last}_j^i(P_\alpha(t))) = \text{label}_t(x)$$
$$= \text{label}_t(\text{last}_l^k(P_\beta(t))) = \text{label}_r(\text{last}_l^k(P_\beta(r))).$$

But, since $i \in \alpha$, $k \in \beta$, applying once again Lemma 4.18 (i) we obtain

$$\text{last}_j^i(P_\alpha(r)) = \text{last}_j^i(P_{\alpha\cup\beta}(r)), \qquad \text{last}_l^k(P_\beta(r)) = \text{last}_l^k(P_{\alpha\cup\beta}(r)).$$

Having the same label, the elements $\text{last}_j^i(P_{\alpha\cup\beta}(r))$ and $\text{last}_l^k(P_{\alpha\cup\beta}(r))$ must be equal, and therefore

$$C_\alpha(x) = \text{last}_j^i(P_\alpha(r)) = \text{last}_l^k(P_\beta(r)) = C_\beta(x),$$

which concludes the proof of (1).

Let $x' := C_\alpha(x) = C_\beta(x)$. Then

$$x' \in \text{LAST}(P_\alpha(r)) \cap \text{LAST}(P_\beta(r)) \subset O(P_\alpha(r)) \cap O(P_\beta(r))$$

and by Lemma 4.25 (v) $x' \in O(r_0)$.

To achieve (ii) is remains to prove that $x' \in \text{Max}(r_0)$. Suppose the contrary, $x' \notin \text{Max}(r_0)$. Then for some $z' \in \text{Max}(r_0)$, $x' <_r z'$. We now apply point (i) of the thesis and (1) substituting z', C_α^{-1}, C_β^{-1}, r for x, C_α, C_β, t. Then we obtain

$$z' \in \text{LAST}(P_\alpha(r)) \cap \text{LAST}(P_\beta(r)) \cap \text{LAST}(P_{\alpha\cup\beta}(r))$$

and $z := C_\alpha^{-1}(z') = C_\beta^{-1}(z')$. Therefore

$$z \in \text{LAST}(P_\alpha(t)) \cap \text{LAST}(P_\beta(t)) \subset O(P_\alpha(t)) \cap O(P_\beta(t)) = O(t_0).$$

But since C_α^{-1} is an isomorphism,

$$x = C_\alpha^{-1}(x') <_t C_\alpha^{-1}(z') = z, \qquad x, z \in O(t_0)$$

in contradiction with the assumption $x \in \text{Max}(t_0)$.

(iii) By (i) $\text{Max}(t_0) \subset \text{LAST}(P_\alpha(t))$ and by Lemma 4.25 (ii) $P_\alpha(t) = t_0 . t'$. Therefore $i \in \text{Dom}(t')$ iff

$$\neg\, \exists\, x \in \text{Max}(t_0),\ \text{last}_i^i(P_\alpha(t)) \leqq_t x. \tag{2}$$

By (ii) $C_\alpha(\text{Max}(t_0)) \subset \text{Max}(r_0)$. On the other hand, applying (ii) to C_α^{-1} we obtain $C_\alpha^{-1}(\text{Max}(r_0)) \subset \text{Max}(t_0)$, thus $C_\alpha(\text{Max}(t_0)) = \text{Max}(r_0)$. Moreover

$\text{LAST}(P_\alpha(t))$ and $\text{LAST}(P_\alpha(r))$ are canonically isomorphic. Thus the condition (2) above holds for $P_\alpha(t)$ iff it holds for $P_\alpha(r)$ and therefore $\text{Dom}(t') = \text{Dom}(r')$. For the same reasons $\text{Dom}(t'') = \text{Dom}(r'')$. □

PROPOSITION 4.27: *Let* $t, r \in E(\Sigma, I)$. *If* $P_\alpha(t) \approx_E P_\alpha(r)$ *and* $P_\beta(t) \approx_E P_\beta(r)$, $\alpha, \beta \subset \text{Proc}$, *then* $P_{\alpha\cup\beta}(t) \approx_E P_{\alpha\cup\beta}(r)$.

Proof: Let $P_{\alpha\cup\beta}(t) = t_0 t' t''$, $P_{\alpha\cup\beta}(r) = r_0 r' r''$ be the factorizations defined in Lemma 4.25. By Lemma 4.26 (iii)

$$\gamma := \text{Dom}(t') = \text{Dom}(r'), \qquad \delta := \text{Dom}(t'') = \text{Dom}(r'').$$

First we prove that

$$\text{last}_j^i(P_{\alpha\cup\beta}(t)) = \begin{cases} \text{last}_j^i(P_\alpha(t)) & \text{if} \quad i \in \gamma \\ \text{last}_j^i(P_\beta(t)) & \text{if} \quad i \in \delta \\ \text{last}_j^i(P_\alpha(t)) = \text{last}_j^i(P_\beta(t)) & \text{if} \quad i \notin \gamma \cup \delta \end{cases} \tag{1}$$

Consider the case $i \in \gamma$. Using Lemmas 4.24 and 4.25, we obtain

$$P_i(P_{\alpha\cup\beta}(t)) = P_i(P_\alpha(t) . t'') = P_{i\cup\xi}(P_\alpha(t)) . P_i(t''),$$

with $\xi = \text{Dom}(P_i(t''))$, but $i \in \gamma$ implies $i \notin \delta$ and $P_i(t'') = \varepsilon$, $\xi = \varnothing$, whence $P_i(P_{\alpha\cup\beta}(t)) = P_i(P_\alpha(t))$. But

$$\text{last}_j^i(P_{\alpha\cup\beta}(t)) = \text{last}_j^j(P_i(P_{\alpha\cup\beta}(t))),$$

which implies (1) for $i \in \gamma$. In the other cases we argue analogously. Clearly, (1) also holds if we replace t by r.

We claim that

$$\left.\begin{array}{c} \forall\, i, j \in \text{Proc}, \\ \text{label}_t(\text{last}_j^i(P_{\alpha\cup\beta}(t))) = \text{label}_r(\text{last}_j^i(P_{\alpha\cup\beta}(r))) \end{array}\right\} \tag{2}$$

Indeed, by (1), if $i \in \gamma$ then

$$\text{last}_j^i(P_{\alpha\cup\beta}(t)) = \text{last}_j^i(P_\alpha(t))$$

and

$$\text{last}_j^i(P_{\alpha\cup\beta}(r)) = \text{last}_j^i(P_\alpha(r)),$$

but $P_\alpha(t) \approx_E P_\alpha(r)$ yields

$$\text{label}_t(\text{last}_j^i(P_\alpha(t))) = \text{label}_r(\text{last}_j^i(P_\alpha(r))),$$

which implies (2). In the other cases we argue in the same way. We shall now verify the following condition

$$\text{last}_j^i(P_{\alpha\cup\beta}(t))\in \text{LAST}(P_\alpha(t)) \Leftrightarrow \text{last}_j^i(P_{\alpha\cup\beta}(r))\in \text{LAST}(P_\alpha(r)). \quad (3)$$

Two cases arise.

In the case $i\in\gamma$, by (1),

$$\text{last}_j^i(P_{\alpha\cup\beta}(t))=\text{last}_j^i(P_\alpha(t))\in \text{LAST}(P_\alpha(t))$$

and

$$\text{last}_j^i(P_{\alpha\cup\beta}(r))=\text{last}_j^i(P_\alpha(r))\in \text{LAST}(P_\alpha(r)).$$

On the other hand, if $i\notin\gamma$ then $\text{last}_j^i(P_{\alpha\cup\beta}(t))=\text{last}_j^i(P_\beta(t))$. Assume in addition that $\text{last}_j^i(P_\beta(t))\in\text{LAST}(P_\alpha(t))$. Then the following implications hold

$$\begin{aligned}\text{last}_j^i(P_\beta(t))&\in \text{LAST}(P_\alpha(t))\cap\text{LAST}(P_\beta(t))\\ &\subset O(P_\alpha(t))\cap O(P_\beta(t))=O(t_0)\\ &\Rightarrow \exists x\in\text{Max}(t_0),\quad \text{last}_j^i(P_\beta(t))\leqq_t x\\ &\Rightarrow \text{last}_j^i(P_\beta(r))\leqq_r C_\beta(x)\in\text{Max}(r_0),\end{aligned}$$

the last implication follows from the definition of C_β and from Lemma 4.26 (ii). But since $i\notin\gamma$, by (1),

$$\text{last}_j^i(P_\beta(r))=\text{last}_j^i(P_{\alpha\cup\beta}(r))$$

and therefore

$$\text{last}_j^i(P_{\alpha\cup\beta}(r))\in O(r_0)\subset O(P_\alpha(r)),$$

and finally, since $P_\alpha(r)$ is a prefix of $P_{\alpha\cup\beta}(r)$, by Lemma 4.20, we get

$$\text{last}_j^i(P_{\alpha\cup\beta}(r))\in\text{LAST}(P_\alpha(r)).$$

This concludes the proof of (3) in one direction. Replacing r by t and vice versa we get the converse.

Let $x=\text{last}_j^i(P_{\alpha\cup\beta}(t))$, $y=\text{last}_l^k(P_{\alpha\cup\beta}(t))$ and $x<_t y$. Then $\text{Dom}(t')\cap\text{Dom}(t'')=\varnothing$ implies that either $x, y\in O(P_\alpha(t))$ or $x, y\in O(P_\beta(t))$. In the first case, by Lemma 4.20, $x, y\in\text{LAST}(P_\alpha(t))$ and by (3) the corresponding elements $x'=\text{last}_j^i(P_{\alpha\cup\beta}(r))$ and $y'=\text{last}_l^k(P_{\alpha\cup\beta}(r))$ belong to $\text{LAST}(P_\alpha(r))$. Moreover, by (2), $\text{label}_t(x)=\text{label}_r(x')$ and $\text{label}_t(y)=\text{label}_r(y')$, thus $P_\alpha(t)\approx_E P_\alpha(r)$ and $x<_t y$ imply $x'<_r y'$. The case $x, y\in O(P_\beta(t))$ can be handled in the same way. Thus we have proved that the partial orders $\leqq_t|\text{LAST}(P_{\alpha\cup\beta}(t))$ and $\leqq_r|\text{LAST}(P_{\alpha\cup\beta}(r))$ are canonically isomorphic and this fact and (2) give the thesis. □

LEMMA 4.28: *Let* $t, r\in E(\Sigma, I)$ *and* $t\approx_E r$. *Then for all* $\eta, \xi\subset\text{Proc}$, $\text{Dom}(P_\eta(S_\xi(t)))=\text{Dom}(P_\eta(S_\xi(r)))$.

Proof: Let $\text{Pref}_\eta(\text{Suff}_\xi(t))$ denote the following subset of $O(t)$ $\{x\in O(t): x\in\text{Suff}_\xi(t)\wedge\exists y\in\text{Suff}_\xi(t),\quad (x\leqq_t y\wedge\eta\cap\text{Dom}(y)\neq\varnothing)\}$. Clearly, from Fact 4.6 it follows that $\text{Pref}_\eta(\text{Suff}_\xi(t))$ contains the action occurrences corresponding to the subtrace $P_\eta(S_\xi(t))$ of t. We shall show that

$$\text{Pref}_\eta(\text{Suff}_\xi(t))=\text{Suff}_\xi(t)\cap\text{Pref}_\eta(t). \quad (1)$$

Indeed, again by Fact 4.6, $x\in\text{Suff}_\xi(t)\cap\text{Pref}_\eta(t)$ iff x fulfils the condition

$$x\in\text{Suff}_\xi(t)\wedge\exists y\in O(t),\qquad (x\leqq y\wedge\eta\cap\text{Dom}(y)\neq\varnothing). \quad (2)$$

Clearly, $x\in\text{Pref}_\eta(\text{Suff}_\xi(t))$ implies (2), since $\text{Suff}_\xi(t)\subset O(t)$. On the other hand, $x\in\text{Suff}_\xi(t)\wedge x\leqq y$ implies $y\in\text{Suff}_\xi(t)$, thus y in (2) belongs to $\text{Suff}_\xi(t)$ and (2) implies $x\in\text{Pref}_\eta(\text{Suff}_\xi(t))$, which concludes the proof of (1). Replacing $\text{Suff}_\xi(t)$ by $O(t)-\text{Pref}_{\bar\xi}(t)$ in (1) we obtain

$$\text{Pref}_\eta(\text{Suff}_\xi(t))=\text{Pref}_\eta(t)-\text{Pref}_{\bar\xi}(t). \quad (3)$$

From (3) it follows that

$$i\in\text{Dom}(P_\eta(S_\xi(t)))\qquad\text{iff}\quad \text{last}_i^i(P_{\bar\xi}(t))<_t\text{last}_i^i(P_\eta(t)) \quad (4)$$

and by Lemma 4.18(ii) the last condition is equivalent with

$$\max\{\text{last}_i^j(t): j\in\bar\xi\}<_t\max\{\text{last}_i^j(t): j\in\eta\}. \quad (5)$$

Obviously, formulae analogous with (3), (4) and (5) hold for the trace r. Let C be the canonical isomorphism of $\leqq_t|\text{LAST}(t)$ and $\leqq_r|\text{LAST}(r)$. Then

$$\max\{\text{last}_i^j(t): j\in\bar\xi\}<_t\max\{\text{last}_i^j(t): j\in\eta\}$$

iff

$$C(\max\{\text{last}_i^j(t): j\in\bar\xi\})<_r C(\max\{\text{last}_i^j(t): j\in\eta\})$$

iff

$$\max\{\text{last}_i^j(r): j\in\bar\xi\}<_r\max\{\text{last}_i^j(r): j\in\eta\},$$

whence by (4) $i\in\text{Dom}(P_\eta(S_\xi(t)))$ iff $i\in\text{Dom}(P_\eta(S_\xi(r)))$. □

Proof of Theorem 4.15: Let $P_{\alpha\cup\beta}(t)=t_0 t' t''.$, $P_{\alpha\cup\beta}(r)=r_0 r' r''$ be the factorizations of $P_{\alpha\cup\beta}(t)$ and $P_{\alpha\cup\beta}(r)$ defined in Lemma 4.25. By Lemma 4.26(iii) $\gamma:=\mathrm{Dom}(t')=\mathrm{Dom}(r')$ and $\delta:=\mathrm{Dom}(t'')=\mathrm{Dom}(r'')$. By Proposition 4.27, to complete the proof of Theorem 4.15 it suffices to show that $P_{\alpha\cup\beta}(t)\approx_T P_{\alpha\cup\beta}(r)$. Let $\eta\subset\mathrm{Proc}$. Then by Lemmas 4.24 and 4.25 we have

$$S_\eta(P_{\alpha\cup\beta}(t))=S_\eta(P_\alpha(t).t'')=S_{\eta-\xi_1}(P_\alpha(t)).S_\eta(t'')$$

and

$$S_\eta(P_{\alpha\cup\beta}(r))=S_\eta(P_\alpha(r).r'')=S_{\eta-\xi_2}(P_\alpha(r)).S_\eta(r''),$$

with

$$\xi_1=\mathrm{Dom}(P_\eta^-(t'')),\qquad \xi_2=\mathrm{Dom}(P_\eta^-(r'')).$$

But by Lemma 4.25(iv) $t''=S_\delta(P_\beta(t))$ and $r''=S_\delta(P_\beta(r))$. Thus finally

$$\xi_1=\mathrm{Dom}(P_\eta^-(S_\delta(P_\beta(t))))\qquad\text{and}\qquad \xi_2=\mathrm{Dom}(P_\eta^-(S_\delta(P_\beta(r)))).$$

Since $P_\beta(t)\approx_E P_\beta(r)$, from Lemma 4.28 it follows that $\xi:=\xi_1=\xi_2$.

Moreover $P_\alpha(t)\approx_T P_\alpha(r)$ yields $S_{\eta-\xi}(P_\alpha(t))\sim_T S_{\eta-\xi}(P_\alpha(r))$, whereas $P_\beta(t)\approx_T P_\beta(r)$ and Lemma 4.23(i) yield

$$S_\eta(t'')=S_\eta(S_\delta(P_\beta(t)))=S_{\eta\cap\delta}(P_\beta(t))\sim_T S_{\eta\cap\delta}(P_\beta(r))$$
$$=S_\eta(S_\delta(P_\beta(r)))=S_\eta(r'').$$

Therefore

$$S_\eta(P_{\alpha\cup\beta}(t))=S_{\eta-\xi}(P_\alpha(t)).S_\eta(t'')\sim_T S_{\eta-\xi}(P_\alpha(r)).S_\eta(r'')=S_\eta(P_{\alpha\cup\beta}(r)),$$

which concludes the proof. □

PROPOSITION 4.29: *Let* $t_1,\ r_1\in E(\Sigma,I),\ a\in\Sigma,\ t_2=t_1 a,\ r_2=r_1 a$ *and* $\forall i\in\mathrm{Dom}(a),\ P_i(t_1)\approx_E P_i(r_1)$. *Then*

$$\forall i\in\mathrm{Dom}(a),\quad P_i(t_2)\approx_E P_i(r_2).$$

Proof: Let $\mathrm{Dom}(a)=\alpha$. Then by Proposition 4.27 $P_\alpha(t_1)\approx_E P_\alpha(r_1)$.

Let $i\in\alpha$. Then by Fact 4.8 (i), $P_i(t_2)=P_\alpha(t_2)=P_\alpha(t_1).a$ and $P_i(r_2)=P_\alpha(r_2)=P_\alpha(r_1).a$.

Let $\#_a t_2=m$, $\#_a r_2=k$, $x=a^m\in O(t_2)$, $y=a^k\in O(r_2)$. By Lemma 4.18(iv) we obtain

$$\mathrm{last}_j^i(P_\alpha(t_2))=\begin{cases}\mathrm{last}_j^i(P_\alpha(t_1)) & \text{if } i\notin\alpha,\ j\in\mathrm{Proc}\\ x & \text{if } i,j\in\alpha\\ \max\{\mathrm{last}_j^k(P_\alpha(t_1)):k\in\alpha\} & \text{if } i\in\alpha,\ j\notin\alpha\end{cases}\tag{1}$$

The same formula holds for the trace r_2 if we replace x by y. Moreover, $\forall z\in O(P_\alpha(t_1))$, $z<_{t_2}x$ and $\forall z\in O(P_\alpha(r_1))$, $z<_{r_2}y$. Let C_1 be the canonical isomorphism of $\leqq_{t_1}|\mathrm{LAST}(P_\alpha(t_1))$ and $\leqq_{r_1}|\mathrm{LAST}(P_\alpha(r_1))$. Then the arguments above show that C_2 defined by

$$C_2(x)=y,\qquad \forall z\in\mathrm{LAST}(P_\alpha(t_2))-\mathrm{LAST}(P_\alpha(t_1)),$$
$$C_2(z)=C_1(z)$$

is the canonical isomorphism of $\leqq_{t_2}|\mathrm{LAST}(P_\alpha(t_2))$ and $\leqq_{r_2}|\mathrm{LAST}(P_\alpha(r_2))$. Moreover, since

$$\forall z\in\mathrm{LAST}(P_\alpha(t_2))-\{x\},$$
$$\mathrm{label}_{t_1}(z)=\mathrm{label}_{r_1}(C_1(z))=\mathrm{label}_{r_2}(C_2(z))$$

and

$$P_x(t_2)=P_\alpha(t_2),\qquad P_y(r_2)=P_\alpha(r_2),$$

Algorithm 4.9 will attach the same label to x in t_2 and to y in r_2. □

Proof of Theorem 4.16: Let $\mathrm{Dom}(a)=\alpha$. Then by Theorem 4.15 $P_\alpha(t_1)\approx P_\alpha(r_1)$ and by Proposition 4.29 $P_\alpha(t_2)\approx_E P_\alpha(r_2)$. Let $i\in\alpha$. Then Fact 4.8(i) yields

$$P_i(t_2)=P_\alpha(t_2)=P_\alpha(t_1).a$$

and

$$P_i(r_2)=P_\alpha(r_2)=P_\alpha(r_1).a.$$

Thus it remains to prove that $P_\alpha(t_2)\approx_T P_\alpha(r_2)$. Let $\eta\subset\mathrm{Proc}$. Then by Lemma 4.24

$$S_\eta(P_\alpha(t_2))=S_\eta(P_\alpha(t_1).a)=S_{\eta-\gamma}(P_\alpha(t_1)).S_\eta(a)$$

and similarly

$$S_\eta(P_\alpha(r_2))=S_{\eta-\gamma}(P_\alpha(r_1)).S_\eta(a),$$

where $\gamma = \mathrm{Dom}(P_\eta(a))$.

Since $P_\alpha(t_1) \approx_T P_\alpha(r_1)$, we have

$$S_{\eta-\gamma}(P_\alpha(t_1)) \sim_T S_{\eta-\gamma}(P_\alpha(r_1)).$$

Multiplying both sides by $S_\eta(a)$ we get $S_\eta(P_\alpha(t_2)) \sim_T S_\eta(P_\alpha(r_2))$. □

5. LOOSLY COOPERATING FINITE ASYNCHRONOUS AUTOMATA

The synchronization mechanism used in finite asynchronous automata is rather complicated. We may ask if it can be simplified without loss of computability power. In ASYN automata when we perform an action $a \in \Sigma$, the next state of a process $i \in \mathrm{Dom}(a)$ depends on a and it depends on current states of all other processes from $\mathrm{Dom}(a)$. In this section we examine parallel automata with a simpler synchronization mechanism.

A loosly cooperating asynchronous automaton, LCASYN in abbreviation, is a tuple $\mathbb{A} = (\mathbb{P}_1, \ldots, \mathbb{P}_n, \mathbb{F})$, where for $i \in \bar{n}$, $\mathbb{P}_i = (\Sigma_i, S_i, s_i^0, \delta_i)$ is the i-th process. Σ_i, S_i, s_i^0 have the same meaning as in the case of ASYN automata.

$\delta_i : S_i \times \Sigma_i \to \mathscr{P}(S_i)$ is the next-state function of $\mathbb{P}_i$. As previously $\mathbb{F} \subset S = \underset{i=1}{\overset{n}{\times}} S_i$ is the set of final states. $\mathbb{A}$ is deterministic if

$$\forall i \in \bar{n}, \quad \forall s_i \in S_i, \quad \forall a \in \Sigma_i, \qquad \mathrm{card}(\delta_i(s_i, a)) \leqq 1.$$

The transition between global states is defined as follows, let

$$(s_1', \ldots, s_n'), (s_1'', \ldots, s_n'') \in S, \qquad a \in \Sigma,$$

then

$$(s_1', \ldots, s_n') \overset{a}{\Rightarrow} (s_1'', \ldots, s_n'')$$

if $\forall i \notin \mathrm{Dom}(a)$, $s_i' = s_i''$ and $\forall i \in \mathrm{Dom}(a)$, $s_i'' \in \delta_i(s_i', a)$.

In the same way as in the case of an ASYN automaton we define how traces act on $\mathbb{A}$, the independency relation $I_{\mathbb{A}}$, the language $L(\mathbb{A})$ and the trace language $T(\mathbb{A})$ recognized by $\mathbb{A}$. Every LCASYN can be transformed to an ASYN automaton if we define

$$\delta_a(s_{i_1}', \ldots, s_{i_k}') = (\delta_{i_1}(s_{i_1}', a), \ldots, \delta_{i_k}(s_{i_k}', a))$$

for $\{i_1, \ldots, i_k\} = \mathrm{Dom}(a)$.

THEOREM 5.1: *For every finite trace language $T \subset E(\Sigma, I)$ there exists a loosly cooperating finite asynchronous automaton $\mathbb{A}$ such that $I_{\mathbb{A}} = I$ and $T(\mathbb{A}) = T$.*

Proof: Let $D = \Sigma \times \Sigma - I$ and $\mathrm{Cliques}(D) = \{\Sigma_1, \ldots, \Sigma_n\}$ and let $h_i : \Sigma^* \to \Sigma_i^*$ be projections of Σ^* onto $\Sigma_i^*, i \in \bar{n}$. We consider the languages $L_i = \{h_i(t) : t \in T\}$ (*see* Proposition 2.12). Every L_i is finite. We can build a deterministic finite state acceptor $A_i = (\Sigma_i, Q_i, q_0^i, \delta_i, F_i)$ of L_i such that

$$\forall u, v \in L_i, u \neq v \quad \Rightarrow \quad \delta_i(q_0^i, u) \neq \delta_i(q_0^i, v).$$

Then $\mathbb{A} = (\mathbb{P}_1, \ldots, \mathbb{P}_n, \mathbb{F})$, where for all $i \in \bar{n}$ $\mathbb{P}_i = (\Sigma_i, Q_i, q_0^i, \delta_i)$ and

$$\mathbb{F} = \{(q_1, \ldots, q_n) : \exists u_1 \in L_1, \ldots, \exists u_n \in L_n, \forall i \in \bar{n}, \delta_i(q_0^i, u_i) = q_i, \mid |_{i=1}^n u_i \in T\}.$$

By Proposition 2.12 this construction is correct. □

PROPOSITION 5.2: *Let $\mathbb{A}$ be can* LCASYN *automaton. Then there exists a deterministic* LCASYN *automaton $\mathbb{B}$ such that $T(\mathbb{A}) = T(\mathbb{B})$.*

Proof: We can transform every process $\mathbb{P}_i$ of $\mathbb{A}$ to a deterministic process in exactly the same way as we transform a nondeterministic finite state acceptor to a deterministic *fsa*, changing suitably the set of final states. □

PROPOSITION 5.3: *For every* LCASYN *automaton $\mathbb{A}$ there exists an* LCASYN *automaton $\mathbb{B}$ in the normal form such $T(\mathbb{A}) = T(\mathbb{B})$.*

Proof: As in Proposition 4.3. □

THEOREM 5.4: *There exist regular trace languages which are not recognizable by the loosly cooperating finite asynchronous automata.* □

A simple proof of the above theorem will be given in the next section.

By $\mathrm{LCReg}(\Sigma, I)$ we shall denote the class of trace languages recognized by LCASYN automata.

PROPOSITION 5.5: *If $T_i \in \mathrm{LCReg}(\Sigma_i, I_i)$, $i = 1,2$, then*

$$T = T_1 \| T_2 \in \mathrm{LCReg}(\Sigma, I),$$

where $(\Sigma, I) = (\Sigma_1, I_1) \| (\Sigma_2, I_2)$.

Proof: Let $\mathbb{A}' = (\mathbb{P}_1', \ldots, \mathbb{P}_n', \mathbb{F}')$, $\mathbb{A}'' = (\mathbb{P}_1'', \ldots, \mathbb{P}_k'', \mathbb{F}'')$ be LCASYN automata recognizing T_1 and T_2. Then

$$\mathbb{A} = (\mathbb{P}_1', \ldots, \mathbb{P}_n', \mathbb{P}_1'', \ldots, P_k'', \mathbb{F}),$$

where

$$\mathbb{F}=\{(s'_1,\ldots,s'_n,s''_1,\ldots,s''_k):(s'_1,\ldots,s'_n)\in\mathbb{F}',(s''_1,\ldots,s''_k)\in\mathbb{F}''\}$$

recognizes T. □

PROPOSITION 5.6: *If* T_1, $T_2\in\mathrm{LCReg}(\Sigma, I)$ *then* $T_1\cup T_2$, $T_1\cap T_2\in$ LCReg(Σ, I).

Hint. Given LCASYN automata $\mathbb{A}'$, $\mathbb{A}''$ in the normal form recognizing T_1 and T_2 we can combine their corresponding processes $\mathbb{P}'_i$, $\mathbb{P}''_i$ applying the standard construction known from the automata theory. □

6. AN ALGEBRAIC CHARACTERIZATION OF REGULAR TRACE LANGUAGES

Theorems 4.4 and 4.5 lead immediately to a new characterization of regular trace languages, different from that given by Ochmanski [13]. While preparing the revised version of our paper we learned about two papers of C. Duboc [7, 8], where, among other things, the results of this section are presented. Thus we give here only some hints, referring the reader to [8] for full proofs. Let (Σ, I) be a concurrent alphabet. If the independency relation is empty then, for $u\in\Sigma^*$, $[u]_I=\{u\}$. Thus the monoids $E(\Sigma, \varnothing)$ and Σ^* are isomorphic. Under this isomorphism, regular trace languages are equivalent with regular languages. From now on every language L will be identified with a trace language $\{\{u\}:u\in L\}\subset E(\Sigma, \varnothing)$.

THEOREM 6.1: *Let* (Σ, I) *be a concurrent alphabet and Cliques* $(D)=\{\Sigma_1,\ldots,\Sigma_n\}$. *For every trace language* $T\in\mathrm{LCReg}(\Sigma, I)$ *there exist regular languages* L_{ij}, $i\in\bar{k}$, $j\in\bar{n}$, such that $T=\bigcup_{i=1}^{k}(\|_{j=1}^{n}L_{ij})$.

Proof: Let $\mathbb{A}=(\mathbb{P}_1,\ldots,\mathbb{P}_n,\mathbb{F})$ be an LCASYN automaton in the normal form recognizing T, for all $i\in\bar{n}$ $\mathbb{P}_i=(\Sigma_i, S_i, s_i^0, \delta_i)$. For every $s=(s_1,\ldots,s_n)\in\mathbb{F}$ we create the following finite state acceptors

$$A_{si}=(\Sigma_i, S_i, s_i^0, \delta_i, \{s_i\}),\qquad i\in\bar{n}.$$

Then

$$T(\mathbb{A})=\bigcup_{s\in F}(\|_{i=1}^{n}L(A_{si})),\qquad (\Sigma, I)=\|_{i=1}^{n}(\Sigma_i, \varnothing).\quad\square$$

As an application of the above theorem we give a proof of Theorem 5.4. Let

$$L=((ab\cup a^2b^2)c)^*,\qquad I=\{(a, b), (b, a)\}$$
$$\Sigma=\{a, b, c\}\qquad\text{and}\qquad T=\{[u]_I:u\in L\}.$$

This trace language is recognized by the ASYN automaton from Example 4.2. Let $L_1=((a\cup a^2)c)^*$ and let $g:\{a, c\}^*\to\{b, c\}^*$ be the homomorphism defined by $g(a)=b$, $g(c)=c$. There for $x\in L_1$ there exists exactly one $y\in\{b, c\}^*$, $y=g(x)$, such that $x\| y\in T$. Thus T cannot be decomposed in the form given by Theorem 6.1.

Let LCReg be the union of the families LCReg(Σ, I) for all concurrent alphabets (Σ, I).

COROLLARY 6.2: LCReg *is the least family* $\mathscr{R}$ *of trace languages such that*
(i) *every regular language belongs to* $\mathscr{R}$
(ii) $\mathscr{R}$ *is closed under* $\cup$ *and* $|\ |$.

Here $\cup$ *is understood as a partial operation defined only for trace languages over the same alphabets.*

Proof: Immediate consequence of Propositions 5.5, 5.6 and Theorem 6.1. □

THEOREM 6.3: *Let* T *be a regular trace language over a concurrent alphabet* (Σ, I) *and let* Cliques$(D)=\{\Sigma_1,\ldots,\Sigma_n\}$. *Then there exist*
(i) *a concurrent alphabet* $(\tilde{\Sigma}, \tilde{I})$,

$$\text{Cliques}(\tilde{D})=\{\tilde{\Sigma}_1,\ldots,\tilde{\Sigma}_n\};$$

(ii) *an elementary homomorphism* $f:E(\tilde{\Sigma}, \tilde{I})\to E(\Sigma, I)$ *such that* $\forall i\in\bar{n}$, $f(\tilde{\Sigma}_i)=\Sigma_i$;

(iii) *a family of regular languages* L_{ij}, $i\in\bar{k}$, $j\in\bar{n}$, *where* $\forall i\in\bar{k}$, $L_{ij}\subset\tilde{\Sigma}_j^*$ *such that*

$$T=f\left(\bigcup_{i=1}^{k}(\|_{j=1}^{n}L_{ij})\right).$$

Proof: Let $\mathbb{A}=(\mathbb{P}_1,\ldots,\mathbb{P}_n,\Delta,\mathbb{F})$ be an ASYN automaton recognizing T. For every $a\in\Sigma$, s', $s''\in\mathop{\mathbf{X}}_{i\in\mathrm{Dom}(a)}S_i$, such that $s''\in\delta_a(s')$, $\delta_a\in\Delta$, we create a new action $(s', a, s'')\in\tilde{\Sigma}$. We define $f((s', a, s''))=a$. The next-state functions

are defined in the following way, for $i_j \in \mathrm{Dom}(a)$,

$$\delta_{i_j}((s', a, s''), s'_{i_j}) = s''_{i_j},$$

where

$$s' = (s'_{i_1}, \ldots, s'_{i_j}, \ldots, s'_{i_k}), \qquad s'' = (s''_{i_1}, \ldots, s''_{i_j}, \ldots, s''_{i_k}).$$

In this way we obtain an LCASYN automaton and applying Theorem 6.1 and the homomorphism f defined above we have the thesis. □

Let Reg be the union of all families $\mathrm{Reg}(\Sigma, I)$ for all concurrent alphabets (Σ, I).

COROLLARY 6.4: *Reg is the least family $\mathscr{R}$ of trace languages such that*

(i) *every regular language belongs to $\mathscr{R}$;*

(ii) *$\mathscr{R}$ is closed under $\| \ \|$ and $\cup$ (here $\cup$ is understood as a partial operation defined only for trace languages over the same alphabets);*

(iii) *for every $T \in \mathscr{R}$ over a concurrent alphabet (Σ, I) and for every elementary homomorphism*

$$f: \quad E(\Sigma, I) \to E(\Sigma', I'), \qquad f(T) \in \mathscr{R}$$

Proof: By Theorem 6.3, Proposition 3.3, Lemmas 3.4, 3.5. □

Conclusing remarks: The concept of the finite asynchronous automaton arises as a natural extension of the concept of the finite state acceptor when we pass from sequential to parallel computations. For this reason all questions and problems which have been considered for *fsa* can be posed for the ASYN automata. In particular, infinite computations seem to be very interesting from the point of view of concurrency theory. However, the development of the theory of the asynchronous automata may need considerable efforts. Another source of problems is concurrency theory, where questions concerning deadlock and fairness seem to be of greatest interest.

ACKNOWLEDGMENTS

The autor wishes to thank Ryszard Janicki for his suggestions and encouragement.

We are also grateful to A. Arnold for his helpful criticism that led to significant improvements in the paper.

REFERENCES

1. A. BERTONI, M. BRAMBILLA, G. MAURI and N. SABADINI, *An Application of the Theory of Tree Partially Commutative Monoids: Asymptotic Densities of Trace Languages*, Lecture Notes in Comp. Sci., Vol. 117, 1981, Springer-Verlag, pp. 205-215.
2. A. BERTONI, G. MAURI and N. SABADINI, *A Hierarchy of Regular Trace Languages and Some Combinatorial Applications*, 2nd World Conf. on Mathematics at the Service of Men, Las Palmas, 1982.
3. P. CARTIER and D. FOATA, *Problèmes combinatoires de commutation et réarrangements*, Lecture Notes in Math., Vol. 85, 1969, Springer-Verlag.
4. M. CLERBOUT and M. LATTEUX, *Partial Commutations and Faithful Rational Transductions*, Theoretical Comp. Science, Vol. 34, 1984, pp. 241-254.
5. R. CORI and Y. METIVIER, *Recognizable Subsets of Partially Abelian Monoids*, Theoretical Comp. Science, Vol. 35, 1985, pp. 179-189.
6. R. CORI and D. PERRIN, *Sur la reconnaissabilité dans les monoïdes partiellement commutatifs libres*, R.A.I.R.O. Inform. Théor., Vol. 19, 1985, pp. 21-32.
7. C. DUBOC, *Commutations dans les monoïdes libres : un cadre théoretique pour l'étude du parallélisme*, Thèse de 3e cycle, also Technical Report 86-25, Laboratoire Informatique Théoretique et Programmation, 1986.
8. C. DUBOC, *Mixed Product and Asynchronous Automata* (submitted for publication).
9. S. EILENBERG, *Automata, Languages and Machines*, Vol. A, 1974, Academic Press.
10. R. JANICKI, *Synthesis of Concurrent Schemes*, Lecture Notes in Comp. Sc., Vol. 64, 1978, Springer Verlag, pp. 298-307.
11. G. LALLEMENT, *Semigroups and Combinatorial Applications*, 1979, J. Wiley and Sons, New York.
12. A. MAZURKIEWICZ, *Concurrent Program Schemes and Their Interpretations*, DAIMI-PB-78, Aarhus University, 1977.
13. E. OCHMAŃSKI, *Regular Behaviour of Concurrent Schemes*, E.A.T.C.S. Bulletin, No. 27, October 1985, pp. 56-67.
14. W. REISIG, *Petrinetze. Eine Einführung*, 1982, Springer Verlag.
15. M. W. SHIELDS, *Non-Sequential Behaviour*: 1, Report CSR-120-82, Department of Comp. Science, University of Edinburgh, 1982.
16. P. H. STARKE, *Free Petri Net Languages*, Lecture Notes in Comp. Sc., Vol. 64, Springer Verlag, pp. 506-515.
17. A. TARLECKI, *Notes on the Implementability of Formal Languages by Concurrent Systems*, Institute of Computer Science of Polish Academy of Sciences, Report 481, Warsaw, 1982.