Time-stamps for Mazurkiewicz traces

Wiesław Zielonka [1]

LIAFA, case 7014, Université Paris 7 and CNRS
2, Place Jussieu, 75251 Paris Cedex 05, France

Abstract

We construct a new time-stamp system for Mazurkiewicz traces. We begin by constructing a sequential time-stamp system which turns out to be optimal for a certain class of time-stamps. In the next step we show that this time-stamp system can be adapted for Mazurkiewicz traces, i.e. it can be used also in a distributed environment.

1 Introduction

Mazurkiewicz traces provide a well-established natural model of the behaviour of simple concurrent systems. The theory is rich and still under active development. Although many facts concerning traces generalise similar results known for words, often these generalisations are far from trivial. The appropriate notion of finite automata for traces, so-called asynchronous automata, was introduced in [11].

The construction of deterministic asynchronous automata was (and still is) much more involved than the construction of finite automata for words and was carried out in two steps. First a special system of finite event labellings – a time-stamp system – was constructed and it was shown that this labelling can be updated locally by an asynchronous automaton. At the second step the information related to the trace language to recognize was incorporated into the automaton.

Although there is no formal proof that the first step is necessary, nobody was able to provide a general construction of deterministic asynchronous automata without some sort of time-stamp system (note however that, as shown in [9], non-deterministic asynchronous automata do not need such time-stamps and

[1] Supported by ACI Sécurité informatique 2003-22 VERSYDIS.

deterministic asynchronous automata for some special dependency relations do without them as well [4,7]). But even if some day it turns out to be possible to construct deterministic asynchronous automata without time-stamps, time-stamp systems are of interest by themselves since they provide pertinent information about the order of events in traces and they allow us to update this information in a distributed way with a finite memory.

Asynchronous automata come in two distinct flavours. For the original model of [11] Mukund and Sohoni [8] provide a new, easier to grasp and more transparent, time-stamp system and apply it to a gossip problem. Another type of asynchronous automata – asynchronous cellular automata – was introduced in [2]. The time-stamp system of [2] is also much easier to comprehend than the original one. As in [8], the system of [2] was applied to time-stamp messages in a distributed environment.

In this paper we deal with the model of asynchronous cellular automata of [2]. We construct a new time-stamp system for this model with $n2^{n-1}$ time-stamps, where n is the number of agents. To compare, the system of [2] uses $O(n^n)$ time-stamps. More remarkably, the system that we present here was initially devised for sequential, non-distributed, environment. In fact, this the minimal possible sequential time-stamp system in a specific class of time-stamp systems. Now it turns out that the same time-stamp system, but with a modified update algorithm, can serve in distributed environments modelled by traces.

The paper is organized as follows. Section 2 defines a class of sequential time-stamp systems. In Section 3 we construct a sequential time-stamp system minimal in this class. Finally, in Section 4 we show how to adapt this time-stamp system to traces. The results presented in Sections 2 and 3 are fairly old, they circulated previously in an unpublished manuscript due to the author and were partially used for example in [3]. However, the application for traces given in Section 4 is new.

2 Sequential time-stamp systems

Let Σ be a finite set of agents that communicate by messages which they leave in a box. The access to the box is sequential and its capacity is bounded, for every agent $a \in \Sigma$ the box can contain at most one message sent by a and if the box contains already a message of a then agent a can put a new message in the box only by replacing his own old out-of-date message. The box is not completely reliable, a message put into the box can at any moment disappear without trace (or equivalently we can assume that there is a malicious adversary which in order to hinder the communication can at

any moment remove any number of messages from the box). The messages are stamped when they enter the box. The aim of the stamping is twofold:

(1) the stamp should provide the identity of the message sender and
(2) comparing the stamps of two messages in the box we should be able to deduce the order in which they were deposited.

Formally, a time-stamp system for a set Σ of agents satisfies the following requirements:

- For each agent $a \in \Sigma$ there is a finite set S_a of time-stamps used by a. The sets S_a are pairwise disjoint for different agents and $S = \bigcup_{a \in \Sigma} S_a$ will stand for the set of all time-stamps.
- $\mathcal{C}$ is a family of subsets of S, elements of $\mathcal{C}$ are called configurations. Intuitively, each configuration $C \in \mathcal{C}$ represents the set of time-stamps of the messages that can be present in the box at some moment. We assume that for each configuration C and each agent $a \in \Sigma$, $|C \cap S_a| \leq 1$, which accounts for the fact that at any moment there can be at most one message from the agent a in the box. Moreover, since messages can be removed from the box at any moment, for any valid configuration $C \in \mathcal{C}$ all subsets C' of C are also valid configurations and belong to $\mathcal{C}$.
- $\prec \subseteq S \times S$ is a binary relation over S. For each configuration $C \in \mathcal{C}$, the relation $\prec$ restricted to C is a total (strict) ordering over C (i.e. $\prec$ is irreflexive, antisymmetric and transitive over C). If C is the set of time-stamps of messages present in the box then for any $s_1, s_2 \in C$, $s_1 \prec s_2$ indicates that the message stamped with s_1 was put in the box *before* the message stamped with s_2. If $t \prec s$ for two time-stamps t, s then we say that s *dominates* t.
- To satisfy the requirement that agents should be able to deposit messages at any moment with time-stamps indicating the correct date ordering we assume that for any agent $a \in \Sigma$ and any configuration $C \in \mathcal{C}$ such that $C \cap S_a = \emptyset$ there exist a time-stamp $s \in S_a$ such that $C \cup \{s\} \in \mathcal{C}$ and, for all $t \in C, t \prec s$.

  Thus when an agent a withdraws, if necessary, his old message from the box and C is the time-stamp configuration of the remaining messages then a can always find a time-stamp $s \in S_a$ dominating all $t \in C$ and use s with his new message.

3 Minimal time-stamp system

In this section we construct a minimal time-stamp system for a given finite set Σ of agents.

For $a \in \Sigma$ the set S_a of time-stamps for a consists of all mappings $f : \Sigma \rightarrow$

$\{0, 1, \perp\}$ such that $f(a) = \perp$ and for all $b \in \Sigma \setminus \{a\}, f(b) \in \{0, 1\}$.

For each time-stamp $f \in S$ we shall denote by a_f the corresponding agent, i.e. a_f is the unique element of Σ for which $f(a_f) = \perp$.

Definition 1 *In the sequel we fix a total order relation $<$ over Σ.*

Let $f, g \in S$ be such that $a_f < a_g$. Then

(i) either $f(a_g) = g(a_f)$ and then we set $f \prec g$,
(ii) or $f(a_g) \neq g(a_f)$ and then $g \prec f$.

If $a_f = a_g$ then we assume that neither $f \prec g$ nor $g \prec f$, i.e. only time-stamps of different agents are comparable by $\prec$.

Note that the definition given above implies that if $a_f \neq a_g$ then either $f \prec g$ or $g \prec f$ and exactly one of these alternatives holds.

A set $C \subseteq S$ of time-stamps is said to be *consistent* if $\forall a \in \Sigma$, $|C \cap S_a| \leq 1$ and the relation $\prec$ restricted to C is a total ordering. As the set $\mathcal{C}$ of configurations we take the set of all consistent subsets of S.

Let us note that if there are more than two agents then there exist inconsistent subsets X of S satisfying $|X \cap S_a| \leq 1$ for all $a \in \Sigma$. Thus the notion of consistency is necessary to exclude such sets from the family of valid configurations.

The following lemma shows that each player has always enough time-stamps to put a new message in the box.

Lemma 2 *Let $a \in \Sigma$ and suppose that $X \subset S$ is a set of time-stamps such that*

(1) $X \cap S_a = \emptyset$ and
(2) $\forall b \in \Sigma \setminus \{a\}, |X \cap S_b| \leq 1$.

Then there exists a time-stamp $g \in S_a$ which dominates all elements of X, i.e. $f \prec g$ for all $f \in X$.

PROOF. Let us take $g \in S_a$ such that for all $f \in X$, if $a_f < a_g$ then $g(a_f) = f(a_g)$, otherwise, if $a_g < a_f$ then $g(a_f) = 1 - f(a_g)$. If for some $b \in \Sigma \setminus \{a\}$ there is no f in X with $f(b) = \perp$ then we can set $g(b)$ either to 0 or to 1. Clearly, g chosen in this way dominates all elements of X. $\square$

Note that Lemma 2 does not require for the set X to be consistent. However, if X is consistent and $X \cap S_a = \emptyset$ then the proof of Lemma 2 provides a

time-stamp $g \in S_a$ such that $X \cup \{g\}$ is also consistent. In particular, starting from the empty set of time-stamps we can add one by one new time-stamps to obtain a consistent set having one time-stamp for each agent.

Now let us note for $n = |\Sigma|$ agents, in the system constructed in this section each agent has 2^{n-1} time-stamps and thus $|S| = n2^{n-1}$. In the next lemma we show that this is also the lower bound for the number of time-stamps necessary for n agents, thus our system has the optimal size.

Lemma 3 *Any time-stamp system for n agents requires at least $n2^{n-1}$ time-stamps.*

PROOF. We proceed by induction on the number n of agents. Let us assume that the lemma holds for all time-stamp systems with less than n agents.

Let us take any time-stamp system for the set Σ of agents with $n = |\Sigma|$. Since the relation $\prec$ is used only to compare time-stamps belonging to different agents we can assume without loss of generality that

$$\text{for all } a \in \Sigma \text{ and all } s, t \in S_a \text{ neither } s \prec t \text{ nor } t \prec s \ . \tag{1}$$

Then direct inspection of the conditions defining time-stamp systems in Section 2 shows that for any agent $a \in \Sigma$ and any time-stamp $s \in S_a$ the set

$$\mathrm{dom}(s) = \{t \in S \mid s \prec t\} \subseteq \bigcup_{b \in S \setminus \{a\}} S_b \tag{2}$$

of all the time-stamps dominating s forms a time-stamp system for the set $\Sigma \setminus \{a\}$ of agents, hence by the induction hypothesis

$$\forall s \in S, |\,\mathrm{dom}(s)| \geq (n-1)2^{n-2} \ . \tag{3}$$

Let us consider the directed graph $G = (S, E_\prec)$ with the set S as the set of vertices and the set of edges $E_\prec$ induced by $\prec$: $(s,t) \in E_\prec$ if and only if $s \prec t$. For each $s \in S$ the set of edges of G that have source s has the same cardinality as $\mathrm{dom}(s)$, thus by (3)

$$|E_\prec| = \sum_{s \in S} |\,\mathrm{dom}(s)| \geq |S|(n-1)2^{n-2} \ . \tag{4}$$

Condition (1) implies that the mapping from S into Σ which maps, for $a \in \Sigma$, all time-stamps of S_a to a is a vertex coloring of the graph G. The lower bound on the graph chromatic number given in [1] (Theorem 3 in Chapter 15) states that

$$|\Sigma| \geq |S|^2/(|S|^2 - 2|E_\prec|) \ ,$$

i.e. $|E_\prec| \leq |S|^2(1-1/n)/2$. The last inequality and (4) imply $|S| \geq n2^{n-1}$. $\square$

4 Time-stamps for traces

In this section we show that the time-stamp system constructed in Section 3 can also be used in a distributed environment modeled by Mazurkiewicz traces.

Let us consider the following situation. As previously, we have a fixed finite set of agents Σ, however, now each agent $a \in \Sigma$ has his own box, it is convenient to name the boxes after their owners, thus a denotes an agent as well as his box. The boxes form the vertices of a simple undirected graph that is called the *dependency graph*.

In the formal reasoning it is convenient to use the *dependency relation*. This is a binary symmetric and reflexive relation $\mathscr{D}$ over Σ such that $(a, b) \in \mathscr{D}$ if either $a = b$ or if there is an edge joining a and b in the dependency graph. We say that a and b are *independent* if $(a, b) \notin \mathscr{D}$.

An agent cannot modify the contents of the boxes belonging to other agents but he can read all adjacent boxes and copy the messages he finds there to his box. He is always interested only in the most recent messages sent by the other agents and therefore in his box he stores for each agent c the most recent message sent by c that is available to him, all old messages from c are discarded.

More precisely, an execution of an action by an agent a takes place in three steps:

(1) first a copies all messages from all the adjacent boxes into his box,
(2) next he determines for each agent c, $c \neq a$, the most recent message from c present in his box and discards from his box all the other messages from c,
(3) finally a discards from his box his own old message (if there is any) and replaces it by his new message.

Such an action is assumed to be atomic which implies that the actions of adjacent agents cannot overlap in time.

Any sequence $a_1 a_2 \ldots a_n \in \Sigma^*$ can represent a sequential order of actions executed by the agents in the system, a_i represents an action executed by agent a_i. Elements of Σ, if they appear in such a sequence, are called actions (thus actions inherit their names from the corresponding agents). Now we should observe that changing the order of consecutive actions executed by independent agents has no influence on the system, if a and b are independent, $(a, b) \notin \mathscr{D}$, then for any $u, v \in \Sigma^*$, the sequences $uabv$ and $ubav$ of actions are indistinguishable in this model. More generally, let $\sim_{\mathscr{D}}$ be the smallest equivalence relation over the elements of Σ^* such that $uabv \sim_{\mathscr{D}} ubav$, for all

$u, v \in \Sigma^*$ and $(a, b) \notin \mathscr{D}$, then the action sequences equivalent under $\sim_{\mathscr{D}}$ are indistinguishable in this model. The equivalence classes of elements of Σ^* under $\sim_{\mathscr{D}}$ are known as Mazurkiewicz traces. It is useless to present here all the elementary introductory machinery related to traces, the reader can consult to this end any of the numerous papers using traces or the monograph [5]; in fact the introductory chapter [6] is sufficient for our purposes.

Since the relation $\sim_{\mathscr{D}}$ is a congruence for the operation of concatenation of words, traces form a monoid denoted here $\mathbb{M}(\Sigma, \mathscr{D})$. A trace t_1 is a prefix of a trace t_2, denoted $t_1 \sqsubseteq t_2$, if there exists a trace t_3 such that $t_1 t_3 = t_2$. If $t_3 \neq \mathbf{1}$, where $\mathbf{1}$ is the empty trace, then t_1 is a proper prefix of t_2 and we write then $t_1 \sqsubset t_2$. The relation $\sqsubset$ is a partial order over the set of traces.

For any non-empty set $A \subset \Sigma$ and a trace t, by $\partial_A(t)$ we denote the shortest prefix of t containing all occurrences of actions of A. If A is reduced to just one action a then we write simply $\partial_a(t)$.

Non-empty traces of the form $\partial_a(t)$, for $a \in \Sigma$, are called prime traces and Prime stands for the set of all such traces.

For a trace t, $\mathrm{alph}(t)$ stands for the set of actions (letters) appearing in t.

For any two dependent actions a and b, $(a, b) \in \mathscr{D}$, and any trace t, either $\partial_a(t) \sqsubseteq \partial_b(t)$ or $\partial_b(t) \sqsubseteq \partial_a(t)$. Moreover, if a and b are dependent and different, $(a, b) \in \mathscr{D}$ and $a \neq b$, and if $\mathrm{alph}(t) \cap \{a, b\} \neq \emptyset$ then either $\partial_a(t) \sqsubset \partial_b(t)$ or $\partial_b(t) \sqsubset \partial_a(t)$.

In general, for any trace t and any actions $a, b \in \mathrm{alph}(t)$, $\partial_a(t) \sqsubset \partial_b(t)$ if and only if there exists a sequence $a = c_1, \ldots, c_k = b$ of actions such that, for all i, $i < k$, $(c_i, c_{i+1}) \in \mathscr{D}$ and $\partial_{c_i}(t) \sqsubset \partial_{c_{i+1}}(t)$; note that this implies in particular that all actions c_i are pairwise different.

The proof of the following elementary but extremely useful fact can be found either in [10] or in [6].

Lemma 4 (Levi lemma for traces) *Let $t = xy = zu$ be two factorizations of a trace t. Then there exist traces t_0, t_1, t_2, t_3 such that every action of $\mathrm{alph}(t_1)$ is independent of every action of $\mathrm{alph}(t_2)$ and $x = t_0 t_1$, $y = t_2 t_3$, $z = t_0 t_2$, $u = t_1 t_3$.*

The following lemma gathers some useful facts concerning traces. The elementary proof is left to the reader.

Lemma 5 *(1) If t_1, t_2 are traces such that $t_1 \sqsubseteq t_2$ then for each non-empty subset A of Σ, $\partial_A(t_1) \sqsubseteq \partial_A(t_2)$.*
(2) For every trace t, $a \in \Sigma$ and a non-empty subset B of Σ, if $\partial_a(t) \sqsubseteq \partial_B(t)$

then $\partial_a(t) = \partial_a(\partial_B(t))$.

(3) Let $t \in \mathbb{M}(\Sigma, \mathscr{D})$, A, B *non-empty subsets of* Σ *and*

$$E(t, A, B) = \{c \in \Sigma \mid \partial_c(\partial_A(t)) = \partial_c(\partial_B(t))\} \ . \tag{5}$$

If $\mathbf{1} \neq \partial_a(\partial_A(t)) \sqsubset \partial_a(\partial_B(t))$ *then there exists* $c \in E(t, A, B)$ *such that* $\partial_a(\partial_A(t)) \sqsubset \partial_c(\partial_A(t)) = \partial_c(\partial_B(t)) \sqsubset \partial_a(\partial_B(t))$.

We shall use here the same time-stamp system as in Section 3. Thus we assume that there is a total order $<$ over the set Σ and the set S_a of time-stamps of agent a consists of all mappings f from Σ into a three-element set $\{\perp, 0, 1\}$ such that a is the only element of Σ mapped by f to $\perp$. As previously, agent a stamps his messages using the stamps from S_a. The only significant difference is that now we shall use directly the order $\prec$ over stamps only to compare time-stamps of two agents that are adjacent in the dependency graph. Determining the order of messages that do not come from adjacent agents needs more complicated methods.

We first define our stamping system globally and next we show how time-stamps can be used if we have at our disposal only local information.

Let $\mathbf{1}_\Sigma$ be the constant mapping that maps each $a \in \Sigma$ to 1.

We define by induction on the length of traces the mapping

$$\lambda : \mathrm{Prime} \cup \{\mathbf{1}\} \to S \cup \{\mathbf{1}_\Sigma\} \ .$$

For the empty trace $\mathbf{1}$, $\lambda(\mathbf{1}) = \mathbf{1}_\Sigma$.

Let Prime_a be the set of prime traces of the form $\partial_a(t)$ for some trace t. Thus Prime_a consists of the traces where the last action is executed by a and for $t \in \mathrm{Prime}_a$, $\lambda(t)$ will be the time-stamp associated with this occurrence of action a.

For $t \in \mathrm{Prime}_a$, $\lambda(t)$ is a time-stamp of S_a defined in the following way:

$$\lambda(t)(b) = \begin{cases} \perp & \text{if } b = a, \\ \lambda(\partial_b(t))(a) & \text{if } b < a, \\ 1 - \lambda(\partial_b(t))(a) & \text{if } a < b \ . \end{cases} \tag{6}$$

Note that λ is really well-defined since for every trace $t \in \mathrm{Prime}_a$ and every $b \neq a$, $\partial_b(t)$ is a proper prefix of t belonging to $\mathrm{Prime}_b \cup \{\mathbf{1}\}$.

The inductive formula (6) was conceived to guarantee the most important feature of λ: for $t \in \mathrm{Prime}_a$ and every $b \in \mathrm{alph}(t) \setminus \{a\}$, $\lambda(\partial_b(t)) \prec \lambda(t)$. In

particular,

$$\forall t \in \mathbb{M}(\Sigma, \mathscr{D}), \forall a, b \in \Sigma,$$
$$\text{if } b \in \mathrm{alph}(\partial_a(t)) \text{ and } a \neq b \text{ then } \lambda(\partial_b(\partial_a(t))) \prec \lambda(\partial_a(t)) \ . \quad (7)$$

Lemma 6 *For each trace $t \in \mathbb{M}(\Sigma, \mathscr{D})$ and all $a, b \in \mathrm{alph}(t)$, if*

$$\partial_a(t) \sqsubset \partial_b(t) \quad (8)$$

then $\lambda(\partial_a(t)) \prec \lambda(\partial_b(t))$.

PROOF. Since $a, b \in \mathrm{alph}(t)$ the traces $\partial_a(t)$ and $\partial_b(t)$ are non-empty and, by Lemma 5, eq. (8) implies $\partial_a(t) = \partial_a(\partial_b(t))$. Eq. (8) implies also that $a \neq b$ therefore (7) applies yielding $\lambda(\partial_a(t)) = \lambda(\partial_a(\partial_b(t))) \prec \lambda(\partial_b(t))$. $\square$

The preceding lemma can be strengthen in the case of dependent actions:

Lemma 7 *For each trace $t \in \mathbb{M}(\Sigma, \mathscr{D})$ and all actions $a, b \in \mathrm{alph}(t)$ such that $(a, b) \in \mathscr{D}$ and $a \neq b$ we have $\partial_a(t) \sqsubset \partial_b(t)$ if and only if $\lambda(\partial_a(t)) \prec \lambda(\partial_b(t))$.*

PROOF. Since a and b are different dependent actions, the traces $\partial_a(t)$ and $\partial_b(t)$ are ordered by the strict prefix relation $\sqsubset$. By Lemma 6, if $\partial_b(t) \sqsubset \partial_a(t)$ then $\lambda(\partial_b(t)) \prec \lambda(\partial_a(t))$ and if $\partial_a(t) \sqsubset \partial_b(t)$ then $\lambda(\partial_a(t)) \prec \lambda(\partial_b(t))$. Since the two-element time-stamp set $\{\lambda(\partial_a(t)), \lambda(\partial_b(t))\}$ containing time-stamps of different agents is totally ordered by $\prec$ relation, the thesis follows. $\square$

Proposition 8 *Let $t \in \mathbb{M}(\Sigma, \mathscr{D})$, $a \in \Sigma$, A, B two non-empty subsets of Σ such that $\partial_a(\partial_A(t)) \neq \mathbf{1} \neq \partial_a(\partial_B(t))$. Then $\partial_a(\partial_A(t)) = \partial_a(\partial_B(t))$ if and only if $\lambda(\partial_a(\partial_A(t))) = \lambda(\partial_a(\partial_B(t)))$.*

PROOF. The left to right implication follows just from the definition of λ.

Suppose that $\partial_a(\partial_A(t))) \neq \partial_a(\partial_B(t)))$. Since the traces $\partial_a(\partial_A(t)))$ and $\partial_a(\partial_B(t)))$ are comparable by the prefix relation without loss of generality we can assume that $\partial_a(\partial_A(t))) \sqsubset \partial_a(\partial_B(t)))$. By Lemma 5 there exists $c \in \Sigma$ such that $\partial_a(\partial_A(t))) \sqsubset \partial_c(\partial_A(t))) = \partial_c(\partial_B(t))) \sqsubset \partial_a(\partial_B(t)))$. Then, by Lemma 6, $\lambda(\partial_a(\partial_A(t)))) \prec \lambda(\partial_c(\partial_A(t)))) = \lambda(\partial_c(\partial_B(t)))) \prec \lambda(\partial_a(\partial_B(t))))$, i.e. $\lambda(\partial_a(\partial_A(t)))) \neq \lambda(\partial_a(\partial_B(t))))$. $\square$

With each trace $t \in \mathbb{M}(\Sigma, \mathscr{D})$ we associate the *time-stamp set* $\Lambda(t)$ of t:

$$\Lambda(t) = \{\lambda(\partial_a(t)) \mid a \in \mathrm{alph}(t)\} \ . \quad (9)$$

The time-stamp set $\Lambda(t)$ induces the following directed *precedence graph* $\mathbf{G}(t)$ of t. The vertices of $\mathbf{G}(t)$ are those actions $a \in \Sigma$ for which there exists a time-stamp $f \in \Lambda(t)$ belonging to the agent a, i.e. such that $f(a) = \perp$. Note that from the definition of $\Lambda(t)$ it follows that the set of vertices of $\mathbf{G}(t)$ is equal to the alphabet $\mathrm{alph}(t)$ of t. The set E_t of edges of $\mathbf{G}(t)$ is constructed in the following way: for $a, b \in \mathrm{alph}(t)$, $(a, b) \in E_t$ if and only if a and b are different dependent actions, $(a, b) \in \mathscr{D}$ and $a \neq b$, and for the time-stamps $f, g \in \Lambda(t)$ such that $f(a) = \perp = g(b)$ we have $f \prec g$.

Note that for each pair of different dependent actions $a, b \in \mathrm{alph}(t)$, $(a, b) \in \mathscr{D}$, $a \neq b$, always one of the traces $\partial_a(t)$ and $\partial_b(t)$ is a prefix of the other and by Lemma 7 this prefix order is captured by the edges of $\mathbf{G}(t)$, i.e. there is an edge from a to b in $\mathbf{G}(t)$ iff $\mathbf{1} \neq \partial_a(t) \sqsubset \partial_b(t)$.

Definition 9 *Let $t \in \mathbb{M}(\Sigma, \mathscr{D})$ be a trace, $a \in \Sigma$, A, B non-empty subsets of Σ. We define*

$$\mathrm{Eq}(t, A, B) = \{c \in \Sigma \mid \exists f \in \Lambda(\partial_A(t)) \cap \Lambda(\partial_B(t)), f(c) = \perp\} \ . \tag{10}$$

In other words, an action $c \in \Sigma$ belongs to $\mathrm{Eq}(t, A, B)$ if and only if both time-stamp sets $\Lambda(\partial_A(t))$ and $\Lambda(\partial_B(t))$ contain the same time-stamp issued by agent c.

We define also

$$\mathrm{Lt}(t, A, B) \tag{11}$$

to be the set consisting of all actions $c \in \mathrm{alph}(\partial_B(t))$ such that there exists an directed path in the precedence graph $\mathbf{G}(\partial_B(t))$ from c to an element of B which does not pass by any element of $\mathrm{Eq}(t, A, B)$.

Proposition 10 *Let $t \in \mathbb{M}(\Sigma, \mathscr{D})$ and A, B non-empty subsets of Σ.*

Then for any action $c \in \Sigma$,

(A) c belongs to $\mathrm{alph}(\partial_{A\cup B}(t))$ if and only if c is a vertex in one of the precedence graphs $\mathbf{G}(\partial_A(t))$ or $\mathbf{G}(\partial_B(t))$,

(B) for all $c \in \mathrm{alph}(\partial_{A\cup B}(t))$, exactly one of the following cases holds:

(i) $\partial_c(\partial_{A\cup B}(t)) = \partial_c(\partial_A(t)) = \partial_c(\partial_B(t))$ if and only if c belongs to $\mathrm{Eq}(t, A, B)$,

(ii) $\partial_c(\partial_A(t)) \sqsubset \partial_c(\partial_B(t)) = \partial_c(\partial_{A\cup B}(t))$ if and only if $c \in \mathrm{Lt}(t, A, B)$,

(iii) $\partial_c(\partial_B(t)) \sqsubset \partial_c(\partial_A(t)) = \partial_c(\partial_{A\cup B}(t))$ if and only if $c \in \mathrm{Lt}(t, B, A)$.

(C) There exists an algorithm that, given A, B, $\Lambda(\partial_A(t))$ and $\Lambda(\partial_B(t))$, calculates $\Lambda(\partial_{A\cup B}(t))$.

PROOF. (A) follows directly from the fact that $\mathrm{alph}(\partial_{A\cup B}(t)) = \mathrm{alph}(\partial_A(t)) \cup$

$\mathrm{alph}(\partial_B(t))$ and the sets of vertices of precedence graphs are the alphabets of the corresponding traces.

(B) Part (i) was in fact already proved in Proposition 8.

To prove (ii) let us first factorize traces $\partial_A(t) = t_0 t_A$ and $\partial_B(t) = t_0 t_B$, where t_0 is the longest common prefix of $\partial_A(t)$ and $\partial_B(t)$. Then every action of $\mathrm{alph}(t_A)$ is independent of every action of $\mathrm{alph}(t_B)$ and moreover $\partial_{A\cup B}(t) = t_0 t_A t_B$.

Suppose now that $\partial_c(\partial_A(t)) \sqsubset \partial_c(\partial_B(t))$. Then $\partial_c(\partial_B(t)) = \partial_c(\partial_{A\cup B}(t))$. Moreover $c \in \mathrm{alph}(t_B)$ and there exists a sequence $c = c_0, c_1, \ldots, c_k \in B$ of actions such that $\forall i, 0 \leq i < k$, $(c_i, c_{i+1}) \in \mathscr{D}$, $c_i \neq c_{i+1}$ and $\partial_{c_i}(\partial_B(t)) \sqsubset \partial_{c_{i+1}}(\partial_B(t))$. All actions c_i in this sequence belong to $\mathrm{alph}(t_B)$ and therefore always $\partial_{c_i}(\partial_A(t)) \sqsubset \partial_{c_i}(\partial_B(t))$ and (i) and Lemma 7 show that the sequence of c_i defined above constitutes the path required in Definition 9 of $\mathrm{Lt}(t, A, B)$.

Conversely, let us suppose now that $c \in \mathrm{Lt}(t, A, B)$. Then, by the definition, there exists an directed path $c = c_0, \ldots, c_k \in B$ in the graph $\mathbf{G}(\partial_B(t))$ such that no c_i is in $\mathrm{Eq}(t, A, B)$. By backward induction we shall prove that $\forall i, 0 \leq i \leq k$,

$$\partial_{c_i}(\partial_A(t)) \sqsubset \partial_{c_i}(\partial_B(t)) \ , \tag{12}$$

which, for $i = 0$, provides the required result.

First note that since $c_k \in B$, $\partial_{c_k}(\partial_A(t)) \sqsubseteq \partial_{c_k}(\partial_{A\cup B}(t)) = \partial_{c_k}(\partial_B(t))$. Since c_k is not in $\mathrm{Eq}(t, A, B)$ by (i) the traces $\partial_{c_k}(\partial_A(t))$ and $\partial_{c_k}(\partial_B(t))$ cannot be equal and therefore (12) holds for $i = k$.

Suppose that (12) holds for $i + 1$ and we shall prove that it holds for i. Since (c_i, c_{i+1}) is an edge of $\mathbf{G}(\partial_B(t))$, we have $\lambda(\partial_{c_i}(\partial_B(t))) \prec \lambda(\partial_{c_{i+1}}(\partial_B(t)))$. However, as $(c_i, c_{i+1}) \in \mathscr{D}$ and $c_i \neq c_{i+1}$ this implies by Lemma 7 that $\partial_{c_i}(\partial_B(t)) \sqsubset \partial_{c_{i+1}}(\partial_B(t))$.

Again since $(c_i, c_{i+1}) \in \mathscr{D}$ and $c_i \neq c_{i+1}$, prime traces $\partial_{c_i}(\partial_A(t))$ and $\partial_{c_{i+1}}(\partial_B(t))$ are comparable by the prefix order. It is impossible to have $\partial_{c_{i+1}}(\partial_B(t)) \sqsubset \partial_{c_i}(\partial_A(t))$ since this would mean that the prime trace $\partial_{c_{i+1}}(\partial_B(t))$ is a prefix of $\partial_A(t)$ implying $\partial_{c_{i+1}}(\partial_B(t)) \sqsubseteq \partial_{c_{i+1}}(\partial_A(t))$, in contradiction with the induction hypothesis requiring that (12) holds for $i + 1$.

Thus $\partial_{c_i}(\partial_A(t)) \sqsubset \partial_{c_{i+1}}(\partial_B(t))$, in particular $\partial_{c_i}(\partial_A(t))$ is a prefix of $\partial_B(t)$. This implies that $\partial_{c_i}(\partial_A(t)) \sqsubseteq \partial_{c_i}(\partial_B(t))$. However, the equality is excluded by the fact that $c_i \notin \mathrm{Eq}(t, A, B)$ and by (i). Thus $\partial_{c_i}(\partial_A(t))$ is a proper prefix of $\partial_{c_i}(\partial_B(t))$ and (12) holds for i. This terminates the proof of (ii).

The last case (iii) is just symmetric to (ii).

Finally, since $\partial_c(\partial_A(t))$ and $\partial_c(\partial_B(t))$ are comparable by the prefix order no

case other than the ones listed in (B) is possible.

(C) Given $\Lambda(\partial_A(t))$ and $\Lambda(\partial_B(t))$ we can, directly from the definition, calculate the set $\mathrm{Eq}(t, A, B)$ as well as find the graphs $\mathbf{G}(\partial_A(t))$ and $\mathbf{G}(\partial_B(t))$ (note that to find both graphs we need also to know the dependency relation $\mathscr{D}$). In the next step, again using the definition, we find $\mathrm{Lt}(t, A, B)$ and $\mathrm{Lt}(t, B, A)$. And finally, (A) and (B) show that $\Lambda(\partial_{A\cup B}(t))$ is correctly calculated by the following algorithm:

- for each action $a \in \mathrm{Lt}(t, A, B)$ pick from $\Lambda(\partial_B(t))$ the time-stamp f such that $f(a) = \bot$ and put it into $\Lambda(\partial_{A\cup B}(t))$,
- for each action $a \in \mathrm{Lt}(t, B, A)$ pick from $\Lambda(\partial_A(t))$ the time-stamp f such that $f(a) = \bot$ and put it into $\Lambda(\partial_{A\cup B}(t))$,
- for each action $a \in \mathrm{Eq}(t, A, B)$ we can pick indifferently either from $\Lambda(\partial_A(t))$ or from $\Lambda(\partial_B(t))$ the time-stamp f such that $f(a) = \bot$ and put in into $\Lambda(\partial_{A\cup B}(t))$.

□

Proposition 11 *Let $t \in \mathbb{M}(\Sigma, \mathscr{D})$ and $a \in \Sigma$. There is an algorithm that, given $\Lambda(\partial_b(t))$ for all b such that $(a, b) \in \mathscr{D}$, calculates $\lambda(\partial_a(ta))$.*

PROOF. Let $A = \{b \in \Sigma \mid (a, b) \in \mathscr{D}\}$ be the set of all actions dependent on a. Suppose that $A = \{b_1, \ldots, b_m\}$ and set $A_i = \{b_1, \ldots, b_i\}$, $i = 1, \ldots, m$. The algorithm given in the proof of Proposition 10 (C) allows to calculate $\Lambda(\partial_{A_{i+1}}(t))$ from $\Lambda(\partial_{A_i}(t))$ and $\Lambda(\partial_{b_{i+1}}(t))$, therefore applying it $m - 1$ times we can calculate $\Lambda(\partial_A(t))$.

Note that $\partial_a(ta) = \partial_A(t)a$ which implies that for all $b \neq a$, $\partial_b(\partial_a(ta)) = \partial_b(\partial_A(t))$. Therefore for such actions b, $\lambda(\partial_b(\partial_a(ta))) = \lambda(\partial_b(\partial_A(t)))$. However, $\lambda(\partial_b(\partial_A(t)))$ is just the time-stamp g in $\Lambda(\partial_A(t))$ such that $g(b) = \bot$.

This shows that we can calculate the time-stamp $f = \lambda(\partial_a(ta)) \in S_a$ in the following way: for all $b \in \Sigma$,

$$f(b) = \begin{cases} \bot & \text{if } b = a, \\ g(a) & \text{if } b < a \text{ and there is } g \in \Lambda(\partial_A(t)) \text{ such that } g(b) = \bot, \\ 1 - g(a) & \text{if } a < b \text{ and there is } g \in \Lambda(\partial_A(t)) \text{ such that } g(b) = \bot, \\ \mathbf{1}_\Sigma(a) & \text{if } b < a \text{ and there is no } g \in \Lambda(\partial_B(t)) \text{ such that } g(b) = \bot, \\ 1 - \mathbf{1}_\Sigma(a) & \text{if } a < b \text{ and there is no } g \in \Lambda(\partial_B(t)) \text{ such that } g(b) = \bot \ . \end{cases}$$

The last two cases in the definition of f above are just the consequences of the fact that if there is no $g \in \Lambda(\partial_A(t))$ such that $g(b) = \bot$ then $\partial_b(\partial_a(ta)) = \mathbf{1}$ and for the empty trace we have set in (6) $\lambda(\mathbf{1}) = \mathbf{1}_\Sigma$. □

References

[1] C. Berge. *Graphs and hypergraphs*. North-Holland, 1976.

[2] R. Cori, Y. Métivier, and W. Zielonka. Asynchronous mappings and asynchronous cellular automata. *Information and Computation*, 106(2):159–202, 1993.

[3] R. Cori and E. Sopena. Some combinatorial aspect of time-stamp systems. *Europ. J. Combinatoric*, 14:95–102, 1993.

[4] V. Diekert and A. Muscholl. A note on Métivier's construction of asynchronous automata for triangulated graphs. *Fundamenta Informatica*, 25:241–246, 1996.

[5] V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, 1995.

[6] A. Mazurkiewicz. Introduction to trace theory. In V. Diekert and G. Rozenberg, editors, *The Book of Traces*, pages 3–41. World Scientific, 1995.

[7] Y. Métivier. An algorithm for computing asynchronous automata in the case of acyclic non-commutation graph. In *ICALP'87*, volume 267 of *LNCS*, pages 226–236. Springer, 1987.

[8] M. Mukund and M. Sohoni. Keeping track of the latest gossip in a distributed system. *Distributed Computing*, 10(3):137–148, 1997.

[9] G. Pighizzini. Synthesis of nondeterministic asynchronous automata. In M. Droste and Y.Gurevich, editors, *Semantics of programming languages and model theory*, volume 5 of *Logic and Appplications*, pages 109–126. Gordon and Breach Science Publ., 1993.

[10] R.Cori and D.Perrin. Automates et commutations partielles. *RAIRO Informatique théorique et applications*, 19:21–32, 1985.

[11] W. Zielonka. Notes on finite asynchronous automata. *RAIRO Informatique thorique et applications*, 21(2):99–135, 1987.