Chapter 8

Construction of Asynchronous Automata

Volker Diekert Anca Muscholl

Universität Stuttgart, Institut für Informatik
Breitwiesenstr. 20–22, 70565 Stuttgart, Germany
{diekert,muscholl}@informatik.uni-stuttgart.de

Contents

8.1 Introduction

In this chapter we consider recognizable trace languages from the viewpoint of automata-theoretical characterizations. We use the asynchronous (cellular) automaton, which is characterized by a distributed control structure. A detailed investigation about these automata and their control structures is given in Chapter 7.

By Zielonka's theorem [277, 278], both, deterministic asynchronous automata (an *exclusive-read-exclusive-write* model) and deterministic cellular asynchronous automata (a *concurrent-read-owner-write* model) characterize recognizability of trace languages precisely. This characterization is in fact one of the major contributions in the theory of Mazurkiewicz traces, due to the non-interleaving semantics of asynchronous (cellular) automata.

Zielonka's construction starts with a trace language recognized by a homomorphism to a finite monoid, and it yields a deterministic asynchronous cellular automaton. Its basic component is a bounded time-stamping which allows to determine the actuality of information received in a distributed way.

A crucial aspect of Zielonka's construction is given however by the notion of asynchronous mapping, which will be presented in Section 8.3.2. Asynchronous mappings, originally introduced by R. Cori and Y. Métivier [45] (see also [53]), are mappings which can be computed in a distributed manner. This property makes them directly translatable into deterministic asynchronous cellular automata. In Zielonka's original construction asynchronous mappings appear implicitly, only. Once one has constructed a deterministic asynchronous cellular automaton, the translation to an asynchronous automaton (as originally defined by Zielonka) is straightforward. The difference between both models is of minor importance here. The polynomial-time transformations from asynchronous cellular automata into equivalent asynchronous automata and vice-versa have been already discussed in Chapter 7, see also e.g. [227, 229].

The question whether a substantially simpler construction for Zielonka's theorem exists, is still open. Recently, two determinization constructions for asynchronous (cellular) automata have been presented [154, 198], both still relying on Zielonka's time-stamping function. So far, only in the special case of acyclic dependence graphs a simpler solution is known: in [196] Y. Métivier exhibited a surprisingly elegant construction which provides in a natural way deterministic asynchronous automata. The drawback of this construction is the fact that the hypothesis of an acyclic dependence relation is a strong restriction, which is often violated. In fact, even the special case of complete dependence, i.e. the free monoid, is not included. Following the presentation of [63] we generalize in the next section Métivier's construction to the larger class of *triangulated* graphs. From a practical point of view, this means that by adding sufficient dependence, i.e. triangulating in a dependence alphabet, we can obtain simple deterministic asynchronous automata for every recognizable trace languages, by loosing some concurrency, only.

We conclude this section by fixing the notations used in this chapter, which are in fact the standard ones.

By (Σ, D) we denote a finite *dependence alphabet*, with Σ being a finite alphabet and $D \subseteq \Sigma \times \Sigma$ a reflexive and symmetric relation called *dependence relation*. The complementary relation $I = (\Sigma \times \Sigma) \setminus D$ is called *independence relation*. The monoid of *finite traces*, $\mathbb{M}(\Sigma, D)$, is defined as a quotient monoid with respect to the congruence relation induced by I, i.e., $\mathbb{M}(\Sigma, D) = \Sigma^* / \{ ab = ba \mid (a, b) \in I \}$.

Recognizability for trace languages can be defined by recognizing homomor-

phisms: a trace language $L \subseteq \mathbb{M}(\Sigma, D)$ is *recognizable* if there exists a finite monoid S and a homomorphism $\eta : \mathbb{M}(\Sigma, D) \to S$ recognizing L, i.e. satisfying $L = \eta^{-1}\eta(L)$. Alternatively, by Zielonka's characterization theorem, recognizability can be defined by means of asynchronous (cellular) automata. (We shall work here with deterministic automata, only.) We denote the family of recognizable trace languages by $\mathrm{Rec}(\mathbb{M}(\Sigma, D))$ (or $\mathrm{Rec}(\mathbb{M})$ for short).

Definition 8.1.1 *An* asynchronous automaton $\mathcal{A}$ *is a tuple*

$$\mathcal{A} = (\prod_{i=1}^{m} Q_i, (\delta_a)_{a\in\Sigma}, q_0, F)$$

satisfying the following conditions:

- *The global state set* $Q = \prod_{i=1}^{m} Q_i$ *is a direct product of local states sets* Q_i, $1 \leq i \leq m$.
- *Each letter* $a \in \Sigma$ *has an associated domain* $dom(a) \subseteq \{1, \ldots, m\}$ *such that for all* $(a, b) \in I$ *we have* $dom(a) \cap dom(b) = \emptyset$.
- *Each letter* $a \in \Sigma$ *has a (partially defined) local transition function* $\delta_a : \prod_{i\in \mathrm{dom}(a)} Q_i \to \prod_{i\in \mathrm{dom}(a)} Q_i$. *A global transition step* $q' = \delta(q, a)$, $q = (q_i)_{1\leq i\leq m}, q' = (q_i')_{1\leq i\leq m} \in Q$, *is defined if and only if* $\delta_a((q_i)_{i\in \mathrm{dom}(a)})$ *is defined. In this case only the components of the domain of a change, i.e. we have*
  1. $q_j' = q_j$ *if* $j \notin dom(a)$
  2. $(q_i')_{i\in \mathrm{dom}(a)} = \delta_a((q_i)_{i\in \mathrm{dom}(a)})$

The language accepted by $\mathcal{A}$ *is* $L(\mathcal{A}) = \{t \in \mathbb{M}(\Sigma, D) \mid \delta(q_0, t) \in F\}$.

Note that asynchronous automata enable concurrent transitions, since write- and read-domains of independent letters do not overlap. They can be seen as EREW (*exclusive-read-exclusive-write*) devices, whereas asynchronous cellular automata, which are defined in the following, correspond to the CROW (*concurrent-read-owner-write*) type.

Definition 8.1.2 *An* asynchronous cellular automaton $\mathcal{A}$ *is a tuple*

$$\mathcal{A} = ((Q_a)_{a\in\Sigma}, (\delta_a)_{a\in\Sigma}, q_0, F)$$

For each letter $a \in \Sigma$ *a set of local states* Q_a *and a (partially defined) local transition function* $\delta_a : (\prod_{b\in D(a)} Q_b) \to Q_a$ *are given. Further,* $q_0 \in \prod_{a\in\Sigma} Q_a$ *denotes the global initial state and* $F \subseteq \prod_{a\in\Sigma} Q_a$ *denotes the set of final global states. The (partially defined) global transition function* $\delta : \prod_{a\in\Sigma} Q_a \times \Sigma \to \prod_{a\in\Sigma} Q_a$ *is defined by*

$$q' = \delta(q, a) \quad \Leftrightarrow \quad \begin{array}{ll} q_a' = \delta_a((q_b)_{b\in D(a)}) & \text{and} \\ q_c{}' = q_c, & \text{for all } c \neq a \end{array}$$

Thus, an a-transition $q' = \delta(q, a)$ is defined if and only if $\delta_a((q_b)_{b \in D(a)})$ is defined; and in this case, the change affects only the local a-state. The accepted language is defined as above.

8.2 Métivier's Construction for Triangulated Dependence Graphs

Before giving the proof of Zielonka's theorem, which turns out to be rather technical, we consider here the much easier case of triangulated dependence graphs. We show that Métivier's construction can be extended from acyclic graphs to this case and that triangulated graphs represent the maximal graph class where this construction may be applied to.

The aim of this section is to give a flavour of the behaviour of asynchronous automata and of the problems the general solution has to deal with, without the necessity of using the bounded time-stamping.

We also think that this section may serve as the basis for a course on distributed automata, where the important aspect of asynchronous automata can be presented without having the time to present the general solution.

Let us start with the paradigm behind Métivier's construction. The idea is that each component of the state set is associated with a single letter of the alphabet. Thus, we consider $Q = \prod_{a \in \Sigma} Q_a$ as set of global states. The communication between different components is realized as usual by defining for each $a \in \Sigma$ a set $\mathrm{dom}(a) \subseteq \Sigma$ such that $(a, b) \in I$ implies $\mathrm{dom}(a) \cap \mathrm{dom}(b) = \emptyset$. The main idea is to use a suitable linear ordering on the alphabet, i.e. we have $\Sigma = \{a_1, \ldots, a_n\}$ and $a_1 < \ldots < a_n$. We now transmit the information between components in a fixed, directed way, by letting $\mathrm{dom}(a) = \{b \in D(a) \mid a \leq b\}$. The natural question is to determine the class of dependence graphs where this approach yields an asynchronous automaton.

8.2.1 Triangulated Graphs

An undirected graph $G = (V, E)$ with vertex set V and edge set E is *triangulated* if all its chord-less cycles are of length three. Trivial examples are complete graphs (i.e. graphs where $E = \binom{V}{2}$), or acyclic graphs.

For $u \in V$ let $E(u)$ denote the set of vertices adjacent to u, i.e. $E(u) = \{v \in V \mid uv \in E\}$. A *perfect vertex elimination scheme* is a linear ordering $\leq$ of the vertices such that for all $u \in V$ the set $\{v \in E(u) \mid u \leq v\}$ forms a clique (i.e. a complete subgraph). We may represent a perfect vertex elimination scheme by a list $[v_1, \ldots, v_n]$ such that $v_i \leq v_j$ if and only if $i \leq j$.

By induction on n it is easy to see that a graph having a perfect vertex elimination scheme is triangulated. The converse is also true. This characterization of triangulated graphs is due to G. A. Dirac [67]. The result can be found e.g. in the textbook of M. C. Golumbic [125][Thm. 4.1]. For convenience we sketch the proof:

let $G = (V, E)$ be a triangulated graph. If G is a complete graph, then any ordering of the vertex set is a perfect vertex elimination scheme. Otherwise, let us prove by induction that there are at least two vertices $c, d \in V$ such that $cd \notin E$ and $E(c)$ as well as $E(d)$ are cliques.

Start with two vertices $a, b \in V$ with $ab \notin E$. Let S be some minimal separator of a, b. This is any subset $S \subseteq V$ of minimal cardinality with the property that removing S from G puts a and b in two different connected components C_a and C_b respectively. Since G is triangulated, an easy reflection shows that S is a (possibly empty) clique.

Let G_a (G_b respectively) be the induced subgraph of G by $C_a \cup S$ ($C_b \cup S$ respectively). Either G_a (G_b respectively) is a complete graph or we apply the induction hypothesis in order to obtain a vertex c of G_a (d of G_b respectively), which does not belong to S and such that the set of vertices adjacent to c (d respectively) is a clique in G_a (G_b respectively). Note that all neighbors of c (d respectively) are contained in G_a (G_b respectively). Hence, $E(c)$ and $E(d)$ are both cliques of G. Since $c, d \notin S$ we also have $cd \notin E$ and the induction step follows. This completes the proof of Dirac's result.

If $G = (V, E)$ is acyclic, then any ordering which represents a topological sorting yields a perfect vertex elimination scheme. For a complete graph every order is a perfect vertex elimination scheme. Hence, the construction given below will apply to acyclic graphs and complete graphs as well.

8.2.2 Métivier's Construction

In [196] Y. Métivier gave an elegant and amazingly simple construction for asynchronous automata in the case of acyclic dependence alphabets (Σ, D) (see also the survey of D. Perrin [223]). The basic idea is to choose a tree (forest) structure and to move the distributed information to the top of the tree(s) as soon as available. We will show here that the construction applies to triangulated graphs, too. All we need is a perfect vertex elimination scheme, as defined in the previous subsection.

Let (Σ, D) be any triangulated dependence alphabet and let $\leq$ be a linear ordering of Σ such that for all $a, b, c \in \Sigma$, $a \leq b \leq c$, $(a, c) \in D$ and $(b, c) \in D$, we have $(a, b) \in D$, too. Thus, if $\Sigma = \{a_1, \ldots, a_n\}$ ($n = |\Sigma|$) is written in increasing order $a_1 < \cdots < a_n$, then $[a_n, \ldots, a_1]$ is a perfect vertex elimination scheme of (Σ, D).

Let $\eta : \mathbb{M}(\Sigma, D) \to S$ be a homomorphism to a finite monoid S and $L = \eta^{-1}(R)$ for some subset $R \subseteq S$. We are going to construct an asynchronous automaton $\mathcal{A}$ recognizing L.

The state set of $\mathcal{A}$ will be an n-fold direct product of S, $n = |\Sigma|$. First we define ordered products in S. Let $\Gamma \subseteq \Sigma$ and $s_c \in S$, for all $c \in \Gamma$. Define the ordered product $\prod_{c \in \Gamma} s_c$ by $\prod_{c \in \Gamma} s_c = s_{c_1} \cdots s_{c_m}$, if $\Gamma = \{c_1, \ldots, c_m\}$, $m \geq 0$, with $c_1 < \cdots < c_m$. All products of elements of S used in the following will be ordered ones.

Let now $\mathcal{A} = (\prod_{a\in\Sigma} Q_a, \delta, q_0, F)$ be defined by

$$\delta : \prod_{a\in\Sigma} Q_a \times \Sigma \to \prod_{a\in\Sigma} Q_a,$$

where for every $a \in \Sigma$:

$$Q_a = S$$

$$\delta(q,a) = q \cdot a = q' \quad \text{with} \quad {q_b}' = \begin{cases} (\prod\limits_{\substack{a\le c,\\ (a,c)\in D}} q_c) \cdot \eta(a) & \text{if } b = a \\ 1 & \text{if } a < b,\ (a,b) \in D \\ q_b & \text{otherwise} \end{cases}$$

Furthermore, let $q_0 = (1)_{a\in\Sigma}$ and $F = \{(q_a)_{a\in\Sigma} \in \prod_{a\in\Sigma} Q_a \mid \prod\limits_{a\in\Sigma} q_a \in \eta(L)\}$.

Proposition 8.2.1 *Let $L \in \mathrm{Rec}(\mathbb{M})$ be recognized by the homomorphism $\eta : \mathbb{M}(\Sigma, D) \to S$ to the finite monoid S. Let $\mathcal{A}$ be the automaton defined above. Then $\mathcal{A}$ is asynchronous and we have $L(\mathcal{A}) = L$.*

Proof: First we show that $\mathcal{A}$ is asynchronous. The write- and read-domain of a letter $a \in \Sigma$ is given by the index set $\{c \in \Sigma \mid a \le c \text{ and } (a,c) \in D\}$. Assume that for some $a, b, c \in \Sigma$, $a \le b \le c$ we have $(a,c) \in D$ and $(b,c) \in D$. Then $(a,b) \in D$ follows, due to the ordering. Hence, if $(a,b) \in I$, then a and b have disjoint write- and read-domains. Thus, $\mathcal{A}$ is asynchronous. In particular, for any trace $w \in \mathbb{M}(\Sigma, D)$, the global state $q_0 \cdot w = \delta((1)_{a\in\Sigma}, w)$ is well-defined.

For $(q_a)_{a\in\Sigma} = q_0 \cdot w$ we show the following two invariants:

1. $a \le b$ and $(a,b) \in I$ imply $q_b\, \eta(a) = \eta(a)\, q_b$.
2. $a \le b \le c$, $(a,b) \in I$, and $(a,c) \in D$ imply $q_b\, q_c = q_c\, q_b$.

In order to verify these invariants we need some alphabetic information. We observe that for each local state q_a, $a \in \Sigma$, there exists some subset $\Gamma_a \subseteq \Sigma$ with the following properties: q_a is an element of the submonoid generated by $\eta(c)$, with $c \in \Gamma_a$, and for each $c \in \Gamma_a$ there exists a chain $a = a_0 < \cdots < a_k = c$, $k \ge 0$, such that $(a_{i-1}, a_i) \in D$ for $1 \le i \le k$. This crucial observation can be easily derived from the inductive definition of the transition function δ.

Consider now $a \le b \le c$ with $(a,b) \in I$ and $\Gamma_b, \Gamma_c \subseteq \Sigma$ as introduced above. Invariant (1) follows by showing that $\{a\} \times \Gamma_b \subseteq I$. Invariant (2) follows by showing that $\Gamma_b \times \Gamma_c \subseteq I$, whenever $(a,c) \in D$.

For invariant (1) let $b_k \in \Gamma_b$ and $b = b_0 < \cdots < b_k$, $k \ge 0$, such that $(b_{i-1}, b_i) \in D$ for $1 \le i \le k$. Assume that we would have $(a, b_k) \in D$. Then $k \ge 1$ and $(b_{k-1}, b_k) \in D$ implies $(a, b_{k-1}) \in D$. We can continue this way and we arrive to $(a, b_0) \in D$ (see also Figure 8.1). This is a contradiction to $(a,b) \in I$. Thus, $(a, b_i) \in I$ for all $1 \le i \le k$; this fact will be used below, too.

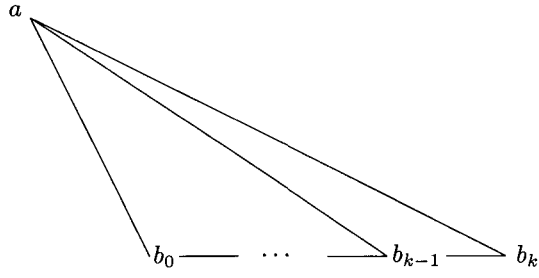


Figure 8.1: Consequence of $(a, b_k) \in D$.

For invariant (2) let $(a, c) \in D$, $b_k \in \Gamma_b$, $c_m \in \Gamma_c$ with $b = b_0 < \cdots < b_k$ and $c = c_0 < \cdots < c_m$, $k, m \geq 0$, such that $(b_{i-1}, b_i) \in D$ and $(c_{j-1}, c_j) \in D$ for $1 \leq i \leq k$, $1 \leq j \leq m$. We show by induction that $(b_i, c_j) \in I$ for all i, j. Assume first that $c_m \leq b_k$. Then we have $k \geq 1$ (since $b < c$) and $(c_m, b_k) \in D$, $(b_{k-1}, b_k) \in D$ imply $(c_m, b_{k-1}) \in D$. Hence we may assume $b_k < c_m$. Suppose next that $m \geq 1$. Then $(b_k, c_m) \in D$, $(c_{m-1}, c_m) \in D$ imply $(b_k, c_{m-1}) \in D$. Hence, we may assume $m = 0$ and we obtain the following situation:

$$a < b = b_0 < \cdots < b_k < c, \quad k \geq 0.$$

Finally, suppose that we would have $(b_i, c) \in D$ for some $1 \leq i \leq k$. Since $(a, c) \in D$ this implies $(a, b_i) \in D$, contradicting the fact mentioned in the proof of (1) above. Thus, both invariants (1) and (2) are valid.

The proposition follows from the following claim:

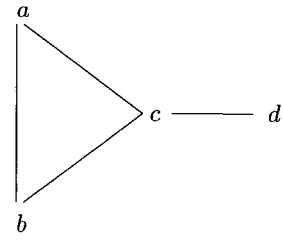
$$\text{For } (q_a)_{a \in \Sigma} = q_0 \cdot w \quad \text{we have} \quad \prod_{a \in \Sigma} q_a = \eta(w).$$

The claim is satisfied for $|w| = 0$ since $q_0 = (1)_{a \in \Sigma}$. By induction assume that the claim holds for $q = q_0 \cdot w$ and let $q' = q \cdot a$ for some letter $a \in \Sigma$. It is enough to show:

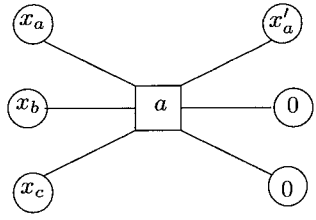
$$(\prod_{a \leq c} q_c)\, \eta(a) = (\prod_{\substack{a \leq c, \\ (a,c) \in D}} q_c)\, \eta(a)\, (\prod_{\substack{a < b, \\ (a,b) \in I}} q_b).$$

However, this last formula is immediate from the two invariants shown above. This completes the proof of the proposition. □

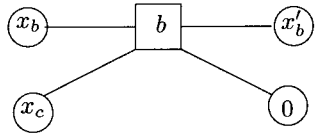
The example below illustrates the construction for a dependence alphabet on four letters $\Sigma = \{a, b, c, d\}$ with the following dependence relation D:



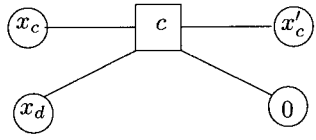
The alphabet Σ can be ordered as $a < b < c < d$ (note that $[d, c, b, a]$ is a perfect vertex elimination scheme). Let $\eta : \mathbb{M}(\Sigma, D) \to \mathbb{Z}/n\mathbb{Z}$ be a homomorphism computing the length of a trace modulo n, $n \geq 1$. Then the construction for asynchronous automata given above yields e.g. the following transition mappings, together with some concrete computations in $\mathbb{Z}/n\mathbb{Z}$:



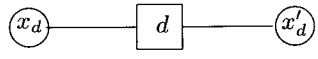
$= x_a + x_b + x_c + 1 \bmod n$



$= x_b + x_c + 1 \bmod n$



$= x_c + x_d + 1 \bmod n$



$= x_d + 1 \bmod n$

$$\delta(q_0, \begin{array}{c} a\text{---}b \\ c \\ \diagdown\ \diagup \\ d \end{array}) = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

$$\delta(q_0, a\,\text{—}b\,\text{—}c\,\text{—}d) = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \qquad \delta(q_0, d\,\text{—}c\,\text{—}b\,\text{—}a) = \begin{pmatrix} 4 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Remark 8.2.2 Note that as soon as the dependence graph contains a chordless cycle of length greater than three, the automaton constructed above is not asynchronous anymore. More precisely, for any ordering $\leq$ of the alphabet Σ there exist letters $a < b < c$ (on the cycle) satisfying $(a, c) \in D$, $(b, c) \in D$, but $(a, b) \in I$. In this case however, c belongs to both write- and read-domains of a and b.

8.3 Asynchronous Cellular Automata for Recognizable Trace Languages

In this section we give the general solution of constructing asynchronous automata for arbitrary (symmetric and reflexive) dependence alphabets. The presentation follows ideas from [277, 278, 45, 53, 46]. The main piece of work is to construct a suitable asynchronous mapping, as proposed in [45] (see also [53] for a corrected version). From an asynchronous mapping one can easily obtain a deterministic asynchronous cellular automaton, which furthermore can be easily transformed in a deterministic asynchronous equivalent automaton. This last step is not done here, since it has been already discussed in Chapter 7.

8.3.1 Prefix Order for Traces

Let us recall in this section some basic notions concerning prefixes of traces and ordering between them. We denote by $\leq$ the partial (prefix) order given by $u \leq v$ if $v = ut$ for some $t \in \mathbb{M}(\Sigma, D)$. As usual, for $u, v \in \mathbb{M}(\Sigma, D)$, $u < v$ if both $u \leq v$ and $v \neq u$; we denote by $u \sqcap v$ the greatest lower bound of u, v with respect to the prefix order. Whenever it exists, the least upper bound of u, v is denoted by $u \sqcup v$. Clearly, $u \sqcup v$ exists if and only if $u \leq w$ and $v \leq w$ for some $w \in \mathbb{M}(\Sigma, D)$.

Another notation used frequently is $\max(t)$ for $t \in \mathbb{M}(\Sigma, D)$, denoting the labellings of the maximal elements of the partial order represented by t, i.e. $\max(t) = \{a \in \Sigma \mid \exists x \in \Sigma^* : xa \in \varphi^{-1}(t)\}$, for the canonical surjective homomorphism $\varphi : \Sigma^* \to \mathbb{M}(\Sigma, D)$. Since the elements of $\max(t)$ are pairwise independent and therefore commute, $\max(t)$ can also be viewed as a trace.

The following definition introduces a basic notation for studying properties of the prefix order.

Definition 8.3.1 *Let $t \in \mathbb{M}(\Sigma, D)$, $A \subseteq \Sigma$. We define the least prefix of t which contains all letters from A by*

$$\partial_A(t) = \sqcap\{u \leq t \mid \forall a \in A : \ |u|_a = |t|_a\}$$

For $A, B \subseteq \Sigma$ we write $\partial_{A,B}(t)$ instead of $\partial_A(\partial_B(t))$. Additionally, whenever $A = \{a\}$ we write directly $\partial_a(t)$ (resp. $\partial_{a,B}(t)$ etc.).

The following lemma gives an equivalent characterization of $\partial_A(t)$.

Lemma 8.3.2 *Let $t \in \mathbb{M}(\Sigma, D)$ and $A \subseteq \Sigma$. Then*

$$\partial_A(t) = \sqcup\{u \leq t \mid \max(u) \subseteq A\}.$$

Proof: Clearly $\max(\partial_A(t)) \subseteq A$, hence $\partial_A(t) \leq \sqcup\{u \leq t \mid \max(u) \subseteq A\}$. For the other direction, consider $u, v \leq t$ such that $\max(u) \subseteq A$ and $|v|_a = |t|_a$, for every $a \in A$. Let $x, y \in \mathbb{M}(\Sigma, D)$ be such that $t = ux = vy$ and let us apply Levi's lemma to this equation. We obtain traces $p, r, s \in \mathbb{M}(\Sigma, D)$ with $\mathrm{alph}(r) \times \mathrm{alph}(s) \subseteq I$ and $u = pr$, $v = ps$.

Finally, $\max(u) \subseteq A$ implies $\max(r) \subseteq A$, hence $r \neq 1$ would contradict the definition of v. □

Remark 8.3.3 Note that for every $A \subseteq \Sigma$, the prefix $\partial_A(t)$ is equal to $\sqcup_{a \in A} \partial_a(t)$. Furthermore, for every $t \in \mathbb{M}(\Sigma, D)$ we have $t = \partial_\Sigma(t) = \partial_{\max(t)}(t)$.

The following proposition summarizes some basic properties of $\partial_A(t)$ prefixes. The easy proof is omitted and left to the reader.

Proposition 8.3.4 *Let $t, u, v \in \mathbb{M}(\Sigma, D)$, $A, B \subseteq \Sigma$ and $a \in \Sigma$. We have*

1. *$t \leq u$ implies $\partial_A(t) \leq \partial_A(u)$.*
2. *$\partial_{A,A}(t) = \partial_A(t)$.*
3. *$A \subseteq B$ implies both $\partial_A(t) \leq \partial_B(t)$ and $\partial_A(t) = \partial_{A,B}(t) = \partial_{B,A}(t)$.*
4. *$\partial_A(tu) = \partial_{A \cup D(B)}(t)\partial_A(u)$, where $B = \mathrm{alph}(\partial_A(u))$. Especially we have $\partial_a(ta) = \partial_{D(a)}(t)a$.*
5. *If $u \sqcup v$ exists, then $\partial_A(u) \sqcup \partial_A(v)$ exists, too, and $\partial_A(u) \sqcup \partial_A(v) = \partial_A(u \sqcup v)$.*
6. *$\partial_{a,A}(t) = \sqcup_{b \in A} \partial_{a,b}(t)$. Hence, for $A \neq \emptyset$ we have $\partial_{a,A}(t) = \partial_{a,b}(t)$ for some $b \in A$.*
7. *For $A \neq \emptyset$, $\partial_a(t) \leq \partial_A(t)$ implies $\partial_a(t) = \partial_{a,b}(t)$ for some $b \in A$.*

In the remaining part of this section we consider a typical situation which will be encountered in the context of asynchronous mappings. Briefly speaking, we are interested in the interconnection between two prefixes $\partial_A(t), \partial_B(t)$ of a trace $t = \partial_{A \cup B}(t)$.

Proposition 8.3.5 *Let $A, B \subseteq \Sigma$ and $t \in \mathbb{M}(\Sigma, D)$ such that $t = \partial_{A\cup B}(t)$. We denote in the following $t_A = \partial_A(t)$, $t_B = \partial_B(t)$. Consider traces r, u, v as given by Levi's lemma such that $r = t_A \sqcap t_B$, $t_A = ru$, $t_B = rv$ with $\mathrm{alph}(u) \times \mathrm{alph}(v) \subseteq I$. Then with $C = \{c \in \Sigma \mid \partial_c(t_A) = \partial_c(t_B)\}$ the following assertions hold:*

1. *For every $a \in \Sigma$ we have $\partial_a(t_A) < \partial_a(t_B)$ if and only if $a \in \mathrm{alph}(v)$. Moreover, $a \in \mathrm{alph}(v)$ implies $\partial_a(t_A) = \partial_a(r)$.*
2. *$r = \partial_C(t) = \partial_C(t_A) = \partial_C(t_B)$. In particular, $\max(r) \subseteq C$.*
3. *$\Sigma \setminus C = \mathrm{alph}(uv)$.*

Proof:

1. Use Proposition 8.3.4(4) and $\mathrm{alph}(u) \times \mathrm{alph}(v) \subseteq I$.
2. With the definition of C and Proposition 8.3.4(5) it is easily seen that we have both $\partial_C(t) = \partial_C(t_A) = \partial_C(t_B)$ and $\partial_C(t) < r$.

   If $r \neq 1$ consider $a \in \max(r)$ and let us assume $\partial_a(t_A) < \partial_a(t_B)$. With (1) we obtain $a \in \mathrm{alph}(v)$. Furthermore, $D(a) \cap \mathrm{alph}(u) \neq \emptyset$, since with Proposition 8.3.4(4)

   $$ru = \partial_A(ru) = \partial_{A\cup D(B)}(r)\partial_A(u), \text{ where } B = \mathrm{alph}(\partial_A(u)).$$

   hence $r = \partial_{A\cup D(B)}(r)$ and $u = \partial_A(u)$, and thus $\max(r) \subseteq A \cup D(\mathrm{alph}(u))$. Now, with $a \in A$ one obtains by Proposition 8.3.4(3) $\partial_a(t_A) = \partial_a(t)$, thus contradicting the assumption $\partial_a(t_A) < \partial_a(t_B)$.

   Hence, we obtained a contradiction to $\mathrm{alph}(u) \times \mathrm{alph}(v) \subseteq I$. By symmetry, $\partial_a(t_A) = \partial_a(t_B)$, which yields $a \in C$ and finally $r = \partial_{\max(r)}(r) \leq \partial_C(r) \leq \partial_C(t)$.
3. Follows directly with the definition of C together with (1).

□

Corollary 8.3.6 *Let $t \in \mathbb{M}(\Sigma, D)$ and $\emptyset \neq A, B \subseteq \Sigma$, and suppose that the sets $C_{a,b} = \{c \in \Sigma \mid \partial_{c,a}(t) = \partial_{c,b}(t)\}$ are known for all $a, b \in A \cup B$. Then we are able to determine for every $c \in \Sigma$ which of the following situations occurs:*

- $\partial_{c,A}(t) = \partial_{c,B}(t)$
- $\partial_{c,A}(t) < \partial_{c,B}(t)$
- $\partial_{c,B}(t) < \partial_{c,A}(t)$

Proof: Assume $a, b \in A \cup B$ and let $r, u, v \in \mathbb{M}(\Sigma, D)$ such that $\partial_a(t) = ru$, $\partial_b(t) = rv$ with $\mathrm{alph}(u) \times \mathrm{alph}(v) \subseteq I$. Applying Proposition 8.3.5(1) we obtain $\partial_{c,a}(t) < \partial_{c,b}(t)$ if and only if $c \in \mathrm{alph}(v)$. By Proposition 8.3.5(3) we obtain that $c \in \mathrm{alph}(v)$ implies that c, b belong to the same connected component of $\Sigma \setminus C_{a,b}$. Furthermore, c and a can not be connected by a path in $\Sigma \setminus C_{a,b}$, since $\mathrm{alph}(u) \times \mathrm{alph}(v) \subseteq I$.

Applying this argument to all pairs $(a, a') \in A \times A$ we determine $a \in A$ such that $\partial_{c,A}(t) = \partial_{c,a}(t)$. An analogous calculation yields b with $\partial_{c,B}(t) = \partial_{c,b}(t)$ and finally we have to compare $\partial_{c,a}(t)$ and $\partial_{c,b}(t)$. □

8.3.2 Asynchronous Mappings

The aim of this section is to introduce the notion of asynchronous mapping due to Cori and Métivier and to show the close connection to asynchronous cellular automata. Briefly speaking, asynchronous mappings are mappings which can be computed stepwise in a distributed way, thus being easily transformed into an equivalent asynchronous cellular automaton (with respect to recognition).

Definition 8.3.7 *A mapping $\mu : \mathbb{M}(\Sigma, D) \to S$ is called* asynchronous *if for any $t \in \mathbb{M}(\Sigma, D)$, $a \in \Sigma$ and $A, B \subseteq \Sigma$ the following conditions are satisfied.*

- *$\mu(\partial_{D(a)}(t))$ and the letter a uniquely determine the value $\mu(\partial_a(ta))$.*
- *$\mu(\partial_A(t))$ and $\mu(\partial_B(t))$ uniquely determine the value $\mu(\partial_{A \cup B}(t))$.*

The relevance of asynchronous mappings is given by the following proposition, which establishes the connection to asynchronous cellular automata. Note that in general, an asynchronous mapping $\mu : \mathbb{M}(\Sigma, D) \to S$ does not give any natural monoid automaton structure. This is due to the fact that for $t \neq \partial_{D(a)}(t)$ the value of $\mu(t)$ does not suffice for computing $\mu(ta)$, where $a \in \Sigma$.

However, given $(\mu(\partial_b(t)))_{b \in \max(t)}$, one can compute $\mu(ta)$, since for $A = \max(t) \cap I(a)$ we have $ta = \partial_A(t) \sqcup \partial_a(ta)$. Furthermore, $\mu(\partial_A(t))$ is computable from $(\mu(\partial_b(t)))_{b \in A}$ (by the 2nd condition in Definition 8.3.7), whereas $\mu(\partial_a(ta))$ is determined analogously using both conditions by a and $(\mu(\partial_b(t)))_{b \in D(a)}$. Finally, $\mu(ta)$ can be computed using again the 2nd condition (note that $\partial_A(t) = \partial_A(ta)$).

Proposition 8.3.8 *Let $\mu : \mathbb{M}(\Sigma, D) \to S$ be an asynchronous mapping and $R \subseteq S$ a subset of S. Then there exists an asynchronous cellular automaton $\mathcal{A}_\mu$ such that $L(A_\mu) = \mu^{-1}(R)$.*

Proof: Let us define $\mathcal{A}_\mu = ((Q_a)_{a \in \Sigma}, (\delta_a)_{a \in \Sigma}, q_0, F)$ by

$$Q_a = \{\mu(\partial_a(t)) \mid t \in \mathbb{M}(\Sigma, D)\}, \qquad a \in \Sigma$$

$$\delta((\mu(\partial_a(t)))_{a \in \Sigma}, b) = (\mu(\partial_a(tb)))_{a \in \Sigma}$$

With $\partial_a(tb) = \partial_a(t)$ for $a \neq b$ it is easily seen that a b-transition merely changes the local b-state. Moreover, with $\partial_b(tb) = \partial_{D(b)}(t)b$ and due to the ability of computing $\mu(\partial_b(tb))$ from $(\mu(\partial_a(t)))_{a\in D(b)}$ and b we obtain that the change of the local b-state depends only on the local states corresponding to the subset $D(b)$. Hence, δ is well-defined and the automaton is asynchronous cellular. We accomplish the construction by letting $q_0 = (\mu(1))_{a\in\Sigma}$ and $F = \{(\mu(\partial_a(t)))_{a\in\Sigma} \mid t \in \mu^{-1}(R)\}$.

In order to show $L(\mathcal{A}_\mu) = \mu^{-1}(R)$ note that $t \in L(\mathcal{A}_\mu)$ if and only if for some $u \in \mu^{-1}(R)$,

$$\mu(\partial_a(t)) = \mu(\partial_a(u)), \qquad \text{for all } a \in \Sigma.$$

Finally, with $t = \partial_\Sigma(t)$ we note that $\mu(t)$ (and $\mu(u)$ as well) is determined by $(\mu(\partial_a(t)))_{a\in\Sigma}$, hence it follows that $\mu(t) = \mu(u)$. □

8.3.3 Asynchronous Mappings for Recognizable Trace Languages

The first aim in this section is to exhibit an asynchronous mapping ν independent from the given recognizable trace language, which corresponds to a bounded time-stamping. This mapping contains crucial information about the prefix order between prefixes $\partial_a(t)$, $a \in \Sigma$, needed in order to be able to establish actuality of information. Concretely, the value $\nu(t)$ contains enough information in order to compute the set $C = \{c \in \Sigma \mid \partial_{c,A}(t) = \partial_{c,B}(t)\}$ for any $t \in \mathbb{M}(\Sigma, D)$, $A, B \subseteq \Sigma$.

The main idea for the construction consists in labelling trace prefixes of the form $\partial_x(t)$, $x \in \Sigma$, in such a way that for $a, b, c \in \Sigma$ the equality $\partial_{c,a}(t) = \partial_{c,b}(t)$ holds if and only if the labellings of these two traces coincide.

Definition 8.3.9 *Let $\nu : \mathbb{M}(\Sigma, D) \to \mathbb{N}^{\Sigma\times\Sigma}$ be defined inductively as follows:*

- $\nu(1)(a, b) = 0$.
- *If $t \neq \partial_{a,b}(t)$ then $\nu(t)(a, b) = \nu(\partial_{a,b}(t))(a, a)$.*
- *If $t = \partial_a(t)$ and $t \neq 1$ then*

$$\nu(t)(a, a) = \min\{n > 0 \mid n \neq \nu(t)(a, c) \text{ for all } c \neq a\}$$

Lemma 8.3.10 *The mapping $\nu : \mathbb{M}(\Sigma, D) \to \mathbb{N}^{\Sigma\times\Sigma}$ is well-defined and satisfies for every $t \in \mathbb{M}(\Sigma, D)$, $a, b, c \in \Sigma$, and $A, B \subseteq \Sigma$ the following properties:*

1. $0 \leq \nu(t)(a, b) \leq |\Sigma|$.
2. $\nu(t)(a, b) = \nu(\partial_b(t))(a, a) = \nu(\partial_{a,b}(t))(a, a)$.
3. $\nu(t)(a, b) = 0$ *is equivalent to* $\partial_{a,b}(t) = 1$.
4. $\partial_{c,a}(t) = \partial_{c,b}(t)$ *is equivalent to* $\nu(t)(c, a) = \nu(t)(c, b)$.
5. $\partial_{c,A}(t) = \partial_{c,B}(t)$ *is equivalent to* $\nu(\partial_A(t))(c, c) = \nu(\partial_B(t))(c, c)$.

Proof:

1. Follows directly from $|\{n > 0 \mid n \neq \nu(t)(a,c) \text{ for every } c \neq a\}| \leq |\Sigma| - 1$.

2. Obvious.

3. Follows directly from $\nu(t)(a,b) = \nu(\partial_{a,b}(t))(a,a)$ using the definition of ν.

4. The implication from left to right is clear due to (2). Conversely, assume w.l.o.g. that $\partial_{c,a}(t) < \partial_{c,b}(t)$. By Proposition 8.3.5 we have with $r = \partial_a(t) \sqcap \partial_{c,b}(t)$ the equality $\partial_{c,a}(t) = \partial_c(r)$. Hence, for some $d \in \max(r)$ (for $r \neq 1$) we obtain $\partial_{c,a}(t) = \partial_{c,d}(r) = \partial_{c,d}(\partial_{c,b}(t))$ $(= \partial_{c,d}(\partial_a(t)))$. Note also that $d \neq c$, since otherwise $\partial_{c,a}(t) = \partial_{c,b}(t)$.

   Finally, the definition of ν yields:

$$\nu(t)(c,b) = \nu(\partial_{c,b}(t))(c,c) \overset{c \neq d}{\neq} \nu(\partial_{c,b}(t))(c,d) = \nu(\partial_{c,d}(\partial_{c,b}(t)))(c,c),$$

   thus contradicting the hypothesis $\nu(t)(c,a) = \nu(t)(c,b)$.

5. Assume that $A, B \neq \emptyset$ (otherwise we use (3) and the definition of ν). For suitable $a \in A$, $b \in B$ we have $\partial_{c,A}(t) = \partial_{c,a}(t)$ and $\partial_{c,B}(t) = \partial_{c,b}(t)$. Together with

$$\nu(\partial_A(t))(c,c) \overset{(2)}{=} \nu(\partial_{c,A}(t))(c,c) = \nu(\partial_{c,a}(t))(c,c) \overset{(2)}{=} \nu(t)(c,a),$$

   $(\nu(t)(c,b) = \nu(\partial_B(t))(c,c)$ analogously) and (4) the result follows immediately.

□

Remark 8.3.11 Note that the value of $\nu(t)$ allows to determine whether $\partial_a(t) \leq \partial_b(t)$ holds, with $a, b \in \Sigma$. To see this note that $\partial_a(t) \leq \partial_b(t)$ is equivalent to $\partial_a(t) = \partial_{a,b}(t)$, hence to $\nu(t)(a,a) = \nu(t)(a,b)$ by Lemma 8.3.10.

For the asynchronous cellular automaton A_ν defined at the beginning of the section this implies the following: given a trace t and the local states $q_x := \delta(q_0, t)_x$, $x \in \{a,b\}$, one can determine if $\partial_a(t) \leq \partial_b(t)$ or $\partial_b(t) \leq \partial_a(t)$ holds, or $\partial_a(t)$ and $\partial_b(t)$ are incomparable, since:

$$\begin{array}{l} \partial_a(t) \leq \partial_b(t) \iff \\ \nu(\partial_a(t))(a,a) \overset{8.3.10(2)}{=} \nu(t)(a,a) = \nu(t)(a,b) \overset{8.3.10(2)}{=} \nu(\partial_b(t))(a,a) \iff \\ q_a(a,a) = q_b(a,a) \end{array}$$

Proposition 8.3.12 *The mapping ν is asynchronous.*

Proof: Let us first show that for any $t \in \mathbb{M}(\Sigma, D)$, $a \in \Sigma$ the value $\nu(\partial_{D(a)}(t))$ and the letter a suffice for determining $\nu(\partial_a(ta))$. With Lemma 8.3.10(2) we directly obtain for $b, c \in \Sigma$ with $c \neq a$ resp. $c = a \neq b$,

$$\nu(\partial_a(ta))(b,c) = \nu(\partial_{D(a)}(t))(b,c),$$

since $\partial_{b,c,a}(ta) = \partial_{b,c}(\partial_{D(a)}(t)a) = \partial_{b,c}(\partial_{D(a)}(t))$, resp. $\partial_{b,a,a}(ta) = \partial_b(\partial_{D(a)}(t)a) = \partial_{b,D(a)}(t) = \partial_{b,a}(\partial_{D(a)}(t))$.

Finally, $\nu(\partial_a(ta))(a,a) = \min\{n > 0 \mid n \neq \nu(\partial_a(ta))(a,d), d \neq a\}$, which is computable using $\nu(\partial_a(ta))(a,d) = \nu(\partial_{D(a)}(t))(a,d)$ (note that $a \neq d$ implies $\partial_{a,d,D(a)}(t) = \partial_{a,d,a}(ta)$).

Now assume that $\nu(\partial_A(t))$ and $\nu(\partial_B(t))$ are given, with $A, B \subseteq \Sigma$. Due to Lemma 8.3.10(2) it suffices to show that using $\nu(\partial_A(t))$, $\nu(\partial_B(t))$ we are able to determine whether for a given $c \in \Sigma$,

$$\partial_{c,B}(t) \leq \partial_{c,A}(t) (= \partial_{c,A\cup B}(t)) \quad \text{or} \quad \partial_{c,A}(t) < \partial_{c,B}(t) (= \partial_{c,A\cup B}(t))$$

holds. By Corollary 8.3.6 it suffices to know the sets $C_{a,b} = \{c \in \Sigma \mid \partial_{c,a}(t) = \partial_{c,b}(t)\}$ for $a, b \in A \cup B$. This can be achieved using the information provided by $\nu(\partial_A(t))$ and $\nu(\partial_B(t))$, since $\nu(t)(c,a) \overset{8.3.10(2)}{=} \nu(\partial_A(t))(c,a)$ for $a \in A$ (analogously for B) and

$$\partial_{c,a}(t) = \partial_{c,b}(t) \overset{8.3.10(4)}{\iff} \nu(t)(c,a) = \nu(t)(c,b)$$

□

The information provided by Zielonka's time-stamping is sufficient in order to obtain a more complex structural information about a trace. More precisely, with the knowledge of $\nu(t)$ we are able to compute the second approximation introduced by Cori and Métivier in [45] and used in the construction of asynchronous cellular automata. The second approximation $\Delta_2(t)$ of a trace t is the directed graph with vertex set $\Sigma \times \Sigma$ and edge set $\{((a,c),(b,d)) \mid \partial_{a,c}(t) \leq \partial_{b,d}(t)\}$. However, Cori, Métivier and Zielonka showed that the computation of the mapping ν provides sufficient information about trace prefixes in order to obtain an asynchronous mapping [46]. Since the second approximation is an interesting information for itself, we show in the following how to compute it.

First, let us show the following property for $\Delta_2(t)$, which is a stronger variant of the first property required for asynchronous mappings.

Lemma 8.3.13 *Let $t \in \mathbb{M}(\Sigma, D)$ and $a \in \Sigma$. Then the second approximation $\Delta_2(ta)$ is computable using $\Delta_2(t)$ and the letter a.*

Proof: Let $b, c \in \Sigma$. Then we have

$$\partial_{b,c}(ta) = \begin{cases} \partial_{b,c}(t) & \text{if } c \neq a \\ \partial_{b,D(a)}(t) & \text{if } b \neq c = a \\ \partial_{D(a)}(t)a = \partial_a(ta) & \text{if } b = c = a \end{cases}$$

Note that if $b \neq c = a$ holds, then we can use $\Delta_2(t)$ for computing a letter $c' \in D(a)$ with $\partial_{b,D(a)}(t) = \partial_{b,c'}(t)$.

Finally, we note that $\partial_{D(a)}(t)a$ is not a prefix of t; conversely, a prefix of t is a prefix of $\partial_{D(a)}(t)a$ if and only if it is a prefix of $\partial_{D(a)}(t)$. To conclude, we have $\partial_{D(x)}(t)x \leq \partial_{D(y)}(t)y$ if and only if $x = y$. □

Proposition 8.3.14 *Let $t \in \mathbb{M}(\Sigma, D)$ and $A, B \subseteq \Sigma$ be given and denote $t_1 = \partial_A(t)$, $t_2 = \partial_B(t)$, $r = t_1 \sqcap t_2$, $t_1 = ru$ and $t_2 = rv$ with $u, v \in \mathbb{M}(\Sigma, D)$, $\mathrm{alph}(u) \times \mathrm{alph}(v) \subseteq I$.*

Let $\mu(t_i) = (\nu(t_i), \Delta_2(t_i))$, $i = 1, 2$. Then we can compute the second approximation $\Delta_2(t)$ from $\mu(t_1), \mu(t_2)$.

Proof: Let $C = \{c \in \Sigma \mid \partial_c(t_1) = \partial_c(t_2)\} \overset{8.3.10}{=} \{c \in \Sigma \mid \nu(t_1)(c,c) = \nu(t_2)(c,c)\}$. For convenience we use the following notations: for $x, y \in \Sigma$ we denote (x,y) a u-point if $\partial_{x,C}(t_1)$ is a strict prefix of $\partial_{x,y}(t_1)$, i.e. $\partial_{x,C}(t_1) < \partial_{x,y}(t_1)$; (x,y) is a v-point if $\partial_{x,C}(t_2) < \partial_{x,y}(t_2)$. Otherwise, (x,y) is r-point. The above notations illustrate the graphical meaning, i.e. the position of the maximal vertex of $\partial_{x,y}(t)$ in the subgraph of the dependence graph which corresponds to u, v, or r respectively.

Note that if $C = \emptyset$ then (x,y) is a u-point if and only if $\partial_{x,y}(t_1) \neq 1$, hence if $\nu(t_1)(x,y) \neq 0$; analogously, (x,y) is a v-point if and only if $\nu(t_2)(x,y) \neq 0$ and finally, (x,y) is a r-point if and only if $\nu(t_1)(x,y) = \nu(t_2)(x,y) = 0$.

If $C \neq \emptyset$, then with Corollary 8.3.6 we may use $\nu(t_1)$ in order to determine whether (x,y) is a u-point (resp. $\nu(t_2)$ for a v-point). We assume in the following $C \neq \emptyset$. Observe that for $a, c \in \Sigma$,

$$\partial_{a,c}(t) = \begin{cases} \partial_{a,c}(t_1) & \text{if } (a,c) \text{ is a } u\text{-point} \\ \partial_{a,c}(t_2) & \text{if } (a,c) \text{ is a } v\text{-point} \\ \bigsqcup_{i=1,2} \partial_{a,c}(t_i) \leq \partial_a(r) & \text{if } (a,c) \text{ is a } r\text{-point} \end{cases}$$

Furthermore, if (a,c) is a r-point, then we have to distinguish if (c,c) is a u-, v- or r-point. Note that we have $\partial_{a,c}(t) = \partial_{a,c}(t_1)$, if (c,c) is a u-point, respectively $\partial_{a,c}(t) = \partial_{a,c}(t_2)$, if (c,c) is a v-point and finally, $\partial_{a,c}(t) = \partial_{a,c}(t_i)$ $(i = 1, 2)$, if (c,c) is a r-point (hence, $c \in C$).

Consider now for $a, b, c, d \in \Sigma$ the question, whether or not $\partial_{a,c}(t) \leq \partial_{b,d}(t)$ holds and suppose (a,c) is a α-point and (b,d) is a β-point, where $\alpha, \beta \in \{u, v, r\}$.

First, observe that if $\{\alpha, \beta\} = \{u, v\}$ or ($\beta = r$ and $\alpha \neq r$), then the answer is negative. Else, if $\alpha = \beta$, then we can provide an answer using either $\Delta_2(t_1)$ or $\Delta_2(t_2)$.

We now consider the situation $\alpha = r$, $\beta = u$ (the case $\beta = v$ can be handled analogously). Now, if (c,c) is either a u- or a r-point, then we can use again $\Delta_2(t_1)$ (see remark above) and check $\partial_{a,c}(t_1) \leq \partial_{b,d}(t_1)$.

Finally, we consider the case where $\alpha = r$, $\beta = u$ and (c,c) is a v-point, hence $\partial_{a,c}(t) = \partial_{a,c}(t_2)$. It suffices to show that a letter $f \in \Sigma$ exists effectively such that $\partial_{a,c}(t_2) = \partial_{a,f}(t_1)$ holds.

Assume $\partial_{a,c}(t_2) \neq 1$ (otherwise, $\partial_{a,c}(t_1) = 1$, too, and we choose $f = c$). Recall $r = \partial_C(t) = \partial_C(t_1) = \partial_C(t_2)$ and consider the trace $r' = r \sqcap \partial_c(t_2)$, with $r = r'x$, $\partial_c(t_2) = r'y$ and $\mathrm{alph}(x) \times \mathrm{alph}(y) \subseteq I$ (see also Figure 8.2).

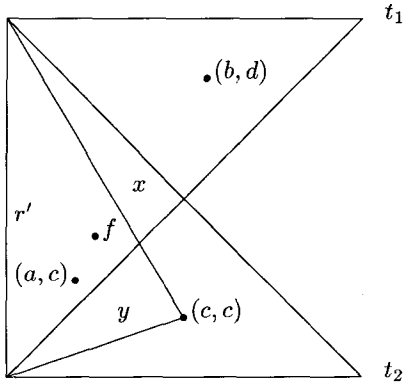


Figure 8.2: $\alpha = r$, $\beta = u$ and (c, c) is a v-point.

With $\partial_{a,c}(t_2) = \partial_{a,c}(t) \leq \partial_a(r)$ we immediately obtain $\partial_{a,c}(t_2) = \partial_a(r')$, in particular $r' \neq 1$. Let $C' = \{e \in \Sigma \mid \partial_e(\partial_c(t_2)) = \partial_e(r)\}$, hence with Proposition 8.3.5 we obtain $C' \supseteq \max(r') \neq \emptyset$. Furthermore, with Proposition 8.3.4 we have $\partial_{a,c}(t_2) = \partial_a(r') = \partial_{a,f}(r') = \partial_{a,f}(r)$ for some $f \in \max(r')$, hence $f \in C'$. We show $\partial_f(r) = \partial_f(t_1)$. For this, note that by definition of r', we have $f \notin \mathrm{alph}(x)$. Moreover, due to $y \neq 1$ (otherwise we contradict (c, c) being a v-point), together with $f \in \max(r')$, we obtain $f \in D(\mathrm{alph}(y))$, hence $f \in D(\mathrm{alph}(v))$. Thus, since $\mathrm{alph}(u) \times \mathrm{alph}(v) \subseteq I$, we deduce $f \notin \mathrm{alph}(u)$. Finally, since $t_1 = r'xu$ we immediately conclude $\partial_f(t_1) = \partial_f(r') = \partial_f(r)$.

A letter f with $\partial_{a,c}(t_2) = \partial_{a,f}(t_1)$ can now be computed effectively as follows: using $\nu(t_1)$, $\nu(t_2)$ we first determine the sets C and $\mathrm{alph}(v)$ (see Corollary 8.3.6). Using again $\nu(t_2)$ we compute C'. Finally, using $\nu(t_1)$ we choose $f \in C' \cap D(\mathrm{alph}(v))$ such that $\partial_{a,C}(t_1) \leq \partial_{a,f}(t_1)$. We obtain finally

$$\partial_{a,f}(t_1) \overset{f \in D(\mathrm{alph}(v))}{=} \partial_{a,f}(r) \overset{f \in C'}{=} \partial_{a,f,c}(t_2) \leq \partial_{a,c}(t_2) \leq \partial_{a,C}(t_2) = \partial_{a,C}(t_1) \leq \partial_{a,f}(t_1)$$

□

In the remaining of this section we accomplish the construction of an asynchronous mapping $\mu : \mathbb{M}(\Sigma, D) \to S$ for a given trace language $L \in \mathrm{Rec}(\mathbb{M})$, such that $L = \mu^{-1}\mu(L)$. The mapping μ is obtained by augmenting the basic mapping ν by a component depending on a homomorphism to a finite monoid recognizing L. For this new component we use Zielonka's ∇ notation.

Definition 8.3.15 *Let $t \in \mathbb{M}(\Sigma, D)$, $A \subseteq \Sigma$. By $\nabla_A(t)$ we denote the maximal suffix of t containing letters from A, only, i.e. the unique suffix s of t with $t = \partial_{\bar{A}}(t)s$, where $\bar{A} = \Sigma \setminus A$.*

Remark 8.3.16 It follows easily that $\nabla_A \nabla_B(t) = \nabla_{A \cap B}(t)$, for every $t \in \mathbb{M}(\Sigma, D)$, $A, B \subseteq \Sigma$.

The next proposition shows that the mapping associating with each trace t the tuple $(\nabla_A(t))_{A \subseteq \Sigma}$ is asynchronous.

Proposition 8.3.17 *We have*

1. *Let $t_1, t_2 \in \mathbb{M}(\Sigma, D)$, $A \subseteq \Sigma$. Then*
$$\nabla_A(t_1 t_2) = \nabla_{A \cap I(B)}(t_1) \nabla_A(t_2),$$
where $B = \mathrm{alph}(\partial_{\bar{A}}(t_2))$.

2. *Let $t \in \mathbb{M}(\Sigma, D)$, $a \in \Sigma$. Then*
$$\nabla_A(\partial_a(ta)) = \begin{cases} \nabla_A(\partial_{D(a)}(t))a & \text{for } a \in A \\ \nabla_{A \cap I(a)}(\partial_{D(a)}(t)) & \text{else} \end{cases}$$

3. *Let $t \in \mathbb{M}(\Sigma, D)$, $A, B \subseteq \Sigma$ such that $t = \partial_{A \cup B}(t)$. With the notations of Proposition 8.3.5 we have for every $E \subseteq \Sigma$*
$$\nabla_E(t) = \nabla_{E \cap I(F)}(t_A) \nabla_{E \cap \bar{C}}(t_B), \quad \text{where } F = \mathrm{alph}(\partial_{\bar{E}}(v)).$$

Proof:

1. By Proposition 8.3.4(4) we obtain $\partial_{\bar{A}}(t_1 t_2) = \partial_{\bar{A} \cup D(B)}(t_1) \partial_{\bar{A}}(t_2)$. With Remark 8.3.16 and the cancellative property of $\mathbb{M}(\Sigma, D)$ the result immediately follows.

2. Is a direct consequence of (1) for $t_1 = \partial_{D(a)}(t)$, $t_2 = a$.

3. First note that for $E \subseteq \Sigma$ and $G := \mathrm{alph}(\partial_{\bar{E}}(uv)) \supseteq F$,
$$\nabla_E(t) \overset{(1)}{=} \nabla_{E \cap I(G)}(r) \nabla_E(uv) = \nabla_{E \cap I(G)}(r) \nabla_E(u) \nabla_E(v),$$
where the last equality is due to $\mathrm{alph}(u) \times \mathrm{alph}(v) \subseteq I$. Let us now consider the right side of the claimed identity, noting that $\partial_{\bar{E} \cup D(F)}(u) = \partial_{\bar{E}}(u)$ (since $F \subseteq \mathrm{alph}(v)$):

   - $\nabla_{E \cap I(F)}(t_A) \overset{(1)}{=} \nabla_{E \cap I(G)}(r) \nabla_E(u)$
   - Let $H = \mathrm{alph}(\partial_{\bar{E} \cup C}(v))$. Then,
   $$\nabla_{E \cap \bar{C}}(t_B) \overset{(1)}{=} \nabla_{E \cap \bar{C} \cap I(H)}(r) \nabla_{E \cap \bar{C}}(v) = \nabla_E(v),$$
   due to $\max(r) \subseteq C$ and $\nabla_{\bar{C}}(v) = v$ by Proposition 8.3.5, together with Remark 8.3.16.

□

Corollary 8.3.18 *Let N be a finite monoid and $\eta : \mathbb{M}(\Sigma, D) \to N$ a homomorphism. Let S be the finite set*

$$S = \{(\nu(t), (\eta(\nabla_A(t)))_{A \subseteq \Sigma}) \mid t \in \mathbb{M}(\Sigma, D)\}$$

Then the mapping $\mu : \mathbb{M}(\Sigma, D) \to S$, $\mu(t) = (\nu(t), (\eta(\nabla_A(t)))_{A \subseteq \Sigma})$ is asynchronous and the homomorphism η factorizes through μ.

Proof: The mapping μ is asynchronous by Propositions 8.3.12, 8.3.17. Since $t = \nabla_\Sigma(t)$ for any $t \in \mathbb{M}(\Sigma, D)$, $\mu(t) = \mu(t')$ implies $\eta(t) = \eta(t')$, hence the assertion. □

Main Theorem 8.3.19 *Let $D \subseteq \Sigma \times \Sigma$ be a symmetric and reflexive dependence relation and $L \subseteq \mathbb{M}(\Sigma, D)$ a recognizable trace language. Then there exists a deterministic finite asynchronous cellular automaton recognizing L.*

Proof: Corollary 8.3.18 provides an asynchronous mapping with finite image such that $L = \mu^{-1}\mu(L)$. The asynchronous cellular automaton $\mathcal{A}_\mu$ constructed in Proposition 8.3.8 accepts exactly L. □

8.4 Concluding Remarks

Relating the algebraic recognizability of trace languages to recognizability by devices with distributed control has been a subject of considerable efforts during the eighties. Very interesting constructions providing partial solutions have been given, like C. Duboc's mixed products of automata [75] or Y. Métivier's solution for acyclic dependence graphs [196], presented in a generalized form in Section 8.2. Of course, the most important contribution is W. Zielonka's solution for the general case [277, 278] (see also [45, 46, 53]). The construction given by Zielonka is involved and the question, whether a simpler construction exists, is still of actuality.

The importance of constructing asynchronous automata is also underlined by applications, where automata of small size are highly required. Considerations concerning efficient (and simpler) constructions are also the topic of two kinds of current contributions. The first one concerns modular constructions of asynchronous automata, based on concurrent rational representations of recognizable trace languages, and yields non-deterministic asynchronous automata [228]. The second one deals also with a classical approach and can be regarded as a completion of the first one: determinization of asynchronous automata. The inherent difficulty of the problem of constructing deterministic asynchronous automata is also suggested by both determinization procedures given recently by Klarlund et. al. [154] and by the second author [198].