A note on Métivier's construction of asynchronous automata for triangulated graphs *

Volker Diekert Anca Muscholl

Universität Stuttgart
Institut für Informatik
Breitwiesenstr. 20-22
D-70565 Stuttgart

1 Introduction

Free partially commutative monoids have been introduced in computer science by Mazurkiewicz as *trace monoids* [Maz77]. They have been widely adopted as a suitable model for concurrency.

A particularly interesting language family is given by considering recognizable subsets. In the context of trace monoids we have a natural finite automaton model with a distributed control structure, the asynchronous (cellular) automaton. By Zielonka's important theorem deterministic asynchronous (cellular) automata precisely characterize recognizability of trace languages. It is well-known that Zielonka's construction is rather involved, making use of a bounded time-stamping technique which gives information about trace prefixes. With this information, distributed control is possible. The question, whether a substantially simpler construction for deterministic asynchronous automata exists, is still open. Recently, two determinization constructions for asynchronous automata have been given [KMS94, Mus94], but both rely on the complicated time-stamping technique.

In the special case of acyclic dependence graphs Y. Métivier exhibited a surprisingly elegant solution [Mét87], providing in a natural way deterministic asynchronous automata for recognizable languages. The drawback is the fact that acyclicity is a strong restriction on concurrency alphabets. In fact, even the special case of complete dependence, i.e. the free monoid, is not included. We show here that for a larger class of graphs –the *triangulated* graphs– the main idea of Métivier's construction still applies. We are thus able to generalize this simple construction to triangulated graphs. From a practical point of view, this means

*Research supported by the ESPRIT Basic Research Action No. 6317 (ASMICS 2). This paper appeared in *The Book of Traces*, World Scientific, 1995.

that by adding sufficient dependency, 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 introducing some notations.
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 morphisms: a trace language $L \subseteq \mathbb{M}(\Sigma, D)$ is *recognizable* if there exists a finite monoid S and a morphism $\eta : \mathbb{M}(\Sigma, D) \to S$ recognizing L, i.e. satisfying $L = \eta^{-1}\eta(L)$. Alternatively, by Zielonka's characterization theorem, recognizability is equivalent to recognition by finite asynchronous automata. (Note that we work here with deterministic automata, only.)
An *asynchronous automaton* $\mathcal{A}$ is a tuple $(\prod_{i=1}^m Q_i, (\delta_a)_{a\in\Sigma}, q_0, F)$ 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 $\mathrm{dom}(a) \subseteq \{1, \ldots, m\}$ such that for all $(a, b) \in I$ we have $\mathrm{dom}(a) \cap \mathrm{dom}(b) = \emptyset$.
- To each letter $a \in \Sigma$ there is 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 $j \notin \mathrm{dom}(a)$
  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.

2 Triangulated graphs

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

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 $i \leq j$.
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 Dirac [Dir61]. The result and the proof hereof can be found e.g. in the textbook of Golumbic [Gol80, Thm. 4.1].
Note that 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.

3 Extending Métivier's construction

In [Mét87] 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 [Per89]). 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 explain this simple construction below. In fact, our result shows that it applies to triangulated graphs. All we need is a perfect vertex elimination scheme.
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. First we define for $\Gamma \subseteq \Sigma$ and $s_c \in S$ ($c \in \Gamma$) the ordered product $\prod_{c \in \Gamma} s_c$ by multiplying the elements according to the ordering in Σ. Thus, $\prod_{c \in \Gamma} s_c = s_{c_1} \cdots s_{c_m}$, 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.
Consider $L = \eta^{-1}(R)$ for some subset $R \subseteq S$. We are going to construct an asynchronous automaton $\mathcal{A}$ recognizing L. Of course, by standard techniques of a simple time-stamping modulo the degree of (Σ, D), the resulting automaton can be transformed into an equivalent asynchronous cellular one.
The state set of $\mathcal{A}$ will be an n-fold direct product of S, $n = |\Sigma|$. The automaton $\mathcal{A} = (\prod_{a \in \Sigma} Q_a, \delta, q_0, F)$ is then defined by

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

where for every $a \in \Sigma$:

$$Q_a = S$$

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

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 3.1 *Let $L \in \mathrm{Rec}(\mathbb{M}(\Sigma, D))$ be recognized by the morphism $\eta : \mathbb{M}(\Sigma, D) \to S$ 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–domain. 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 1. This is a contradiction to $(a,b) \in I$. Thus, $(a, b_i) \in I$ for all $1 \le i \le k$; a fact which will be used below, too.
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 $c = c_0 < \cdots < c_m$, $k,m \ge 0$, such that $(b_{i-1}, b_i) \in D$, $(c_{j-1}, c_j) \in D$ for $1 \le i \le k$, $1 \le j \le m$. We show by induction that $(b_i, c_j) \in I$ for all i,j. Assume first that $c_m \le b_k$. Then we have $k \ge 1$ (since $b < c$) and $(c_m, b_k) \in D$, $(b_{k-1}, b_k) \in D$

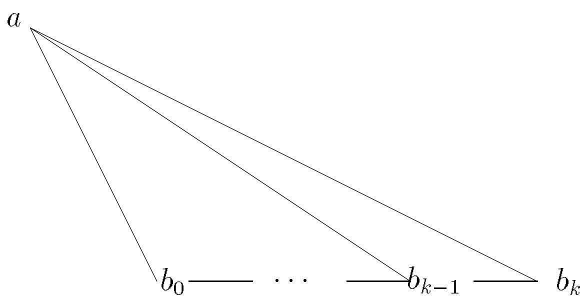


Figure 1: Consequence of $(a, b_k) \in D$.

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:

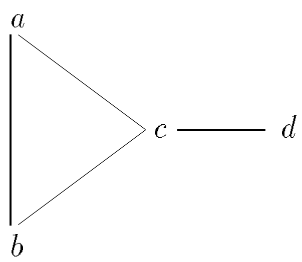
$$\text{For } (q_a)_{a \in \Sigma} = q_0 \cdot w \quad \text{we have} \quad \prod_{a \in \Sigma} q_a = \eta(w).$$

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:

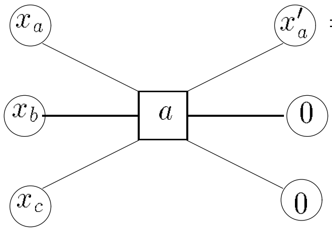
$$\Big(\prod_{a \leq c} q_c\Big)\ \eta(a) = \Big(\prod_{\substack{a \leq c, \\ (a,c) \in D}} q_c\Big)\ \eta(a)\ \Big(\prod_{\substack{a < b, \\ (a,b) \in I}} q_b\Big).$$

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

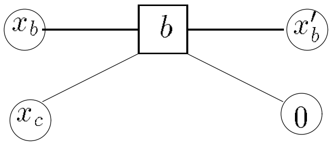
The example below illustrates the construction for a dependence alphabet on four letters $\Sigma = \{a, b, c, d\}$ with the following dependence relation D:



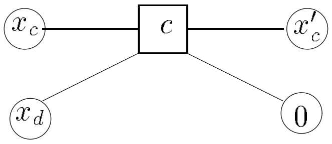
The alphabet Σ can be ordered as $a < b < c < d$ (note that $[d, c, b, a]$ is a perfect vertex elimination scheme). Let $\eta : \mathbb{M}(\Sigma, D) \to \mathbb{Z}/n\mathbb{Z}$ be a morphism computing the length of a trace modulo n, $n \geq 1$. Then the construction for asynchronous automata given above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



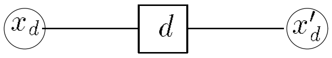
$= x_a + x_b + x_c + 1 \bmod n$



$= x_b + x_c + 1 \bmod n$



$= x_c + x_d + 1 \bmod n$



$= x_d + 1 \bmod n$

$$\delta\left(q_0, c \begin{array}{l} \diagup\, a — b \\ \diagdown\, d \end{array}\right) = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

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

Remark 3.2 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–domain of a and b.

References

[Dir61] G. A. Dirac. On rigid circuit graphs. Abh. Math. Sem. 25, Univ. Hamburg, 1961.

[DR95] V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, Singapore, 1995.

[Gol80] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.

[KMS94] N. Klarlund, M. Mukund, and M. Sohoni. Determinizing asynchronous automata. In Serge Abiteboul and Eli Shamir, editors, *Proceedings of the 21st International Colloquium on Automata, Languages and Programming (ICALP'94), Jerusalem (Israel) 1994*, number 820 in Lecture Notes in Computer Science, pages 130–141, 1994.

[Maz77] A. Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977.

[Mét87] Y. Métivier. An algorithm for computing asynchronous automata in the case of acyclic non-commutation graph. In Th. Ottmann, editor, *Proceedings of the 14th International Colloquium on Automata, Languages and Programming (ICALP'87), Karlsruhe (FRG) 1987*, number 267 in Lecture Notes in Computer Science, pages 226–236, Berlin-Heidelberg-New York, 1987. Springer.

[Mus94] A. Muscholl. On the complementation of Büchi asynchronous cellular automata. In Serge Abiteboul and Eli Shamir, editors, *Proceedings of the 21st International Colloquium on Automata, Languages and Programming (ICALP'94), Jerusalem (Israel) 1994*, number 820 in Lecture Notes in Computer Science, pages 142–153. Springer, 1994.

[Per89] D. Perrin. Partial commutations. In *Proceedings of the 16th International Colloquium on Automata, Languages and Programming (ICALP'89), Stresa (Italy) 1989*, number 372 in Lecture Notes in Computer Science, pages 637–651, Berlin-Heidelberg-New York, 1989. Springer.