Automata, Languages and Programming, 21st International Colloquium, Proceedings, Serge Abiteboul, Eli Shamir (Eds.),
Lecture Notes in Computer Science **820**, Springer-Verlag (1994) 130–141.

Determinizing Asynchronous Automata

Nils Klarlund[1*] Madhavan Mukund[2**], Milind Sohoni[2]

[1] BRICS Center, Aarhus University, Ny Munkegade,
DK 8000 Aarhus C, Denmark. E-mail: klarlund@daimi.aau.dk
[2] School of Mathematics, SPIC Science Foundation, 92 G N Chetty Rd
Madras 600 017, India. E-mail: {madhavan,sohoni}@ssf.ernet.in

Abstract. An asynchronous automaton consists of a set of processes that cooperate in processing letters of the input. Each letter read prompts some of the processes to synchronize and decide on a joint move according to a non-deterministic transition relation.

Zielonka's theorem tells us that these automata can be determinized while retaining the synchronization structure. Unfortunately, this construction is indirect and yields a triple-exponential blow-up in size.

We present a direct determinization procedure for asynchronous automata which generalizes the classical subset construction for finite-state automata. Our construction is only double-exponential and thus is the first to essentially match the lower bound.

Introduction

Asynchronous automata, introduced by Zielonka [Zie1], are a natural generalization of finite-state automata for concurrent systems. An asynchronous automaton consists of a set of processes which periodically synchronize to process their input. Each letter a in the alphabet is associated with a subset $\theta(a)$ of processes which jointly decide on a move on reading a.

A distributed alphabet of this type gives rise to an *independence relation* I between letters: $(a,b) \in I$ iff a and b are processed by disjoint sets of components.

Alphabets equipped with independence relations, also called *concurrent alphabets*, were introduced by Mazurkiewicz for studying concurrent systems from the viewpoint of formal language theory [Maz]. Given a concurrent alphabet (Σ, I), I induces a equivalence relation $\sim$ on Σ^*: for any two words w and w', $w \sim w'$ iff w' can be obtained from w by a sequence of permutations of adjacent independent letters. The equivalence class $[w]$ containing w is called a *trace*.

A language $L \subseteq \Sigma^*$ is said to be a *trace language* if L is $\sim$-consistent—i.e., for each $w \in \Sigma^*$, if w is in L then all of $[w]$ is contained in L. A trace language is *recognizable* if it is accepted by a conventional finite state automaton.

Asynchronous automata are natural machines for recognizing trace languages. Given a conventional finite automaton recognizing a trace language over (Σ, I),

* The author is supported by a fellowship from the Danish Research Council.
** The author was partially supported by IFCPAR Project 502-1

we can construct a *deterministic* asynchronous automaton over a distributed alphabet (Σ, θ) recognizing the same language, such that the independence relation induced by θ is precisely I. This result was first proved by Zielonka [Zie1].

Contributions of this paper

In this paper, we generalize the classical subset construction of Rabin and Scott and obtain a direct procedure for determinizing an asynchronous automaton $\mathfrak{A}$. This construction is the first that involves only a double-exponential blow-up in the size of the state spaces. We also show that this bound is essentially optimal.

The only other known way to determinize a non-deterministic asynchronous automaton $\mathfrak{A}$ is *indirect*: view it as a normal non-deterministic automaton at the level of "global states" and then apply Zielonka's construction to obtain a deterministic asynchronous automaton. This route leads to a triple exponential blowup in general, even if we use the newer constructions of [CMZ, MS].

Our determinization construction is important for dealing with asynchronous automata over infinite traces [GP]. These automata generalize automata on infinite strings and lead to a nice theory of ω-regular trace languages. However, Büchi asynchronous automata [GP] are necessarily non-deterministic and lack an effective complementation construction. With a Muller acceptance condition, deterministic automata suffice [DM], but an effective determinization procedure is still missing. We believe that our subset construction will lead to direct constructions for determinizing and complementing these automata.

Recently, Muscholl [Mus] has described a subset construction for asynchronous *cellular* automata [Zie2], as part of a complementation procedure for Büchi asynchronous cellular automata. Though asynchronous cellular automata are expressively equivalent to asynchronous automata, the operational intuition underlying the two models is quite different and there appears to be no obvious way to connect the construction in [Mus] to the one presented here.

1 Preliminaries

Distributed alphabet Let $\mathcal{P}$ be a finite set of processes. A *distributed alphabet* is a pair (Σ, θ) where Σ is a finite set of *actions* and $\theta : \Sigma \to 2^{\mathcal{P}}$ assigns a non-empty set of processes to each $a \in \Sigma$.

State spaces With each process p, we associate a finite set of states denoted V_p. Each state in V_p is called a *local state*. For $P \subseteq \mathcal{P}$, we use V_P to denote the product $\prod_{p \in P} V_p$. An element $\vec{v}$ of V_P is called a *P-state*. A $\mathcal{P}$-state is also called a *global state*. Given $\vec{v} \in V_P$, and $P' \subseteq P$, we use $\vec{v}_{P'}$ to denote the projection of $\vec{v}$ onto $V_{P'}$. Also, $\vec{v}_{\overline{P'}}$ abbreviates $\vec{v}_{P-P'}$. For a singleton $p \in P$, we write $\vec{v}_p$ for $\vec{v}_{\{p\}}$. For $a \in \Sigma$, we write V_a to mean $V_{\theta(a)}$ and $V_{\overline{a}}$ to mean $V_{\overline{\theta(a)}}$. Similarly, if $\vec{v} \in V_P$ and $\theta(a) \subseteq P$, we write $\vec{v}_a$ for $\vec{v}_{\theta(a)}$ and $\vec{v}_{\overline{a}}$ for $\vec{v}_{\overline{\theta(a)}}$.

Asynchronous automaton An *asynchronous automaton* $\mathfrak{A}$ over (Σ, θ) is of the form $(\{V_p\}_{p \in \mathcal{P}}, \{\to_a\}_{a \in \Sigma}, \mathcal{V}_0, \mathcal{V}_F)$, where $\to_a \subseteq V_a \times V_a$ is the *local transition* *relation* for a, and $\mathcal{V}_0, \mathcal{V}_F \subseteq V_{\mathcal{P}}$ are sets of *initial* and *final* global states. Each relation $\rightarrow_a$ specifies how the processes $\theta(a)$ that meet on a may decide on a joint move. Other processes do not change their state. Thus we define the *global transition relation* $\Rightarrow \subseteq V_{\mathcal{P}} \times \Sigma \times V_{\mathcal{P}}$ by $\vec{v} \stackrel{a}{\Longrightarrow} \vec{v}'$ if $\vec{v}_a \rightarrow_a \vec{v}'_a$ and $\vec{v}_{\overline{a}} = \vec{v}'_{\overline{a}}$.

$\mathfrak{A}$ is called *deterministic* if the global transition relation of $\mathfrak{A}$ is a function from $V_{\mathcal{P}} \times \Sigma$ to $V_{\mathcal{P}}$ and if the set of initial states $\mathcal{V}_0$ is a singleton.

Runs Let $u \in \Sigma^*$ be of length of m. It is convenient to think of u as a function $u : [1..m] \rightarrow \Sigma$, where for $i \leq j$, $[i..j]$ abbreviates the set $\{i, i{+}1, \ldots, j\}$.

A *(global) run* of $\mathfrak{A}$ on u is a function $\rho : [0..m] \rightarrow V_{\mathcal{P}}$ such that $\rho(0) \in \mathcal{V}_0$ and for $i \in [1..m]$, $\rho(i-1) \stackrel{u(i)}{\Longrightarrow} \rho(i)$.

The word u is *accepted* by $\mathfrak{A}$ if there is a run ρ of $\mathfrak{A}$ on u such that $\rho(m) \in \mathcal{V}_F$. $L(\mathfrak{A})$, the language recognized by $\mathfrak{A}$, is the set of words accepted by $\mathfrak{A}$.

The problem For a given non-deterministic asynchronous automaton $\mathfrak{A}$ over (Σ, θ), construct a deterministic asynchronous automaton $\mathfrak{B}$ over (Σ, θ), such that $L(\mathfrak{A}) = L(\mathfrak{B})$.

For sequential finite automata, the determinization problem can be solved using the classical subset construction of Rabin and Scott. At the end of any word u, the subset automaton maintains the set of possible states that the original (non-deterministic) automaton could be in after reading u.

To determinize a non-deterministic asynchronous automaton $\mathfrak{A}$, we have to modify the subset construction to work at the level of local states. The main hurdle is the following: In general, the set of reachable global states of $\mathfrak{A}$ after reading u is *not* just the product of the subsets of reachable local states of the individual processes.

For instance, suppose we have two processes p and q with local state spaces $V_p = \{0_p, 1_p, 2_p\}$ and $V_q = \{0_q, 1_q, 2_q\}$. Let a be a letter such that $\theta(a) = \{p, q\}$ and $\rightarrow_a = \{(\langle 0_p, 0_q\rangle, \langle 1_p, 1_q\rangle), (\langle 0_p, 0_q\rangle, \langle 2_p, 2_q\rangle)\}$. Suppose p and q start in the local states 0_p and 0_q respectively. After reading a, the state of p could be either 1_p or 2_p. Similarly, local state of q could either be 1_q or 2_q. However, the set of possible $\{p, q\}$-states after reading a is *not* the naïve product $\{1_p, 2_p\} \times \{1_q, 2_q\}$— the only reachable $\{p, q\}$-states are $\langle 1_p, 1_q\rangle$ and $\langle 2_p, 2_q\rangle$.

So, to determinize an asynchronous automaton $\mathfrak{A}$, we need to record more than just the subset of reachable local states for each process in order to keep track of the valid global states of $\mathfrak{A}$. We also have to "remember" how each current local state arose. We then need to compose these *histories* so that each consistent composition of histories yields a valid global state of $\mathfrak{A}$ and, moreover, *all* the reachable global states of $\mathfrak{A}$ can be obtained in this manner.

2 Local and global views

We represent words as labelled partial orders. The notions we use are essentially those of trace theory [Maz].

Events With $u : [1..m] \rightarrow \Sigma$, we associate a set of *events* $\mathcal{E}_u$. Each event e is of the form $(i, u(i))$, where $i \in [1..m]$. In addition, we define an *initial event* denoted 0. The initial event marks the beginning when all processes synchronize and agree on an initial global state. Usually, we will write $\mathcal{E}$ for $\mathcal{E}_u$. If $e = (i, a)$ is an event, then we may use e instead of a in abbreviations such as V_e, which stands for V_a, i.e., $V_{\theta(a)}$, or $\rightarrow_e$, which is just $\rightarrow_a$. For $p \in \mathcal{P}$ and $e \in \mathcal{E}$, we write $p \in e$ to denote that $p \in \theta(u(i))$ when $e = (i, u(i))$; for $e = 0$, we define $p \in e$ to hold for all $p \in \mathcal{P}$. If $p \in e$, then we say that e is a *p-event*.

Ordering relations on $\mathcal{E}$ The word u imposes a total order on events: define $e < f$ if $e \neq f$ and either $e = 0$ or $e = (i, u(i))$, $f = (j, u(j))$, and $i < j$. We write $e \leq f$ if $e = f$ or $e < f$. Moreover, each process p orders the events in which it participates: define $\lessdot_p$ to be the strict ordering

$$e \lessdot_p f \textit{ iff } e < f,\ p \in e \cap f \textit{ and for all } e < g < f,\ p \notin g.$$

The set of all p-events in $\mathcal{E}$ is totally ordered by $\lessdot_p^*$, the reflexive, transitive closure of $\lessdot_p$.

Define $e \sqsubset f$ if for some p, $e \lessdot_p f$ and $e \sqsubseteq f$ if $e = f$ or $e \sqsubset f$. The *causality relation* $\sqsubseteq^*$ is the transitive closure of $\sqsubseteq$. If $e \sqsubseteq^* f$ then we say that e is *below* f. Note that 0 is below any event. The set of events below e is denoted $e\!\downarrow$. These represent the only synchronizations in $\mathcal{E}$ that may have affected the state of the processes in e when e occurs. The *neighbourhood* of e, $nbd(e)$, consists of e together with all its "$\sqsubset$-predecessors"—i.e., $nbd(e) = \{e\} \cup \{f \mid f \sqsubset e\}$.

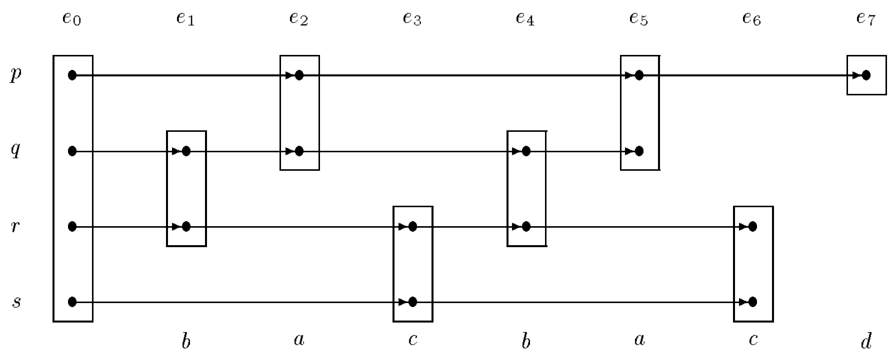

Fig. 1. An example

EXAMPLE 1. *Consider the word bacbacd over the alphabet (Σ, θ) for $\mathcal{P} = \{p, q, r, s\}$, where $\Sigma = \{a, b, c, d\}$ and $\theta(a) = \{p, q\}$, $\theta(b) = \{q, r\}$, $\theta(c) = \{r, s\}$ and $\theta(d) = \{p\}$. The set of events $\mathcal{E}_u$ is then $\{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7\} = \{0, (1, b), (2, a), (3, c), (4, b), (5, a), (6, c), (7, d)\}$.*

Figure 1 describes $(\mathcal{E}_u, \sqsubseteq^*)$. The arrows between the events indicate the relations $\lessdot_p$, $\lessdot_q$, $\lessdot_r$ and $\lessdot_s$. For example, $e_0 \lessdot_r e_1$ holds, but $e_0 \lessdot_p e_1$ does not hold.

The set of events $e_7{\downarrow}$ *is* $\{e_0, e_1, e_2, e_3, e_4, e_5, e_7\}$ *while* $e_6{\downarrow}$ *is* $\{e_0, e_1, e_2, e_3, e_4, e_6\}$. *Thus* $e_6{\downarrow} \cup e_7{\downarrow} = \mathcal{E}$. *The neighbourhood of* e_5, $nbd(e_5)$, *is* $\{e_2, e_4, e_5\}$.

Ideals $I \subseteq \mathcal{E}$ is called an *(order) ideal* if I is closed with respect to $\sqsubseteq^*$—i.e., $e \in I$ and $f \sqsubseteq^* e$ implies $f \in I$ as well.

Clearly, the entire set $\mathcal{E}$ is an ideal, as is $e{\downarrow}$ for any $e \in \mathcal{E}$.

P-views Let I be an ideal. The $\sqsubseteq^*$-maximum p-event in I is denoted $max_p(I)$. The *p-view of* I is the set $I|_p = max_p(I){\downarrow}$. So, $I|_p$ is the set of all events in I which p can "see". For $P \subseteq \mathcal{P}$, the P-view of I, denoted $I|_P$, is $\bigcup_{p\in P} I|_p$. Notice that $I|_P$ is always an ideal. In particular, we have $I|_{\mathcal{P}} = I$.

EXAMPLE 2. *In the example of Figure 1,* $max_r(\mathcal{E}) = e_6$. *So* $\mathcal{E}|_r = e_6{\downarrow} = \{e_0, e_1, e_2, e_3, e_4, e_6\}$. *On the other hand,* $max_p(\mathcal{E}) = e_7$ *and* $\mathcal{E}|_p = e_7{\downarrow} = \{e_0, e_1, e_2, e_3, e_4, e_5, e_7\}$.

3 Local runs and histories

Local runs Let I be an ideal. A *local run* on I is a function r that assigns to each $e \in I$ an e-state—i.e., a state in V_e—such that $r(0) \in \mathcal{V}_0$ and for all $e \neq 0$, r is consistent with $\rightarrow_e$ in $nbd(e)$. In other words, for $e \neq 0$ we have $\vec{v} \rightarrow_e r(e)$, where $\vec{v}$ is the e-state such that for all $q \in e$, $\vec{v}_q = r(f)_q$, where $f \lessdot_q e$.

So, a local run on $\mathcal{E}$ assigns an e-state to each $e \in \mathcal{E}$ in such a way that all neighbourhoods in $\mathcal{E}$ are consistently labelled. Let $\mathcal{R}(I)$ denote the set of all local runs on I. The following is easy to verify.

Proposition 1. *Given* $u : [1..m] \rightarrow \Sigma$, *there is a 1-1 correspondence between* $\mathcal{R}(\mathcal{E}_u)$, *the set of local runs on* $\mathcal{E}_u$, *and the set of global runs on* u.

EXAMPLE 3. *Let* $\mathfrak{A} = (\{V_p\}_{p\in\mathcal{P}}, \{\rightarrow_a\}_{a\in\Sigma}, \mathcal{V}_0, \mathcal{V}_F)$ *be an asynchronous automaton over* (Σ, θ), *where* $\mathcal{P}$ *and* (Σ, θ) *are as in our previous example. Each process has four local states. Thus,* $V_p = \{1_p, 2_p, 3_p, 4_p\}$, $V_q = \{1_q, 2_q, 3_q, 4_q\}$ *etc.*

Let the local transition relations of $\mathfrak{A}$ *be defined as in the table below:*

$\rightarrow_a$	$\rightarrow_b$	$\rightarrow_c$	$\rightarrow_d$
$\{(\langle 1_p, 2_q\rangle, \langle 3_p, 3_q\rangle)$	$\{(\langle 1_q, 1_r\rangle, \langle 2_q, 2_r\rangle)$	$\{(\langle 2_r, 1_s\rangle, \langle 4_r, 4_s\rangle)$	$\{(\langle 3_p\rangle, \langle 4_p\rangle)$
$(\langle 1_p, 3_q\rangle, \langle 4_p, 4_q\rangle)$	$(\langle 1_q, 1_r\rangle, \langle 3_q, 3_r\rangle)$	$(\langle 2_r, 3_s\rangle, \langle 4_r, 4_s\rangle)$	$(\langle 4_p\rangle, \langle 3_p\rangle)\}$
$(\langle 3_p, 2_q\rangle, \langle 4_p, 4_q\rangle)$	$(\langle 3_q, 4_r\rangle, \langle 2_q, 2_r\rangle)$	$(\langle 2_r, 4_s\rangle, \langle 1_r, 1_s\rangle)$	
$(\langle 4_p, 2_q\rangle, \langle 3_p, 3_q\rangle)\}$	$(\langle 4_q, 3_r\rangle, \langle 2_q, 2_r\rangle)\}$	$(\langle 3_r, 1_s\rangle, \langle 3_r, 3_s\rangle)\}$	

$\mathcal{V}_0 = \{\langle 1_p, 1_q, 1_r, 1_s\rangle\}$ *and* $\mathcal{V}_F = \{\langle 4_p, 3_q, 1_r, 1_s\rangle, \langle 3_p, 4_q, 4_r, 4_s\rangle\}$.

Then, the local runs corresponding to the only two possible global runs of $\mathfrak{A}$ *on* $u = bacbacd$ *are shown in Figure 2. The left half of each event is labelled by the first run and the right half by the second run. Neither run leads to a state in* $\mathcal{V}_F$ *so* $\mathfrak{A}$ *does not accept* u.

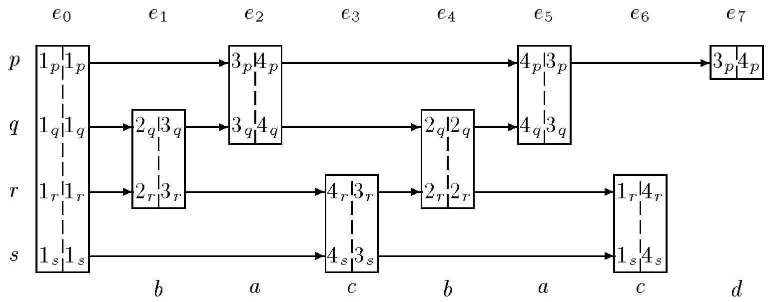

Fig. 2. Local runs

Histories Let I be an ideal. A *history* on I is a partial function h which assigns states to some of the events in I; i.e., $domain(h) \subseteq I$ and $h(e) \in V_e$ for each $e \in domain(h)$. A history h is *reachable* if there is some local run r on I such that $h(e) = r(e)$ for all $e \in domain(h)$. Let $\mathcal{H}(I)$ denote the set of all histories on I. Clearly, every local run on I is a history; i.e., $\mathcal{R}(I) \subseteq \mathcal{H}(I)$.

Choices Let I be an ideal. Given a collection $\{H_p\}_{p\in P}$ of sets of histories on the p-views $I|_p$, a *P-choice* $\{h_p\}_{p\in P}$ of $\{H_p\}_{p\in P}$ assigns to each $p \in P$, a history h_p from H_p. The choice is *consistent* if for each $p, q \in P$, for every $e \in domain(h_p) \cap domain(h_q)$, $h_p(e) = h_q(e)$.

Let $\{H_p\}_{p\in P}$ be a collection of sets of histories for $P \subseteq \mathcal{P}$. We define the product

$$\bigotimes_{p\in P} H_p = \{h \in \mathcal{H}(I|_P) \mid \textit{There exists a consistent } P\textit{-choice } \{h_p\}_{p\in P} \textit{ of } \{H_p\}_{p\in P} \textit{ such that } domain(h) = \textstyle\bigcup_p domain(h_p) \textit{ and } \forall p \in P.\ \forall e \in domain(h_p).\ h(e) = h_p(e)\}.$$

So, $\bigotimes_{p\in P} H_p$ contains all the histories on $I|_P$ which may be pieced together using mutually consistent histories from the sets H_p.

EXAMPLE 4. *Consider the local runs shown in Figure 2. For p, let h_p^1 and h_p^2 be two histories where*
$h_p^1 = \{(e_3 \mapsto \langle 4_r, 4_s\rangle, e_4 \mapsto \langle 2_q, 2_r\rangle, e_5 \mapsto \langle 4_p, 4_q\rangle, e_7 \mapsto \langle 3_p\rangle)\}$, *and*
$h_p^2 = \{(e_3 \mapsto \langle 3_r, 3_s\rangle, e_4 \mapsto \langle 2_q, 2_r\rangle, e_5 \mapsto \langle 3_p, 3_q\rangle, e_7 \mapsto \langle 4_p\rangle)\}$.
For s, let h_s^1 and h_s^2 be two histories where
$h_s^1 = \{(e_2 \mapsto \langle 3_p, 3_q\rangle, e_4 \mapsto \langle 2_q, 2_r\rangle, e_6 \mapsto \langle 4_r, 4_s\rangle)\}$, *and*
$h_s^2 = \{(e_2 \mapsto \langle 4_p, 4_q\rangle, e_4 \mapsto \langle 2_q, 2_r\rangle, e_6 \mapsto \langle 1_r, 1_s\rangle)\}$.
All four of these histories are reachable. There are four possible $\{p, s\}$-choices for $\{\{h_p^1, h_p^2\}, \{h_s^1, h_s^2\}\}$. Each of these choices is consistent. However, only two of the runs in $\{h_p^1, h_p^2\} \bigotimes \{h_s^1, h_w^2\}$ are reachable—those generated by the choices (h_p^1, h_s^1) and (h_p^2, h_s^2). The "bad" choice (h_p^2, h_s^1) implies that $\langle 4_p, 3_q, 1_r, 1_s\rangle$ is a valid global state of $\mathfrak{A}$ after u, which leads to the erroneous conclusion that $\mathfrak{A}$ accepts u. The task is to rule out such bad choices by maintaining reachable histories whose products are also reachable.

In particular, for $P \subseteq \mathcal{P}$ we may form the product $\bigotimes_{p\in P} \mathcal{R}(I|_p)$, which generates the set $\mathcal{R}(I|_P)$ of local runs on $I|_P$.

Lemma 2. *Let I be an ideal and $P \subseteq \mathcal{P}$. Then, $\mathcal{R}(I|_P) = \bigotimes_{p\in P} \mathcal{R}(I|_p)$.*

Proof The fact that $\mathcal{R}(I|_P) \subseteq \bigotimes_{p\in P} \mathcal{R}(I|_p)$ is obvious.

To show the converse, let $r \in \bigotimes_{p\in P} \mathcal{R}(I|_p)$ where the consistent P-choice is $\{r_p\}_{p\in P}$. Clearly $domain(r) = I|_P$. We just have to check that $nbd(e)$ is labelled consistently by r for each $e \in I|_P$. But, if $e \in I|_P$ then $e \in I|_p$ for some $p \in P$ and so $nbd(e) \subseteq I|_p$ as well. Since $r = r_p$ on $I|_p$ and r_p is a local run on $I|_p$, r_p must have assigned consistent values to $nbd(e)$. □

4 Finite histories and frontiers

Frontiers Let I be an ideal and $p, q, s \in \mathcal{P}$. We say that event $e \in I|_p$ is an *s-sentry* for p with respect to q if $e \in I|_p \cap I|_q$ and $e \lessdot_s f$ for some $f \in I|_q - I|_p$. Thus e is an event known to p and q whose s-successor is known only to q.

Define $frontier_{pq}(I)$ to be the set of all s-sentries which exist for p with respect to q. Observe that this definition is asymmetric—$frontier_{pq}(I) \neq frontier_{qp}(I)$.

Example 5. *In the example of Figure 1, $\mathcal{E}|_p \cap \mathcal{E}|_s = \{e_0, e_1, e_2, e_3, e_4\}$. Then $frontier_{ps}(\mathcal{E}) = \{e_3, e_4\}$, whereas $frontier_{sp}(\mathcal{E}) = \{e_2, e_4\}$. So, e_4 belongs to both frontiers—it is an r-sentry in $frontier_{ps}(\mathcal{E})$ and a q-sentry in $frontier_{sp}(\mathcal{E})$.*

In general, $e \in frontier_{pq}(I)$ could simultaneously be an s-sentry for several different s. However, $frontier_{pq}(I)$ is always a bounded set, since for each $s \in \mathcal{P}$ there is at most one s-sentry $e \in frontier_{pq}(I)$.
For $P \subseteq \mathcal{P}$ and $p \in P$, define the P-frontier of p at I to be the set

$$\bigcup_{q\in P-\{p\}} frontier_{pq}(I) \cup frontier_{qp}(I).$$

Lemma 3. *Let I be an ideal and $\{h_p\}_{p\in P}$ be a consistent P-choice of $\{\mathcal{H}(I|_p)\}_{p\in P}$ such that for each $p \in P$,*
(i) h_p is reachable; and
(ii) the P-frontier of p is included in $domain(h_p)$.
Then $\bigotimes_{p\in P}\{h_p\}$ is a reachable history in $\mathcal{H}(I|_P)$.

Proof Order the processes in P as $p_1, p_2, \ldots, p_k$. For $i \in [1..k]$, let $P_i = \bigcup_{j\in[1..i]}\{p_j\}$. By assumption, for each p_i, h_{p_i} is a reachable history. So, we have a local run r_{p_i} on $I|_{p_i}$ which agrees with h_{p_i} on $domain(h_{p_i})$. To show that $h = \bigotimes_{p\in P}\{h_p\}$ is reachable, we must construct a local run r on $I|_P$ which agrees with h on $domain(h) = \bigcup_{p\in P} domain(h_p)$.

Define r as follows:

- For all $e \in I|_{p_1}$, $r(e) = r_{p_1}(e)$.

– For $i \in [2..k]$, for all $e \in I|_{p_i} - I|_{P_{i-1}}$, $r(e) = r_{p_i}(e)$.

So, we "sweep across" $I|_P$ starting from $I|_{p_1}$ and ending at $I|_{p_k}$, assigning states according to $r_{p_1}, r_{p_2}, \ldots, r_{p_k}$ in k "stages". Clearly $domain(r) = I|_P$ and r agrees with h on $domain(h)$. We have to show that r is a local run; i.e., we have to show that r is consistent with $\rightharpoonup_e$ across $nbd(e)$ for each $e \in I|_P$.

Let $e \in I|_P$. We know that $r(e)$ was assigned at some stage $i \in [1..k]$. Clearly, $e \in I|_{p_i}$ and so $nbd(e) \subseteq I|_{p_i}$ as well. If $nbd(e) \subseteq I|_{p_i} - I|_{P_{i-1}}$, then all the events in $nbd(e)$ are assigned r values at stage i according to r_{p_i}. Since r_{p_i} is a local run on I, these values must be consistent with $\rightharpoonup_e$.

The crucial case is when some $f \in nbd(e)$ lies in $I|_{P_{i-1}}$ and so has already been assigned a value. But then $f \in I|_{P_{i-1}} \cap I|_{p_i}$ which is the same as $\bigcup_{j \in [1..i-1]}(I|_{p_j} \cap I|_{p_i})$. In other words, for some p_ℓ, $\ell \in [1..i{-}1]$, f belongs to $frontier_{p_\ell p_i}(I)$. So $f \in domain(h_{p_\ell}) \cap domain(h_{p_i})$, by assumption. Therefore, the value $r(f)$ must agree with $h_{p_\ell}(f) = h_{p_i}(f)$ and hence must agree with $r_{p_i}(f)$ as well. So, even though $f \in nbd(e)$ has already been assigned a value before stage i, the value agrees with r_{p_i}. Thus, effectively, $nbd(e)$ is assigned values as given by r_{p_i} and these must be consistent with $\rightharpoonup_e$ since r_{p_i} is a local run on I. □

This is a finite version of Lemma 2 above. Suppose that at the end of a word u, each process p maintains all reachable histories on a finite (bounded) set of events spanning the $\mathcal{P}$-frontier of p in $\mathcal{E}_u$. Then, by the previous lemma, the product of these histories will generate all the reachable global states of $\mathfrak{A}$ after u. Further, the number of possible histories on a bounded set is also finite.

The problem now is with maintaining frontier information locally—i.e., how can a process p compute and locally update its frontier? This is done using slightly larger, but still bounded, sets of events called primary and secondary information, which between them subsume the frontier. It turns out that these sets can be updated locally with each synchronization between processes. These then will be the domains of the histories maintained by each process.

5 Primary and secondary information

Primary information Let I be an ideal and $p, q \in \mathcal{P}$. Then $latest_{p\to q}(I)$ denotes the maximum q-event in $I|_p$. So, $latest_{p\to q}(I)$ is the latest q-event in I that p knows about. The *primary information* of p after I, $primary_p(I)$, is the set $\{latest_{p\to q}(I)\}_{q\in\mathcal{P}}$. As usual, for $P \subseteq \mathcal{P}$, $primary_P(I) = \bigcup_{p\in P} primary_p(I)$.

Secondary information The *secondary information* of p after I, $secondary_p(I)$, is the set $\bigcup_{q\in\mathcal{P}} primary_q(latest_{p\to q}(I){\downarrow})$. In other words, this is the latest information that p has in I about the primary information of q, for each $q \in \mathcal{P}$. Once again, for $P \subseteq \mathcal{P}$, $secondary_P(I) = \bigcup_{p\in P} secondary_p(I)$.

Each event in $secondary_p(I)$ is of the form $latest_{q\to s}(latest_{p\to q}(I)\downarrow)$ for some $q, s \in \mathcal{P}$. This is the latest s-event which q knows about upto the event $latest_{p\to q}(I)$. We abbreviate $latest_{q\to s}(latest_{p\to q}(I){\downarrow})$ by $latest_{p\to q\to s}(I)$. Notice that each primary event $latest_{p\to q}(I)$ is also a secondary event $latest_{p\to p\to q}(I)$.

EXAMPLE 6. *In Figure 1,* $latest_{s \to p}(\mathcal{E}) = e_2$ *whereas* $latest_{p \to s}(\mathcal{E}) = e_3$*. Also,* $latest_{s \to p \to r}(\mathcal{E}) = e_1$ *while* $latest_{p \to s \to r}(\mathcal{E}) = e_3$*.*

The following result, which is proved in [KMS, MS], lets us identify all the frontier events in $\mathcal{E}$ using primary and secondary information.

Lemma 4. *Let I be an ideal, $p, q, s \in \mathcal{P}$ and $e \in frontier_{pq}(I)$ an s-sentry. Then $e = latest_{p \to s}(I)$. Also, for some $s' \in \mathcal{P}$, $e = latest_{q \to s' \to s}(I)$.*

So, for every $p \in \mathcal{P}$ and $u \in \Sigma^*$, each process p maintains all reachable histories over the finite set $secondary_p(\mathcal{E}_u)$. By the preceding lemma, this set includes all events in the $\mathcal{P}$-frontier of $\mathcal{E}_u$ as well as the maximal event $max_p(\mathcal{E}_u) = latest_{p \to p \to p}(\mathcal{E}_u)$.

We now need to show that these sets may be updated locally—i.e., if $w = ua$, then $secondary_p(\mathcal{E}_w)$ may be computed from $secondary_p(\mathcal{E}_u)$ for each process $p \in a$ using only the information available with the processes in a. (Note that $secondary_{\mathcal{P}}(\mathcal{E}_{ua})$ is a subset of $secondary_{\mathcal{P}}(\mathcal{E}_u)$ together with the new a-event.)

Since we are manipulating events, in general, we require a mechanism for assigning "unambiguous" labels to the events in $\mathcal{E}_u$ while u is being read. In our context, "unambiguous" means that distinct events in $secondary_{\mathcal{P}}(\mathcal{E}_u)$ are assigned distinct labels. It turns out that we can maintain such a labelling using a finite set of labels $\mathcal{L}$ and simultaneously update the labels assigned to $secondary_{\mathcal{P}}(\mathcal{E}_u)$ in a consistent way with each local synchronization. This involves running the "gossip automaton" [MS] in the background. We will suppress the details here but assume that there is an asynchronous automaton $\mathfrak{A}_G$ with the same structure as the automaton $\mathfrak{A}$ which works as follows.

Let $u \in \Sigma^*$. Each process p in $\mathfrak{A}_G$ inductively maintains a labelling function $\lambda_p(u) : \mathcal{P} \times \mathcal{P} \to \mathcal{L}$ such that:

1. *Let $q, s \in \mathcal{P}$. Then for all $q', q'', s', s'' \in \mathcal{P}$, $\lambda_q(u)(q', q'') = \lambda_s(u)(s', s'')$ iff $latest_{q \to q' \to q''}(\mathcal{E}_u) = latest_{s \to s' \to s''}(\mathcal{E}_u)$.*
2. *If $w = ua$ and $q, q', q'', s, s', s'' \in \mathcal{P}$ such that $latest_{q \to q' \to q''}(\mathcal{E}_w) = latest_{s \to s' \to s''}(\mathcal{E}_u)$ then $\lambda_q(w)(q', q'') = \lambda_s(u)(s', s'')$.*

So, after reading u, each process p in $\mathfrak{A}_G$ maintains a labelling $\lambda_p(u)$ of $secondary_p(\mathcal{E}_u)$ such that, across the system, distinct events in $secondary_{\mathcal{P}}(\mathcal{E}_u)$ are assigned distinct labels. The map $\lambda_p(u)$ is a finite object and can be incorporated into the local state of p. Further, all the processes which synchronize on an action a can consistently update their primary and secondary information and extend the labellings $\{\lambda_p(u)\}_{p \in a}$ to new labellings $\{\lambda_p(ua)\}_{p \in a}$ which maintain the inductive assertion. The second condition ensures that the new labellings $\{\lambda_p(ua)\}_{p \in a}$ do not reassign fresh labels to "old" events in $secondary_{\mathcal{P}}(\mathcal{E}_{ua})$.

6 The determinization algorithm

We are now ready to present our deterministic asynchronous automaton $\mathfrak{B} = (\{V_p^{\mathfrak{B}}\}_{p \in \mathcal{P}}, \{\to_a^{\mathfrak{B}}\}_{a \in \Sigma}, \mathcal{V}_0^{\mathfrak{B}}, \mathcal{V}_F^{\mathfrak{B}})$ corresponding to our original non-deterministic asynchronous automaton $\mathfrak{A}$ such that $L(\mathfrak{A}) = L(\mathfrak{B})$. $\mathfrak{B}$ will incorporate the gossip automaton $\mathfrak{A}_G$.

Formally, a state in $V_p^{\mathfrak{B}}$ consists of the following information:

- A labelling $\lambda_p : (\mathcal{P} \times \mathcal{P}) \to \mathcal{L}$.
- A set of histories $\mathcal{RH}_p$ where each $h \in \mathcal{RH}_p$ is a partial function on $\mathcal{L}$ that maps a label ℓ to a P-state (where $P \subseteq \mathcal{P}$ is the set of processes that participated in the event that ℓ was assigned to) such that $domain(h) = range(\lambda_p)$.

Intuitively, after reading a word u, process p has computed λ_p as the labelling of $secondary_p(\mathcal{E}_u)$ given by the gossip automaton $\mathfrak{A}_G$. Thus, the label $\lambda_p(q,r)$ represents the secondary event $latest_{p \to q \to r}(\mathcal{E}_u)$.

The set $\mathcal{RH}_p$ is supposed to contain all reachable histories over the secondary events $secondary_p(\mathcal{E}_u)$. Since distinct events in $secondary_p(\mathcal{E}_u)$ are assigned distinct labels, each history in $\mathcal{RH}_p$ is a function from $range(\lambda_p)$ to P-states.

Initially, each $p \in \mathcal{P}$ stores the following:

- For each pair of processes $q, r \in \mathcal{P}$, $\lambda_p(q,r) = \ell_0$, where ℓ_0 is some arbitrary but fixed label from $\mathcal{L}$.
- For each initial state $\vec{v}$ of $\mathfrak{A}$, there is a history h in $\mathcal{RH}_p$ such that $h(\ell_0) = \vec{v}$.

The initial state $\mathcal{V}_0^{\mathfrak{B}}$ is the product of the initial states of the individual processes. We now describe the transition rules $\{\to_a^{\mathfrak{B}}\}_{a \in \Sigma}$. Suppose $\mathfrak{B}$ reads a when the global state of $\mathfrak{B}$ is $\{\langle \lambda'_p, \mathcal{RH}'_p \rangle\}_{p \in \mathcal{P}}$. Then the local states of processes in a are updated as follows.

- For each $p \in a$, construct a new labelling function $\lambda''_p : \mathcal{P} \times \mathcal{P} \to \mathcal{L}$. This is done by the gossip automaton, $\mathfrak{A}_G$. For the new event e_a, let $\lambda''_p(e_a) = \ell_a$.
- Compute new histories $\mathcal{RH}''_p$ for each $p \in a$ as follows. Consider $h_a \in \bigotimes_{p \in a}\{\mathcal{RH}'_p\}$. Let $\vec{v}$ be the global a-state corresponding to h_a—i.e., $\vec{v}_p = (h_p(\lambda'_p(p,p)))_p$ for each $p \in a$ (recall that $latest_{p \to p \to p}(I) = max_p(I)$). Let $\mathcal{V}_{h_a} = \{\vec{v}' \mid \vec{v} \to_a \vec{v}'\}$. So, $\mathcal{V}_{h_a}$ is the set of all possible a-states v' which can be used to extend h_a to cover the new event e_a so that $nbd(e_a)$ is consistently labelled with respect to $\to_a$.

  Each element $\vec{v}' \in \mathcal{V}_{h_a}$ together with h_a generates a history h'_p in $\mathcal{RH}''_p$:

  $$\forall \ell \in range(\lambda''_p).\ h'_p(\ell) = \begin{cases} \vec{v}' & if\ \ell = \ell_a \\ h_a(\ell) & otherwise \end{cases}$$

  So, the new a-event is assigned the a-tuple $\vec{v}'$ while the other secondary events of p (after reading a) inherit their h'_p values from h_a. Repeat this procedure for each $h_a \in \bigotimes_{p \in)}\{\mathcal{RH}'_p\}$ to generate the entire set $\mathcal{RH}''_p$ for each $p \in a$.

We now define the final states of $\mathfrak{B}$. Let $\vec{\sigma}$ be a global state of $\mathfrak{B}$, where $\vec{\sigma}_p = \langle \lambda'_p, \mathcal{RH}'_p \rangle$ for each $p \in \mathcal{P}$. Each history $h \in \bigotimes_{p \in \mathcal{P}} \mathcal{RH}'_p$ gives rise to a global state $\vec{v}$ of $\mathfrak{A}$ as follows: for each $p \in \mathcal{P}$, $\vec{v}_p = (h(\lambda'_p(p,p)))_p$. Let $subset(\vec{\sigma})$ denote the set of global states of $\mathfrak{A}$ generated from $\vec{\sigma}$ in this manner. Then we can define $\mathcal{V}_F^{\mathfrak{B}} = \{\vec{\sigma} \mid subset(\vec{\sigma}) \cap \mathcal{V}_F \neq \emptyset\}$.

Theorem 5. $L(\mathfrak{A}) = L(\mathfrak{B})$.

Proof For $u \in \Sigma^*$, let $\vec{\sigma}$ be the global state of $\mathfrak{B}$ after reading u such that $\vec{\sigma}_p = \langle \lambda''_p, \mathcal{RH}''_p \rangle$ for each $p \in \mathcal{P}$. We claim the following:

Claim For each $p \in \mathcal{P}$, $\mathcal{RH}''_p$ is precisely the set of all reachable histories on the set of events $secondary_p(\mathcal{E}_u)$.

Assuming the claim, we know from Proposition 1 and Lemmas 2, 3 and 4 that the global states in $subset(\vec{\sigma})$ are precisely the global states that $\mathfrak{A}$ could be in after u. So, $\mathfrak{B}$ accepts u iff $subset(\vec{\sigma}) \cap \mathcal{V}_F \neq \emptyset$ iff there is a run of $\mathfrak{A}$ on u leading to a final state iff $\mathfrak{A}$ accepts u and we are done.

Proof of Claim To prove the claim, we proceed by induction on $|u|$.

($|u| = 0$). Then $u = \varepsilon$, the empty word. The claim is trivially true at this state since all the secondary events in $\mathcal{E}_u$ are the initial event 0 and each process maintains a set of histories which assigns all possible initial states of $\mathfrak{A}$ to the initial event.

($|u| > 0$). Let $u = wa$ and assume inductively that after reading w, the local states $\{\langle \lambda'_p, \mathcal{RH}'_p \rangle\}_{p \in \mathcal{P}}$ satisfy the Claim. We have to argue that the procedure for updating the local states of $p \in a$ maintains the property asserted in the Claim.

Now, assume $\mathcal{RH}'_p$ contains all reachable histories over $secondary_p(\mathcal{E}_w)$ for each $p \in a$. The gossip automaton $\mathfrak{A}_G$ guarantees that λ''_p labels precisely the events in $secondary_p(\mathcal{E}_u)$. We have to show that $\mathcal{RH}''_p$ contains all reachable histories over $secondary_p(\mathcal{E}_u)$.

For all $p \in a$, $\mathcal{E}_u|_p = \mathcal{E}_w|_a \cup \{e_a\}$, where e_a is the new a-event. So, any local run on $\mathcal{E}_u|_p$ consists of a local run on $\mathcal{E}_w|_a$ extended to cover e_a such that $nbd(e_a)$ is consistently labelled.

We argue that the product $\bigotimes_{p\in a} \mathcal{RH}'_p$ is precisely the projection of $\mathcal{R}(\mathcal{E}_w|_a)$ onto $secondary_a(\mathcal{E}_w)$. By Lemma 3, every history $h \in \bigotimes_{p\in a} \mathcal{RH}'_p$ is a reachable history on $\mathcal{E}_w|_a$ and so is the projection of some local run on $\mathcal{E}_w|_a$ onto $secondary_a(\mathcal{E}|_w)$.

Conversely, consider any local run r on $\mathcal{E}_w|_a$. Decompose r into local runs r_p over $\mathcal{E}_w|_p$ for each $p \in a$ by looking at r restricted to $\mathcal{E}_w|_p$. Since $\mathcal{RH}'_p$ has all reachable histories on $secondary_p(\mathcal{E}_w)$, the projection h_p of r_p on $secondary_p(\mathcal{E}_w)$ belongs to $\mathcal{RH}'_p$. So, the projection of r onto $secondary_a(\mathcal{E}_w)$ belongs to $\bigotimes_{p\in a} \mathcal{RH}'_p$.

So, we can reconstruct all possible a-moves of $\mathfrak{A}$ after w by looking at $\bigotimes_{p\in a} \mathcal{RH}'_p$. The procedure for updating $\mathcal{RH}'_p$ to $\mathcal{RH}''_p$ in the definition of $\rightarrow^{\mathfrak{B}}_p$ then guarantees that $\mathcal{RH}''_p$ contains all reachable p-histories over $secondary_p(\mathcal{E}_u)$ for each $p \in a$. □

7 The complexity of determinization

Theorem 6. *Let* $\mathfrak{A} = (\{V_p\}_{p\in\mathcal{P}}, \{\rightarrow_a\}_{a\in\Sigma}, \mathcal{V}_0, \mathcal{V}_F)$ *be a non-deterministic asynchronous automaton with* N *processes such that* $max_{p\in\mathcal{P}}|V_p| = M$. *Then, in the* *corresponding deterministic automaton $\mathfrak{B}$ that we construct, each process has at most $2^{M^{O(N^3)}}$ states.*

Proof Each local state of $\mathfrak{B}$ is of the form $\langle \lambda_p, \mathcal{RH}_p \rangle$. From [MS], $\mathfrak{A}_G$ can maintain and update λ_p consistently using $O(N^3 \log N)$ bits.

We now need to maintain histories over secondary events. Each history h consists of N^2 P-states. Since a P-state can be written down using $N \log M$ bits, h can be written down using $N^3 \log M$ bits. The number of different histories possible over N^2 events is $\left(M^{O(N)}\right)^{N^2} = M^{O(N^3)}$. A set of histories can therefore be written down using $M^{O(N^3)}$ bits. So, in total we need $M^{O(N^3)} + O(N^3 \log N) = M^{O(N^3)}$ bits. □

The upper bound we describe above is essentially optimal because of the following double-exponential lower bound, which is proved in [KMS].

Theorem 7. *There is a sequence of languages L_{K^N} over distributed alphabets $(\Sigma_{K^N}, \theta_{K^N})$, $K, N \geq 2$, such that L_{K^N} is recognized by a non-deterministic asynchronous automaton with local state spaces and transition relations whose sizes are polynomial in K and N, whereas any deterministic asynchronous automaton recognizing L_{K^N} must have at least one process with $2^{K^N/N}$ states.*

References

[CMZ] R. Cori, Y. Metivier, W. Zielonka: Asynchronous mappings and asynchronous cellular automata, *Inf. and Comput.*, **106** (1993) 159–202.

[DM] V. Diekert, A. Muscholl: Deterministic asynchronous automata for infinite traces, *Proc. STACS '93*, *LNCS* **665** (1993) 617–628.

[GP] P. Gastin, A. Petit: Asynchronous cellular automata for infinite traces, *Proc. ICALP '92*, *LNCS* **623** (1992) 583–594.

[KMS] N. Klarlund, M. Mukund, M. Sohoni: Determinizing asynchronous automata, *Report DAIMI-PB 460*, Computer Science Department, Aarhus University, Aarhus, Denmark (1993).

[Maz] A. Mazurkiewicz: Basic notions of trace theory, in: J.W. de Bakker, W.-P. de Roever, G. Rozenberg (eds.), *Linear time, branching time and partial order in logics and models for concurrency*, *LNCS* **354**, (1989) 285–363.

[MS] M. Mukund, M. Sohoni: Gossiping, asynchronous automata and Zielonka's theorem, *Report TCS-94-2*, School of Mathematics, SPIC Science Foundation, Madras (1994). See also "Keeping track of the latest gossip: Bounded timestamps suffice", *Proc. FST&TCS '93, LNCS* **761** (1993) 388–399.

[Mus] A. Muscholl: On the complementation of Büchi asynchronous cellular automata, *Proc. ICALP 1994*.

[Zie1] W. Zielonka: Notes on finite asynchronous automata, *R.A.I.R.O.—Inf. Théor. et Appl.*, **21** (1987) 99–135.

[Zie2] W. Zielonka: Safe executions of recognizable trace languages, in *Logic at Botik*, *LNCS* **363** (1989) 278–289.