RWTH Aachen

Department of Computer Science
Technical Report

Languages of Infinite Traces and Deterministic Asynchronous Automata

Namit Chaturvedi

ISSN 0935–3232 · Aachener Informatik-Berichte · AIB-2014-04

RWTH Aachen · Department of Computer Science · February 2014 (revised)

Languages of Infinite Traces and Deterministic Asynchronous Automata

Namit Chaturvedi*

Lehrstuhl für Informatik 7
RWTH Aachen, Germany
Email: chaturvedi@automata.rwth-aachen.de

Abstract. In the theory of deterministic automata for languages of infinite words, a fundamental fact relates the family of infinitary limits of regular languages and the family of ω-languages recognized by deterministic Büchi automata. With the known definitions of asynchronous automata, this observation does not extend to the context of traces. A major difficulty is posed by processes that stall after finitely many transitions. We introduce the family of deterministic, synchronization-aware asynchronous automata which – using as parameter the set of processes that stay live ad infinitum – allows us to settle an open question, namely, whether there exists a deterministic Büchi automaton recognizing precisely the infinitary limit of a regular trace language. Also, the corresponding class of unparameterized Muller automata captures all ω-regular trace languages.

1 Introduction

Traces were introduced as models of concurrent computations of distributed systems by Antoni Mazurkiewicz, who later also provided the first explicit definition of infinite traces [4]. Zielonka established the correspondence between regular languages of finite traces and regular languages of finite words that are the "linearizations" of traces, and also characterized these as languages recognized by asynchronous automata [7] (see [3] for a gentle introduction). Intuitively, an asynchronous automaton comprises of a number of processes, and a mapping that assigns to each letter of the alphabet a set of processes that are responsible for jointly executing the transitions upon this letter. A transition relation defined for each letter of the alphabet associates with a tuple of states a set of tuples of states, where each tuple contains a state from every participating process. The acceptance condition is given in terms of global final states.

With regards to infinite traces, Gastin-Petit [2] define the class of recognizable languages of infinite traces in terms of monoid morphisms (we refer to this as the class of ω-regular trace languages). Then they show that this class coincides with the class of languages recognized by the family of suitably defined non-deterministic Büchi asynchronous cellular automata. Later, Diekert-Muscholl [1] present the family of deterministic asynchronous cellular Muller automata recognizing precisely the class of ω-regular trace languages. Muscholl also studies the languages recognized by the deterministic variant of Gastin-Petit's Büchi asynchronous cellular automata and by some other modifications thereof [5].

It is known that a language of infinite words over the alphabet Σ is deterministically Büchi recognizable if and only if it can be written in the form

* This work was supported in part by DFG Research Training Group-1298 AlgoSyn, and in part by the CASSTING Project funded by the European Union under the 7th Research Framework Programme.

$\mathsf{lim}(K) := \{\alpha \in \Sigma^\omega \mid \alpha$ has infinitely many prefixes in $K\}$ for some regular language $K \subseteq \Sigma^*$. Diekert-Muscholl generalize the definition of the operator lim for the case of trace languages [1]. Muscholl also provides a definition of the A-infinitary limit, an operator lim_A, where the parameter A reveals information about the set of letters that are allowed to appear infinitely often [5]. Every infinitary limit language can be expressed as a finite union of A-infinitary limit languages, whereas the reverse in not necessarily true. The languages accepted by the class of deterministic, asynchronous Büchi automata introduced by Muscholl can be expressed as finite unions of A-infinitary limit languages. But some fundamental questions are still unanswered:

For languages of infinite traces, does there exist a model of Büchi automata accepting precisely the class of finite unions of lim_A *languages? In particular, is every* lim *language accepted by such an automaton?*

We suspect that these questions remain open owing to the current definitions of models of asynchronous (cellular) automata. These models are oblivious to the fact that every run of an asynchronous automaton over an infinite trace reveals a maximal partitioning of the set of processes in a manner that each part is minimal and, after a finite prefix, processes belonging to one part never interact directly or indirectly with a process belonging to another part. The processes ought to infer this partition by observing their infinitely recurring interactions.

One possible reason why current models lack this power is that, similar to the case of word languages, they are straightforward adaptations of automata models recognizing languages of finite traces, which are not required to perform any infinitary inferencing. Additionally, different traces in a given language may induce infinite runs where different sets of processes remain *live* for infinitely many transitions. We observe that once we parameterize each acceptance tuple with a set of live processes, reasonable results emerge; for example, the characterization of linearizations of deterministic, real trace languages in terms of I-diamond Büchi (word) automata with "extended" acceptance conditions [5].

We introduce a model of asynchronous automaton, which we refer to as the *synchronization-aware asynchronous automaton*, that is capable of inferring the maximal partitioning of the set of processes induced by any infinite trace.

Upon fixing precisely the set of processes that remain live w.r.t. each acceptance tuple, we obtain the family of synchronization-aware Büchi automata which accepts precisely the family of finite unions of lim_A languages (see Thm. 28). This answers the above-mentioned open questions in the affirmative. At the same time, the family of synchronization-aware Muller automata that we introduce here recognize precisely the class of ω-regular trace languages (see Thm. 31). As is the norm, the tuples in the Muller acceptance condition in our model are not parameterized by the set of processes that may and must remain live.

The next section presents the concepts, definitions and terminology that we use in the paper. In Section 3 we describe our model of synchronization-aware asynchronous transition systems. We present a mechanism to construct such transition systems from arbitrary asynchronous transition systems. Then, upon equipping these with suitable acceptance conditions to obtain Büchi and Muller automata, we proceed establish our major results. Finally, we conclude with an outlook on some fundamental results that we believe are well within our grasp.

2 Preliminaries

2.1 Finite and Infinite Traces

Over a finite alphabet Σ, let $D \subseteq \Sigma^2$ be a binary, reflexive, and symmetric *dependence relation*. Whenever convenient, we also refer to the corresponding *independence relation* $I = \Sigma^2 \setminus D$. The *independence alphabet* is denoted by the pair (Σ, I). Over such an independence alphabet, a *finite trace* is an isomorphism class of directed acyclic graphs $t = [V, \lessdot, \lambda]$ where V is a finite set of events, $\lambda \colon V \to \Sigma$ is a labeling function, and for two distinct events $e, e' \in V, \lambda(e) D \lambda(e') \Leftrightarrow e \lessdot e'$ or $e' \lessdot e$ or $e = e'$. The *concatenation* of two finite traces $t_1 = [V_1, \lessdot_1, \lambda_1]$ and $t_2 = [V_2, \lessdot_2, \lambda_2]$ is given by $t_1 \odot t_2 = [V_1 \uplus V_2, \lessdot', \lambda_1 \uplus \lambda_2]$, where $\lessdot' = \lessdot_1 \uplus \lessdot_2 \uplus \{(e_1, e_2) \in V_1 \times V_2 \mid \lambda_1(e_1) D \lambda_2(e_2)\}$. Over an independence alphabet, (Σ, I), we denote the set of all finite traces with $\mathbb{M}(\Sigma, I)$.

For convenience, we work with a "simplified" view of traces $t = [V, \lessdot, \lambda]$ where we remove all edges that may be inferred from others, i.e. by $\lessdot$ we mean $\lessdot \setminus \lessdot^2$ as shown in Fig. 1a. We also refer to the partial order $<$ obtained from the transitive closure of the simplified edge relation. Relations $\leq$, $\gtrdot$, $\geq$, and $>$ are also defined in the natural manner. We use the abbreviation $e \in t$ to imply that $t = [V, \lessdot, \lambda]$ and $e \in V$.

An *infinite trace* is a directed acyclic graph $\theta = [V, \lessdot, \lambda]$ where V is a countable set of events, and λ and $\lessdot$ are like above except $\lessdot$ satisfies an additional requirement, namely, for each $e \in \theta$, the set $\{e' \in \theta \mid e' \leq e\}$ is finite. Denote the set of all infinite traces with $\mathbb{R}(\Sigma, I)$. For traces $t \in \mathbb{M}(\Sigma, I), \theta \in \mathbb{R}(\Sigma, I)$, we refer to sets $\mathsf{alph}(t), \mathsf{alph}(\theta)$ of letters occurring in them, and to the set $\mathsf{alphinf}(\theta)$ of letters occurring infinitely often in θ. Define concatenation $t \odot \theta$ as above.

We say t_1 is a *prefix* of t_2, i.e. $t_1 \sqsubseteq t_2 :\Leftrightarrow \exists t' : t_2 = t_1 \odot t'$, and $t_1 \sqsubset t_2$ iff $t_1 \sqsubseteq t_2$ and $t_1 \neq t_2$. We also refer to prefixes t of some $\theta \in \mathbb{R}(\Sigma, I)$ in a similar way. If $E \subseteq t$ is a set of events, then $t[E] = [V', \lessdot', \lambda']$ is a prefix of t with the set $V' = \{f \in t \mid f \leq e \text{ for some } e \in E\}$ of events, and $\lessdot'$ and λ' are obtained from restricting the corresponding entities in t to V'. The least upper bound of two traces t_1, t_2, whenever it exists, denoted $t_1 \sqcup t_2$ is the smallest trace s such that $t_1 \sqsubseteq s \wedge t_2 \sqsubseteq s$. Similarly, if it exists, the greatest lower bound of t_1 and t_2, denoted $t_1 \sqcap t_2$, is the largest trace s such that $s \sqsubseteq t_1 \wedge s \sqsubseteq t_2$.

2.2 Asynchronous Transition Systems

We refer to a deterministic asynchronous automaton as a pair $\mathfrak{A} = (\mathfrak{T}, \mathcal{F})$, where $\mathfrak{T}$ is a deterministic asynchronous transition system and $\mathcal{F}$ is an appropriate acceptance condition. We discuss these components separately.

Over an alphabet (Σ, I), an asynchronous transition system consists of a set $\mathcal{P}$ of *processes*, a mapping $\mathsf{dom} : \Sigma \to 2^{\mathcal{P}}$ assigning the *domain* of each letter such that $\bigcup_{a \in \Sigma} \mathsf{dom}(a) = \mathcal{P}$ and $a\ I\ b \Leftrightarrow \mathsf{dom}(a) \cap \mathsf{dom}(b) = \emptyset$. Naturally, for $\Sigma' \subseteq \Sigma$, we also refer to $\mathsf{dom}(\Sigma') := \bigcup_{a \in \Sigma'} \mathsf{dom}(a)$. Moreover for an event $e \in t$, we refer to $\mathsf{dom}(e)$ instead of referring to $\mathsf{dom}(\lambda(e))$. Similarly, for $E \subseteq t$.

Processes p have sets X_p of *local p-states*. Introducing a symbol $\$ \notin \bigcup_{p \in \mathcal{P}} X_p$, for a set $P \subseteq \mathcal{P}$, the set X_P of *P-states* is a defined as $X_P = \{(x_p)_{p \in \mathcal{P}} \mid x_i \in X_{p_i} \text{ if } p_i \in P, \text{ otherwise } x_i = \$\}$. We find it convenient to assume an order over $\mathcal{P}$ and view a P-state as a tuple. So we refer to a *state* as a tuple $\pi \in X_P$ for some $P \subseteq \mathcal{P}$. A state is a *global state* if $P = \mathcal{P}$. We always distinguish between a $\{p\}$-state π and a local p-state x; and for a state π, define the *p-state in* π as $\pi_{|p} \coloneqq x_p \in X_p \cup \{\$\}$, and similarly the *$P$-state* $\pi_{|P}$ *in* π. Also, $\mathsf{dom}(\pi) \coloneqq \{p \in \mathcal{P} \mid \pi_{|p} \neq \$\}$. Finally, we denote the set of all states $X_{2^{\mathcal{P}}} \coloneqq \bigcup_{P \subseteq \mathcal{P}} X_P$.

Over a fixed independence alphabet (Σ, I), a set $\mathcal{P}$ of processes, and a mapping dom, we now define a *deterministic asynchronous transition system* (an *ATS*) as a tuple $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$, where X_p are sets of local p-states; transition functions $\delta_a \colon X_{\mathsf{dom}(a)} \to X_{\mathsf{dom}(a)}$ define how processes jointly perform state transitions letters a; and $\pi_0 \in X_{\mathcal{P}}$ is the global initial state of $\mathfrak{T}$.

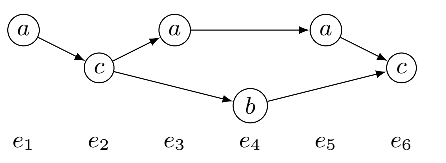
Given a trace $t = [V, \lessdot, \lambda] \in \mathbb{M}(\Sigma, I)$, or $\theta = [V, \lessdot, \lambda] \in \mathbb{R}(\Sigma, I)$, we define the corresponding *run* $\rho = [V', \lessdot', \lambda', \Lambda]$ of $\mathfrak{T}$ on the trace where $V' \coloneqq V \cup \{e_\perp\}$ contains a fictional, minimum event $e_\perp$. The relation $\lessdot'$ is identical to the edge relation $\lessdot$, except that $e_\perp$ is the unique minimum event.

During the run ρ of an ATS $\mathfrak{T}$ over a trace, each process p makes state transitions on events $e \in \mathsf{dom}^{-1}(p)$. Each such event may be called a *p-event* as well as a *P-event* where $P = \mathsf{dom}(e)$. All p-events in the run are totally ordered, and this order $<'_p$ can be defined with the help of the order $<$ of the trace. The *maximum p-event in* ρ according to the ordering $<'_p$ is denoted as $\max_p(\rho) \geq e_\perp$. If it exists, the p-predecessor f of an event e is denoted by $f \lessdot'_p e$. The labeling λ' is defined similarly except $\lambda'(e_\perp) \coloneqq \epsilon$; and $\Lambda \colon V' \to X_{2^{\mathcal{P}}}$ is defined inductively:

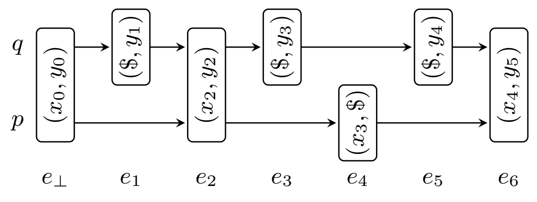
- Introduce a fictional, minimum event $e_\perp$ and define $V' \coloneqq V \cup \{e_\perp\}$;
- define $\lessdot'$ identically to the edge relation $\lessdot$, except that $e_\perp$ is the unique minimum event; similarly λ', except for convenience $\lambda'(e_\perp) \coloneqq \epsilon$;
- $\Lambda \colon V' \to X_{2^{\mathcal{P}}}$ is defined inductively as
  - $\Lambda(e_\perp) \coloneqq (\pi_0)$,
  - for any $e > e_\perp$, if
    - $a = \lambda(e)$, and
    - for the p-predecessors $e_p \lessdot_p e$, if $x_p = \Lambda(e_p)_{|p}$ are the most recent p-states just before e,

    then $\Lambda(e) \coloneqq \delta_a((y_p)_{p\in\mathcal{P}})$, where $y_p = x_p$ if $p \in \mathsf{dom}(e)$, $y_p = \$$ otherwise.

Fig. 1 shows the labeled events of a trace and the corresponding run; but λ' is omitted in ρ for readability. The processes are assumed to be lexicographically ordered, hence the representation of states as tuples. Note that, in Fig. 1b, the edges are shown as per the relations $\lessdot'_p, p \in \mathcal{P}$. Importantly, although $e_\perp <' e_2$ and $e_\perp \lessdot'_p e_2$, it is not the case that $e_\perp \lessdot' e_2$.



(a) Trace prefix $t = [V, \lessdot, \lambda]$.



(b) Run $\rho = [V \cup \{e_\perp\}, \lessdot', \lambda', \Lambda]$.

Fig. 1: For $\Sigma = \{a, b, c\}$, $a \ I \ b$, a finite trace (prefix) $t \in \mathbb{M}(\Sigma, I)$ and the run ρ of an ATS, with $\mathsf{dom}(a) = \{q\}, \mathsf{dom}(b) = \{p\}$, and $\mathsf{dom}(c) = \{p, q\}$.

Analogous to traces prefixes, we refer to run prefixes, and to prefixes $\rho[e], \rho[E]$ for $e \in \rho$ and $E \subseteq \rho$ respectively. For $e \in \rho$, we also refer to the label $\Lambda(e)$ as the *state of* $\mathfrak{T}$ *at* e. Similarly, if ρ is a finite run, then the *state of* $\mathfrak{T}$ *at* ρ is given by $\Lambda(\rho) = (x_p)_{p\in\mathcal{P}}$ where $x_p = \Lambda(\max_p(\rho))_{|p}$ is the p-state of $\mathfrak{T}$ at $\max_p(\rho)$; $x_p = \pi_{0|p}$ if $\max_p(\rho) = e_\perp$. Obviously, $\Lambda(\rho)$ is always a global state.

Finally, an *asynchronous automaton* over finite traces is a pair $\mathfrak{A} = (\mathfrak{T}, F)$, where $\mathfrak{T}$ is an ATS and $F \subseteq X_\mathcal{P}$ is a set of global states of $\mathfrak{T}$. A finite trace $t \in \mathbb{M}(\Sigma, I)$ is said to be *accepted* by $\mathfrak{A}$ if $\Lambda(t) \in F$. The set $L(\mathfrak{A}) \subseteq \mathbb{M}(\Sigma, I)$ denotes the set of all finite traces accepted by the asynchronous automaton $\mathfrak{A}$.

2.3 Deterministic Automata and ω-Regular Languages

A language $T \subseteq \mathbb{M}(\Sigma, I)$ is called *recognizable* or *regular* if there exists an asynchronous automaton $\mathfrak{A}$ such that $T = L(\mathfrak{A})$. The definition of regular languages of infinite traces, or *ω-regular trace languages* was first provided by Gastin-Petit [2] in terms of monoid morphisms. However, we use as definition, the characterization of the same family of languages in terms of deterministic (cellular) Muller automata [1, 5]. Although this definition was in terms of asynchronous cellular transition systems, they are equivalent to the ATS's that we have defined.

The notion of acceptance of an infinite trace $\theta \in \mathbb{R}(\Sigma, I)$ by an ATS $\mathfrak{T}$ is defined by referring to the sets of local states that occur infinitely often during the run ρ of $\mathfrak{T}$ over θ. For each process $p \in \mathcal{P}$, the set $\mathsf{Inf}_p(\rho)$ of local states visited infinitely often is constructed as follows:

$$\mathsf{Inf}_p(\rho) := \begin{cases} \left\{ x \in X_p \mid \exists^\infty e \in \rho : \Lambda(e)_{|p} = x \right\} & \text{if } p \in \mathsf{dom}(\mathsf{alphinf}(\theta)), \\ \left\{ x \in X_p \,\middle|\, \begin{array}{l} \exists e \in \rho : e = \max_p(\rho) \\ \text{and } \Lambda(e)_{|p} = x \end{array} \right\} & \text{otherwise.} \end{cases}$$

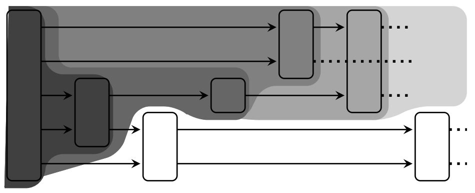
Let $\mathcal{F} = \{F_1, F_2, \dots\}$ be a table with $F_i = (F_i^p)_{p\in\mathcal{P}}$ being a tuple of subsets of local states of the processes. A *deterministic asynchronous Büchi automaton* (a *DABA*) is a pair $\mathfrak{A} = (\mathfrak{T}, \mathcal{F})$. A DABA is said to accept a trace $\theta \in \mathbb{R}(\Sigma, I)$ if, on the run ρ of $\mathfrak{A}$ on θ, there exists a tuple $F_i \in \mathcal{F}$ such that for each process p, $F_i^p \subseteq \mathsf{Inf}_p(\rho)$ [2, 1]. A *deterministic asynchronous Muller automaton* (a *DAMA*) is a pair $\mathfrak{A} = (\mathfrak{T}, \mathcal{F})$, and is said to accept a trace θ if there exists a tuple $F_i \in \mathcal{F}$ such that for each process p, $F_i^p = \mathsf{Inf}_p(\rho)$ [1].

Definition 1. *A language $\Theta \subseteq \mathbb{R}(\Sigma, I)$ is said to be an* ω-regular trace language *if it is recognized by a deterministic asynchronous Muller automaton.*

Definition 2 ([1]). *Let $T \subseteq \mathbb{M}(\Sigma, I)$ be a language of finite traces. The* infinitary limit *of T, denoted $\mathsf{lim}(T)$, is the language containing traces $\theta \in \mathbb{R}(\Sigma, I)$ such that there exists a sequence $(t_i)_{i\in\mathbb{N}}, t_i \in T$ satisfying $t_i \sqsubset t_{i+1}$ and $\bigsqcup_{i\in\mathbb{N}} t_i = \theta$.*

Fig. 2 illustrates the definition of $\mathsf{lim}(T)$ with the help of a run of an asynchronous automaton recognizing T. Shaded prefixes end in global accepting states. Fig. 2a illustrates an induced run if the trace $\theta \notin \mathsf{lim}(T)$, whereas Fig. 2b illustrates the contrary.

In the theory of ω-regular word languages, the family of deterministically Büchi recognizable languages can be characterized in terms of infinitary limit


(a) $\theta \notin \mathsf{lim}(T)$ since $\bigsqcup_{i \in \mathbb{N}} t_i \neq \theta$.

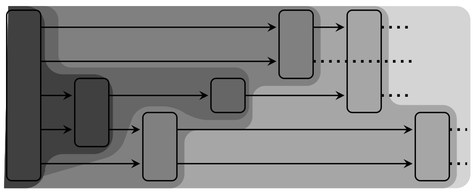

(b) $\theta \in \mathsf{lim}(T)$ if each event is eventually covered by one of the accepting runs over the sequence of strict prefixes.

Fig. 2: Shaded regions constitute a sequence of accepting runs of an automaton recognizing T.

languages $\mathsf{lim}(K)$ for $K \subseteq \Sigma^*$ regular. With the current definitions, it is still open whether there exists a DABA recognizing the language $\mathsf{lim}(T)$ for any given regular trace language $T \subseteq \mathbb{M}(\Sigma, I)$. It is however known that if we impose the classical Büchi acceptance condition on an ATS – namely, $(\mathfrak{T}, \mathcal{F})$ accepts θ if for its run ρ over θ there exists of $F_i \in \mathcal{F}$ with $F_i^p \cap \mathsf{Inf}_p(\rho) \neq \emptyset$ for each $p \in \mathcal{P}$ – then there exist regular languages $T \subseteq \mathbb{M}(\Sigma, I)$ such that $\mathsf{lim}(T)$ is not recognized by any $(\mathfrak{T}, \mathcal{F})$ (see e.g. [5]).

Muscholl also studies infinitary limits that are parameterized by a set of letters. This set governs which letters from the alphabet must occur infinitely often in the traces, and which letters may not.

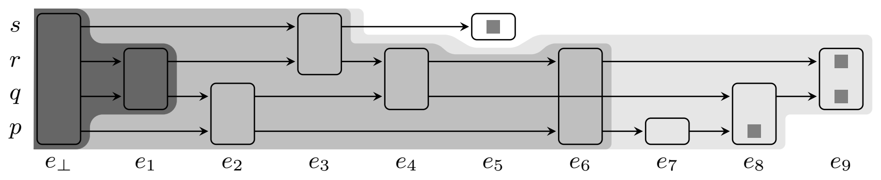
Definition 3 ([5]). *For $T \subseteq \mathbb{M}(\Sigma, I)$ and some $A \subseteq \Sigma$, the* A-infinitary limit *of T is defined as* $\mathsf{lim}_A(T) := \{\theta \in \mathsf{lim}(T) \mid D(\mathsf{alphinf}(\theta)) = D(A)\}$.

Definition 4 ([5]). *An ω-regular trace language is called a* deterministic trace language *if it can be expressed as a finite union of restricted infinitary limits of regular trace languages.*

Clearly, for $T \subseteq \mathbb{M}(\Sigma, I)$, the language $\mathsf{lim}(T)$ is a deterministic trace language since $\mathsf{lim}(T) = \bigcup_{A \subseteq \Sigma} \mathsf{lim}_A(T)$. However, every finite union of restricted infinitary limit languages may not be expressible in the form $\lim(T)$ for any T. Muscholl provides a natural extension to the definition of DABA wherein the acceptance condition $\mathcal{F}$ consists of pairs (F, A). A trace $\theta \in \mathbb{R}(\Sigma, I)$ is said to be accepted by this variant of DABA if there exists such a pair (F, A) where the run ρ of θ satisfies the Büchi condition w.r.t. F, as mentioned above, and $A \subseteq \mathsf{alphinf}(\theta)$. However, there exist deterministic trace languages that are not accepted by any DABA [5].

2.4 Secondaries and Frontiers

During a run ρ of an ATS, the processes can be thought of as "possessing and updating information" regarding other processes [3]. If ρ is finite and $p, q \in \mathcal{P}$, the *first-hand information* that p has about q at ρ, denoted by $\mathsf{latest}_{p \to q}(\rho)$, is the maximal q-event in the prefix $\rho[\max_p(\rho)]$. Trivially, $\mathsf{latest}_{p \to p}(\rho) = \max_p(\rho)$. Similarly, for $p, q, r \in \mathcal{P}$, the *second-hand information* that p has about r via q at ρ, denoted by $\mathsf{latest}_{p \to q \to r}(\rho)$, is the maximal r-event in the prefix $\rho[\mathsf{latest}_{p \to q}(\rho)]$. Trivially, $\mathsf{latest}_{p \to p \to q}(\rho) = \mathsf{latest}_{p \to q}(\rho)$.



Partial frontiers for ρ: $\{e_5\}$, $\{e_9\}$, $\{e_5, e_9\}$, $\{e_8, e_9\}$, and $\{e_5, e_8, e_9\}$.
At e_4, $\rho_\sqcap = \rho[e_1]$; and at e_9, $\rho_\sqcap = \rho[e_6]$. Note that $e_5 \notin \rho[e_9]$.

Fig. 3: Partial frontiers (see Sec. 2.4); and illustration of Lemma 7 (see Ex. 8).

The *primary information* of p at ρ is defined as the ordered set $\mathsf{Pri}_p(\rho) \coloneqq \{\mathsf{latest}_{p\to q}(\rho) \mid q \in \mathcal{P}\}$. The *secondary information* of p at ρ is given by the set $\mathsf{Sec}_p(\rho) \coloneqq \{\mathsf{latest}_{p\to q\to r}(\rho) \mid q, r \in \mathcal{P}\}$. It is easy to see that on the one hand $\mathsf{Pri}_p(\rho) \subseteq \mathsf{Sec}_p(\rho)$, and on the other hand the events of $\mathsf{Sec}_p(\rho)$ may be ordered as per the partial order $<$ of ρ. This gives us a view of the *secondary graph* of p at ρ, which we identify with secondary information itself. Clearly, $\max_p(\rho) = \mathsf{latest}_{p\to p\to p}(\rho)$ is the unique maximal event in $\mathsf{Sec}_p(\rho)$, and $|\mathsf{Sec}_p(\rho)| \leq |\mathcal{P}|^2$.

In this paper, we are mainly interested in secondary information of the form $\mathsf{Sec}_p(\rho[e])$ for $p \in \mathsf{dom}(e)$. Since, $\mathsf{Sec}_p(\rho[e]) = \mathsf{Sec}_q(\rho[e])$ for all $p, q \in \mathsf{dom}(e)$, for convenience we denote this information simply as $\mathsf{Sec}(e)$.

There exists a distributed algorithm, implemented by way of the so-called *gossip algorithm* [3], that enables processes to update their secondary graphs at the points of synchronization. When processes synchronize at an event e, the gossip algorithm takes the secondary sets $\mathsf{Sec}(f_p), f_p \lessdot_p e$ for each $p \in \mathsf{dom}(e)$, and outputs the updated secondary set $\mathsf{Sec}(e)$ reflecting the consistent, most recent information available collectively among processes in $\mathsf{dom}(e)$.

While referring to finite runs ρ over finite traces, or over finite prefixes of infinite traces, it is useful to refer to their maximum p-events as a set. Define *frontier of* ρ as $H_\rho \coloneqq \{e \in \rho \mid \exists p \in \mathcal{P}, e = \max_p(\rho)\}$. Any upward closed subset $H \subseteq H_\rho$ is called a *partial frontier*. E.g., the set $\{e_5, e_8\}$ in Fig. 3 is not a partial frontier of ρ since it is not an upward closed subset of the frontier $\{e_5, e_8, e_9\}$.

Finally, for event $e \in \rho$, define the *top of* e *in* ρ as $\top_\rho(e) \coloneqq \{f \in \rho \mid e \leq f \land \exists p \in \mathcal{P}\colon f = \max_p(\rho)\}$. Note that for any $e_1, \dots, e_n \in \rho$, $\bigcup_{i=1}^{n} \top_\rho(e_i)$ is a partial frontier of ρ.

Partial frontiers are significant because they consist of precisely the events that help describe the partial state of the automaton. If $\Lambda(\rho)$ is the global state of an automaton after the run ρ, and if H is a (partial) frontier of ρ, then we define $\Lambda(H) \coloneqq \Lambda(\rho)_{|\mathsf{dom}(H)}$. Roughly speaking, identifying a reasonable set of partial frontiers is necessary and sufficient for computing the global state that an automaton acquires at the end of a run.

3 A New Model of Asynchronous Automata

Any infinite run ρ of an ATS $\mathfrak{T}$ over a trace $\theta \in \mathbb{R}(\Sigma, I)$ yields a partition $\Psi = (P_1, \dots, P_n)$ of set $\mathcal{P}$ of processes such that each part $P_i \subseteq \mathcal{P}$ is minimal, and after a finite prefix $\rho_i \sqsubset \rho$, the processes $p \in P_i$ no longer interact directly or indirectly with another process $p' \in P_j, i \neq j$. A process p may infer that it belongs to part P_i by observing that it infinitely often updates its first-hand information w.r.t each $q \in P_i$. Our primary aim is to obtain a family of deterministic asynchronous transition system where such inferencing is possible.

3.1 Degrees of Synchronization

For an ATS $\mathfrak{T}$ and a run ρ of $\mathfrak{T}$ over any trace, we associate with each event $e \in \rho$ a measure of how much information is exchanged among the processes in $\mathsf{dom}(e)$. We use sets $P \subseteq \mathcal{P}$ of processes as the gauge for this measure.

Definition 5. *For a run ρ of an ATS and an event $e \in \rho$, the* secondary update *at e is the set $\mathcal{U}_e \coloneqq \{g \in \rho[e] \mid \exists p, q, r \in \mathcal{P}, \exists f_p \lessdot_p e : g = \mathsf{latest}_{p\to q\to r}(f_p) \neq \mathsf{latest}_{p\to q\to r}(e)\}$.*

Definition 6. *In a run ρ of an ATS, the* degree of synchronization *at an event $e \in \rho$ is defined as $\mathsf{ds}(e) \coloneqq \bigcup_{g\in\mathcal{U}_e} \mathsf{dom}(\top_{\rho[e]}(g))$. By default, $\mathsf{ds}(e_\perp) \coloneqq \mathcal{P}$.*

The set $\mathsf{ds}(e)$ implies that there must exist prefixes $\rho' \sqsubseteq \rho[e]$ with partial frontiers H, $\mathsf{dom}(H) = \mathsf{ds}(e)$, such that for some process $p \in \mathsf{dom}(e)$ with a predecessor $f_p \lessdot_p e$, $H \not\subseteq \rho[f_p]$. The following lemma illustrates this point, and demonstrates the importance of the set $\mathcal{U}_e$.

Lemma 7. *For $e \in \rho$, $e > e_\perp$, let $\rho_\sqcap \coloneqq \bigsqcap_{f_p \lessdot_p e} \rho[f_p]$ be the greatest lower bound of all its p-prefixes. For every prefix $\rho' \sqsubseteq \rho[e]$ with $\rho' \not\sqsubseteq \rho_\sqcap$, there exist $H \subseteq \rho'$ and $U \subseteq \mathcal{U}_e$ such that 1. H is a partial frontier in ρ' with $\mathsf{dom}(H) = \mathsf{ds}(e)$; and 2. $\bigcup_{g\in U} \top_{\rho'}(g) = H$.*

Before we prove this lemma, we consider an example that demonstrates the claims of its statement. For this, we refer back to Fig. 3.

Example 8. Referring to Fig. 3, at e_4, we have $e_2 \lessdot_q e_4$ and $e_3 \lessdot_r e_4$. Then, $\mathsf{ds}(e_4) = \mathcal{P}$ because $\mathcal{U}_{e_4} = \{e_\perp, e_1, e_2, e_3\}$. For instance $e_\perp = \mathsf{latest}_{q\to r\to s}(e_2) \neq \mathsf{latest}_{q\to r\to s}(e_4)$. Since $\rho_\sqcap = \rho[e_1]$, we have four possibilities of ρ', viz. $\rho'_1 = \rho[e_4]$, $\rho'_2 = \rho[e_2, e_3]$, $\rho'_3 = \rho[e_3]$, and $\rho'_4 = \rho[e_2]$. For ρ'_4, $H = \{e_\perp, e_1, e_2\}$ and we can choose $U = e_\perp \subseteq \mathcal{U}_{e_4}$. Symmetrically for ρ'_3. Also verify that, for ρ'_2, $H = U = \{e_2, e_3\}$; and for ρ'_1, $H = \{e_2, e_3, e_4\}$ and $U = \{e_\perp\}$.

Considering e_9 next, we have $e_8 \lessdot_q e_9$, $e_6 \lessdot_r e_9$, and $\mathcal{U}_{e_9} = \{e_2, e_4, e_6, e_8\}$. For instance, $e_2 = \mathsf{latest}_{r\to q\to p}(e_6) \neq \mathsf{latest}_{r\to q\to p}(e_9) = e_8$. Clearly, $\mathsf{ds}(e_9) = \{p, q, r\}$. And since $\rho_\sqcap = \rho[e_6]$, we have three possibilities of $\rho' \sqsubseteq \rho[e_9]$ s.t. $\rho' \not\sqsubseteq \rho_\sqcap$, the most interesting one being $\rho' = \rho[e_7]$. Now $H = \{e_4, e_6, e_7\}$ is the partial frontier of $\rho[e_7]$ with $\mathsf{dom}(H) = \mathsf{ds}(e_9)$, so we choose $U = \{e_2\} \subseteq \mathcal{U}_{e_9}$. ⊠

Proof (Lemma 7). We first prove a modified version of the lemma with a weaker claim, which corresponds to proving, "if the first condition holds then the second condition holds too."
Claim: For every prefix $\rho' \sqsubseteq \rho[e]$ such that $\rho' \not\sqsubseteq \rho_\sqcap$, *if* there exists a partial frontier H in ρ' with $\mathsf{dom}(H) = \mathsf{ds}(e)$ *then* there exists a subset $U \subseteq \mathcal{U}_e$ such that $\bigcup_{g\in U} \mathsf{dom}(\top_{\rho'}(g)) = H$.
Proof: Let ρ' be as in this claim, and let H be the partial frontier of ρ' with $\mathsf{dom}(H) = \mathsf{ds}(e)$.

If $|H| = 1$ then it must be the case that $H = \{e\}$ – because otherwise either $\rho' \sqsubseteq \rho_\sqcap$ which is prohibited, or for some pair $p, q \in \mathsf{dom}(e)$, $H \subseteq \rho[f_p]$ and

$H \cap \rho[f_q] = \emptyset$ which implies that $\mathsf{dom}(H) \subsetneq \mathsf{ds}(e)$ – and we can assign $U := \{f_p\}$ for any $p \in \mathsf{dom}(e)$.

Now, for the case where $|H| \geq 2$, the following condition must hold.

$$\forall f \in H,\ \exists p \in \mathsf{dom}(e),\ \exists q, r \in \mathsf{ds}(e)\colon \\ \mathsf{latest}_{p \to q \to r}(f_p) \leq f \ \wedge\ \mathsf{latest}_{p \to q \to r}(f_p) \neq \mathsf{latest}_{p \to q \to r}(e) \qquad (1)$$

Condition (1) is simple a restatement of the claim above, because if it is true then for each $f \in H$ we obtain $g_f = \mathit{latest}_{p \to q \to r}(f_p)$ such that $g_f \in \mathcal{U}_e$ and $g_f \leq f$. Now let $\bigcup_{f \in H} \top_{\rho'}(g_f) = H'$. Then it must be the case that $H' = H$, and this can be demonstrated by the following argument.

– $g_f \leq f \Rightarrow f \in \top_{\rho'}(g_f)$, therefore $H \subseteq H'$, implying that $\mathsf{dom}(H) \subseteq \mathsf{dom}(H')$;
– $\bigcup_{f \in H}\{g_f\} \subseteq \mathcal{U}_e$, and therefore by definition $\mathsf{ds}(e) \supseteq \bigcup_{f \in H} \mathsf{dom}(\top_{\rho[e]}(g_f)) \supseteq \bigcup_{f \in H} \mathsf{dom}(\top_{\rho'}(g_f))$;
– by assumption $\mathsf{ds}(e) = \mathsf{dom}(H)$ and $\bigcup_{f \in H} \mathsf{dom}(\top_{\rho'}(g_f)) = \mathsf{dom}(H')$, implying that $\mathsf{dom}(H) \supseteq \mathsf{dom}(H')$, and hence $\mathsf{dom}(H) = \mathsf{dom}(H')$;
– and finally, since H and H' are partial frontiers of the same trace ρ', and since $\mathsf{dom}(H) = \mathsf{dom}(H')$, it must necessarily hold that $H = H'$.

This will give us the desired set $U := \bigcup_{f \in H}\{g_f\}$. Towards a contradiction, assume that condition (1) is false. Then its negation must be true, which is:

$$\exists f \in H,\ \forall p \in \mathsf{dom}(e),\ \forall q, r \in \mathsf{ds}(e)\colon \\ \mathsf{latest}_{p \to q \to r}(f_p) \leq f \ \Rightarrow\ \mathsf{latest}_{p \to q \to r}(f_p) = \mathsf{latest}_{p \to q \to r}(e) \qquad (2)$$

In particular, $\forall h \in \rho[e]\colon h \leq f \Rightarrow h \notin \mathcal{U}_e$. Since $H \not\subseteq \rho_\sqcap$, there must exist an event $g \in H,\ g \notin \rho_\sqcap$. Moreover since H is a partial frontier and $f \in H$, there exists $r \in \mathcal{P}\colon f = \max_r(\rho')$. And because f and g belong to the same partial frontier, there must exist no r-event in the path from f to g (if such a path exists). This implies that $\forall q' \in \mathsf{dom}(g),\ \mathsf{latest}_{q' \to q' \to r}(g) \leq f$. Without loss of generality, we treat it as an equality.

Note the following. 1. $\mathsf{ds}(e) = \mathsf{dom}(H) \Rightarrow r \in \mathsf{ds}(e)$, and therefore there must exist a smallest r-event $f' \in \rho[e], f' > f$ and hence $f' \in \rho_\sqcap$, because otherwise for some $p \in \mathsf{dom}(e)$ the primary $\mathsf{latest}_{p \to p \to r}(f_p) = h \leq f$ and then p will update its primary information about r upon synchronizing at e resulting in $h \in \mathcal{U}_e$; 2. from condition (2) above, $\forall p \in \mathsf{dom}(e),\ \forall q' \in \mathsf{dom}(g)\colon \mathsf{latest}_{p \to q' \to r}(e) = f$, because otherwise for the same reason, $h = \mathsf{latest}_{p \to q' \to r}(f_p) < \mathsf{latest}_{p \to q' \to r}(e) = f$ implying $h \in \mathcal{U}_e$; and consequently 3. $q' \notin \mathsf{dom}(e)$, because otherwise q' will witness f' as a later r-event and update its primary from f to f' resulting in $f \in \mathcal{U}_e$. Recall that $g \notin \rho_\sqcap$, so it is possible to consider two events: one $f_p \lessdot_p e$ s.t. $g \notin \rho[f_p]$ and another $f_q \lessdot_q e$ s.t. $g \in \rho[f_q]$. Further, let $e_1 = \mathsf{latest}_{p \to p \to q'}(f_p)$ and $f_1 = \mathsf{latest}_{p \to p \to q}(f_p)$. The situation is shown in Fig. 4.

Since $g \in \rho[f_q]$, without loss of generality, let f_q be the event where q' and q synchronize. Otherwise there must exist a sequence of synchronizations among processes $q', q'', \ldots \notin \mathsf{dom}(e)$ before the last process in this sequence synchronizes with q. However, upon this last synchronization, because $\mathsf{latest}_{q \to q \to r}(f_q) \geq f'$ it must be the case that $\mathsf{latest}_{q' \to q' \to r}(f_q) = \mathsf{latest}_{q \to q' \to r}(f_q) \geq f'$. Finally at e, process p updates its information as $f = \mathsf{latest}_{p \to q' \to r}(f_p) \neq \mathsf{latest}_{p \to q' \to r}(e) \geq f'$ contradicting the assumption that $f \notin \mathcal{U}_e$.

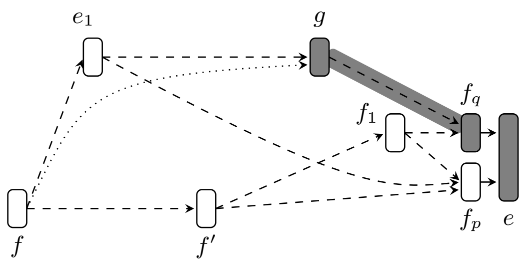


Fig. 4: From the proof of Lemma 7 – the shaded region lies outside of $\rho[f_p]$; $e_1, f_1 \in \mathsf{Pri}(f_p)$; no r-events along any (dotted or dashed) paths from f to g.

This concludes the proof of our modified claim. Now we prove that the first condition of the lemma holds as well, which will conclude our proof.
Claim: If $\rho' \sqsubseteq \rho[e]$ and $\rho' \not\sqsubseteq \rho_\sqcap$, then there exists a partial frontier H in ρ' with $\mathsf{dom}(H) = \mathsf{ds}(e)$.
Proof: Now let us assume that there does not exist any such frontier. Then for each partial frontier $H \subseteq H_{\rho'}$ with $\mathsf{ds}(e) \subseteq \mathsf{dom}(H)$ it holds that $\mathsf{ds}(e) \subsetneq \mathsf{dom}(H)$. Consider one such H.

For some $q \notin \mathsf{ds}(e)$, there must exist a q-event $f = \max_q(\rho') \in H$ such that for some $r \in \mathsf{ds}(e)$ there is a r-event $h = \max_r(\rho') \in H$ with $h \leq f$; otherwise we will find the frontier we are looking for. Without loss of generality, let $h = f$.

Since $\rho' \not\sqsubseteq \rho_\sqcap$, there exists at least one event $g \in \rho'$, $g \notin \rho_\sqcap$. Therefore, there exists $p \in \mathsf{dom}(e)$ s.t. $g \notin \rho[f_p]$, and it follows from the definitions $\mathsf{dom}(g) \subseteq \mathsf{ds}(e)$ and thus $g \in H$. Since $f = \max_r(\rho')$ there cannot be another r-event in any path from f to g (if any such path exists). Therefore for each $q' \in \mathsf{dom}(g)$, $\mathsf{latest}_{q' \to q' \to r}(g) \leq f$. Once more, without loss of generality, we assume $\mathsf{latest}_{q' \to q' \to r}(g) = f$. Also, since $r \in \mathsf{ds}(e)$ and $q \notin \mathsf{ds}(e)$ there must exist a minimum r-event $f' \in \rho[e]$, $f' > f$ and hence $f' \in \rho_\sqcap$. Moreover, $\forall p \in \mathsf{dom}(e)$, $\forall q' \in \mathsf{dom}(g)$: $\mathsf{latest}_{p \to q' \to q}(f_p) = f$ implying $q' \notin \mathsf{dom}(e)$.

So once more we arrive at the situation depicted in Fig. 4, and we proceed in a similar manner to arrive at the inference that $f = \mathsf{latest}_{p \to q' \to r}(f_p) \neq \mathsf{latest}_{p \to q' \to r}(e) \geq f'$, which in turn implies that $f \in \mathcal{U}_e$, and hence $q \in \mathsf{dom}(f) \cap \mathsf{ds}(e)$ which is a contradiction. ■

Remark 9. Lemma 7 would not hold if $\mathcal{U}_e$ were defined as the set of "primary updates", i.e. if $\mathcal{U}_e := \{g \in \rho[e] \mid \exists p, q \in \mathcal{P}, \exists f \lessdot_p e \colon g = \mathsf{latest}_{p \to p \to q}(f_p) \neq \mathsf{latest}_{p \to p \to q}(e)\}$. ⊠

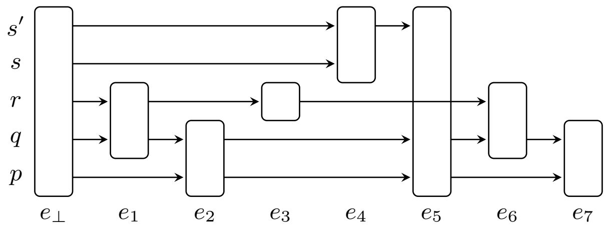


Fig. 5: Considering only primary updates is insufficient.

Proof. Consider the run $\rho \coloneqq \rho[e_7]$ as shown in the Fig. 5, and note that for event e_7, $e_5 \lessdot_p e_7$, and $e_6 \lessdot_q e_7$. The set of primary updates $\mathcal{U}_{e_7} = \{e_1, e_5, e_6\}$ since $e_1 = \mathsf{latest}_{p\to p\to r}(e_5) \neq \mathsf{latest}_{p\to p\to r}(e_7) = e_6$, $e_5 = \mathsf{latest}_{p\to p\to q}(e_5) \neq \mathsf{latest}_{p\to p\to q}(e_7) = e_7$, and $e_6 = \mathsf{latest}_{q\to q\to q}(e_6) \neq \mathsf{latest}_{q\to q\to q}(e_7) = e_7$. Further, as per Def. 6, $\mathsf{ds}(e_7) = \{p, q, r, s'\}$.

Now let $E = \{e_2, e_3, e_4\}$ and consider $\rho' = \rho[E]$. Clearly, $\rho' \sqsubseteq \rho$ and $e_3 \notin \rho[e_5] \Rightarrow \rho' \not\sqsubseteq \rho[e_5] \Rightarrow \rho' \not\sqsubseteq \rho_\sqcap$. However, there exists no partial frontier H of ρ' with $\mathsf{dom}(H) = \mathsf{ds}(e_7)$ because $\mathsf{dom}(e_4) = \{s, s'\} \not\subseteq \mathsf{ds}(e_7)$. Therefore, Lemma 7 breaks if we consider only the set of primary updates. ■

Corollary 10. *If $\mathcal{M}_e$ is the set of the minimal events of $\mathcal{U}_e$, then it suffices to always consider $U = \mathcal{M}_e$ in Lemma 7.*

Lemma 7 illustrates that degree of synchronization at event e corresponds precisely to some of the frontiers that had been missing from the view of some of the processes until they participated in e. Why we are interested in precisely these frontiers will be clear from Lemma 11 and Corollary 12. Presently, with respect to the partial frontiers H that are revealed by Lemma 7 at an event e, we refer to the set Y_e of states $\Lambda(H)$ as the *yield at* e. Clearly, for each $\pi_1, \pi_2 \in Y_e$: $\mathsf{dom}(\pi_1) = \mathsf{dom}(\pi_2) = \mathsf{ds}(e)$. We say that a yield Y_e is *bigger* than yield Y_f if $\mathsf{ds}(f) \subsetneq \mathsf{ds}(e)$.

The subset relation over $\mathcal{P}$ provides a natural order for comparing various degrees of synchronizations. The following lemma demonstrates that Def. 6 is well behaved over infinite runs in the sense that it helps each process p in identifying the maximal set of processes P regarding which p repeatedly updates in secondary, and hence primary, information.

Lemma 11. *For an infinite run ρ and $p \in \mathcal{P}$, if $p \in \mathsf{dom}(\mathsf{alphinf}(\rho))$ then there exists a unique maximal $P \subseteq \mathcal{P}$ such that $\exists^\infty e \in \rho : p \in \mathsf{dom}(e) \wedge \mathsf{ds}(e) = P$.*

Proof. For the given process p, we claim that P is the maximally interacting set of p and is defined as $P \coloneqq \{q \in \mathcal{P} \mid \forall e \in \rho, \exists e' \in \rho : e < e' \wedge \mathsf{latest}_{p\to p\to q}(e) \neq \mathsf{latest}_{p\to p\to q}(e')\}$. That is, $\mathcal{P} \setminus P$ is precisely the set of processes with whom p ceases to "interact" after a finite prefix.

Now let us assume that the statement of the lemma does not hold. This implies that there must exist at least two minimal sets $P_1, P_2 \subseteq P$ such that:

1. for infinitely many events $e \in \rho$: $p \in \mathsf{dom}(e) \wedge \mathsf{ds}(e) = P_1$,
2. for infinitely many events $e \in \rho$: $p \in \mathsf{dom}(e) \wedge \mathsf{ds}(e) = P_2$, and
3. there exists $e' \in \rho$ such that $\forall e \in \rho$: $e' < e \wedge p \in \mathsf{dom}(e) \Rightarrow \neg\big(P_1 \subsetneq \mathsf{ds}(e) \vee P_2 \subsetneq \mathsf{ds}(e)\big)$, and consequently
4. there exists a process $r \in P_2 \setminus P_1$ but no p-event $e > e'$ with $P_1 \cup \{r\} \subseteq \mathsf{ds}(e)$.

Consider a p-event $e_1 > e'$ with $\mathsf{ds}(e_1) = P_1$, and let $\mathcal{U}_{e_1}$ be the secondary update per Def. 5. Since $r \notin \mathsf{ds}(e_1)$, it follows from Def. 6 that for each $q \in P_1$ there exists $g_{q,r} \in \rho[e_1]$ such that

5. for each $p' \in \mathsf{dom}(e_1)$ and each $f \in \rho[e_1]$: $f \lessdot_{p'} e_1 \Rightarrow g_{q,r} = \mathsf{latest}_{p'\to q\to r}(f) = \mathsf{latest}_{p'\to q\to r}(e_1)$.

Inference 5 formally states the fact that all the synchronizing processes p' in $\mathsf{dom}(e_1)$ already agree on the second-hand information they have for process r via all processes $q \in \mathcal{P}$ in general, and via all processes $q \in P_1$ in particular. Moreover, since for all $q \in P_1$ the events $g_{q,r}$ are all r-events, they are all totally ordered in ρ. Let $q_1 \in P_1$ be such that $g_{q_1,r} \leq g_{q,r}$ for each $q \in P_1$. Therefore:

6. $P_1 \subseteq \mathsf{dom}(\top_{\rho[e_1]}(g_{q_1,r}))$.

Now, <u>if</u> p updates information about r infinitely often <u>and</u> p updates information about q_1 infinitely often <u>and</u> the set $\mathcal{P}$ of processes is finite, <u>then</u> r and q_1 must both update their information about each other infinitely often. From our choice of P, it follows that there exists an event e_{q_1} where q_1 will update its primary w.r.t. r, i.e. $g_{q_1,r} < \mathsf{latest}_{q_1 \to q_1 \to r}(e_{q_1})$. It further follows that there must exist a minimum p-event $e_2 > e_{q_1}$ where p will update its secondary for r via q_1, i.e. $g_{q_1,r} \neq \mathsf{latest}_{p \to q_1 \to r}(e_2)$. This implies that $g_{q_1,r} \in \mathcal{U}_{e_2}$. From inference 6 it follows that $P_1 \cup \{r\} \subseteq \mathsf{dom}(\top_{\rho[e_1]}(g_{q_1,r})) \subseteq \mathsf{dom}(\top_{\rho[e_2]}(g_{q_1,r})) \subseteq \mathsf{ds}(e_2)$, which contradicts inference 4. ■

We call the set P from Lemma 11 the *max-degree of p-synchronizations in* ρ, denoted by $\lceil \mathsf{ds}_p(\rho) \rceil$. For processes $p \notin \mathsf{dom}(\mathsf{alphinf}(\rho))$ that eventually halt, $\lceil \mathsf{ds}_p(\rho) \rceil := \{p\}$. The following corollary, that follows easily from Lemma 11, demonstrates the "symmetric" nature of max-degree of synchronizations.

Corollary 12. *For an infinite run ρ and $p, q \in \mathcal{P}$, either $\lceil \mathsf{ds}_p(\rho) \rceil = \lceil \mathsf{ds}_q(\rho) \rceil$ or $\lceil \mathsf{ds}_p(\rho) \rceil \cap \lceil \mathsf{ds}_q(\rho) \rceil = \emptyset$.*

In particular, for each part $P_i \in \Psi \colon q \in P_i \Leftrightarrow \lceil \mathsf{ds}_q(\rho) \rceil = P_i$. This concretizes our observation about a run ρ inducing a partition Ψ of the set of states, where each part is minimal and after a finite prefix, a process belonging to one part never interacts with a process belonging to another.

Definition 13. *A* synchronization-aware transition system *(an* SATS*) is a pair $(\mathfrak{T}, \mathcal{D})$ where $\mathfrak{T} = ((X_p)_{p \in \mathcal{P}}, (\delta_a)_{a \in \Sigma}, \pi_0)$ is an ATS and $\mathcal{D} = (\mathcal{D}_p)_{p \in \mathcal{P}}$ is a collection of mappings $\mathcal{D}_p \colon X_p \to 2^{\mathcal{P}}$ such that 1. $\mathcal{D}_p(\pi_{0|p}) = \mathcal{P}$, and 2. for every run ρ of $\mathfrak{T}$ and every event $e \in \rho$, if $\Lambda(e) = \pi$ and $p \in \mathsf{dom}(e)$ then $\mathsf{ds}(e) = P \Leftrightarrow \mathcal{D}_p(\pi_{|p}) = P$.*

The definition implies that, over any event, the processes of a synchronization-aware system always make transitions to local states that directly correspond to the degree of synchronization of the event in question. It is easy to see that properties 2 therein is in fact decidable, whence the definition is "syntactic".

3.2 Synchronization-aware Asynchronous Büchi Automata

A set $X \subseteq X_p$ of local p-states is called *homosynchronous* if for all local p-states $x, y \in X \colon \mathcal{D}_p(x) = \mathcal{D}_p(y)$. For an infinite run ρ of an SATS, we define the homosynchronous *maximal local infinity sets* $\lceil \mathsf{Inf}_p(\rho) \rceil$ as follows.

$$\lceil \mathsf{Inf}_p(\rho) \rceil = \begin{cases} \left\{ x \in X_p \;\middle|\; \begin{array}{l} \mathcal{D}_p(x) = \lceil \mathsf{ds}_p(\rho) \rceil \text{ and} \\ \exists^{\infty} e \in \rho : \Lambda(e)_{|p} = x \end{array} \right\} & \text{if } p \in \mathsf{dom}(\mathsf{alphinf}(\theta)), \\ \left\{ x \in X_p \;\middle|\; \begin{array}{l} \exists e \in \rho : e = \max_p(\rho) \\ \text{and } \Lambda(e)_{|p} = x \end{array} \right\} & \text{otherwise.} \end{cases}$$

Definition 14. *A* deterministic, synchronization-aware asynchronous Büchi automaton *(a* D-SABA*) is a tuple* $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$*, where* $(\mathfrak{T}, \mathcal{D})$ *is an SATS, and the acceptance table* $\mathcal{F} = \{(Q_1, F_1), \dots (Q_k, F_k)\}$ *is such that each* $Q_i \subseteq \mathcal{P}$ *and* $F_i = (F_i^p)_{p \in \mathcal{P}}$ *is a tuple of homosynchronous sets* F_i^p*. A D-SABA* $\mathfrak{A}$ accepts a trace $\theta \in \mathbb{R}(\Sigma, I)$ *if, for the run* ρ *of* $\mathfrak{A}$ *on* θ*, there exists a pair* $(Q_i, F_i) \in \mathcal{F}$ *s.t.* $\mathsf{dom}(\mathsf{alphinf}(\theta)) = Q_i$ *and for each process* $p \in \mathcal{P}$: $F_i^p \cap \lceil \mathsf{Inf}_p(\rho) \rceil \neq \emptyset$*.*

The above definition essentially requires that processes p ignore all of their infinitely occurring local p-states except those whose image under $\mathcal{D}_p$ matches the the maximal degree of p-synchronizations. A high level intuition behind this – and a reason why acceptance of traces via asynchronous Büchi automata is not as straightforward as acceptance of words via Büchi automata – is as follows.

Example 15. Consider an asynchronous automaton over finite traces, which has two processes and whose global acceptance state set $F = \{(x_1, y_1)\}$ is a singleton. Analogous to the word case, over the same ATS, one could define a Büchi automaton with acceptance condition $\mathcal{F} = \{(\{x_1\}, \{y_1\})\}$. Suppose a trace induces as shown in Fig. 6 below.

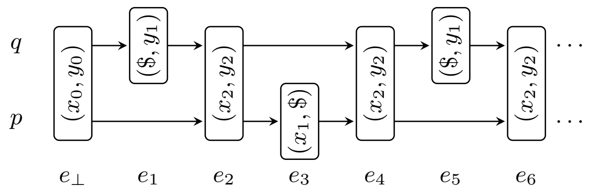


Fig. 6: The run segment corresponding to $e_1 e_2 e_3 e_4$ repeats ad infinitum. Local states x_1 and y_1 occur infinitely often, but the global state (x_1, y_1) never occurs.

Standard asynchronous automata suffer from the shortcoming that they cannot deduce that the global state (x_1, y_1) never occurs in the above run. This is in spite of the fact that the maximal degree of process synchronizations is $\mathcal{P}$. Ideally, at events $e_i \in \{e_2, e_4, e_6\}$ with $\mathsf{ds}(e_i) = \mathcal{P}$, each process should have been able to conclude in hindsight what the other process was up to; and only such repeated global inferences should govern whether or not to accept a trace. ⊠

Lemma 16. *If* $T \subseteq \mathbb{M}(\Sigma, I)$ *is a regular trace language, then there exists a D-SABA accepting* $\Theta = \mathsf{lim}(T)$*.*

We start with an asynchronous automaton $\mathfrak{A} = (\mathfrak{T}, F)$ recognizing a language $L(\mathfrak{A}) = T \subseteq \mathbb{M}(\Sigma, I)$ and show that it is possible to construct a D-SABA $\overline{\mathfrak{A}} = (\overline{\mathfrak{T}}, \mathcal{D}, \mathcal{F})$ recognizing $\mathsf{lim}(T)$. Intuitively, we construct $\overline{\mathfrak{T}}$ in such a manner that on one hand, it mimics the run of $\mathfrak{T}$ on every trace, and on the other hand its local states collect enough information about $\mathfrak{T}$'s run such that at each event e in its own run, $\overline{\mathfrak{T}}$ can compute the yield Y_e for the corresponding event e in $\mathfrak{T}$'s run. Now, fix the meanings of $\mathfrak{A}, \mathfrak{T}, \overline{\mathfrak{A}}$ and $\overline{\mathfrak{T}}$, and let $|\mathcal{P}| = N$.

Definition 17. *Let* $\pi_1 = (x_1, x_2, \dots x_N)$ *and* $\pi_2 = (y_1, y_2, \dots y_N)$ *be two elements of* $X_{2^\mathcal{P}}$*. The* projection of π_2 on π_1 *is defined as* $\pi_1 \lhd \pi_2 = (z_1, z_2, \dots z_N)$ *where the local* p_i*-states* z_i *are defined as* $z_i := \begin{cases} y_i & \text{if } y_i \neq \$ \\ x_i & \text{otherwise.} \end{cases}$

By extension, define $\Pi \lhd \Pi' := \{\pi \lhd \pi' \mid \pi \in \Pi,\ \pi' \in \Pi'\}$, and similarly $\pi \lhd \Pi$. We say that two states $\pi_1 = (x_1, \ldots, x_N)$ and $\pi_2 = (y_1, \ldots, y_N)$ are *compatible* if for each process p_i, $x_i = \$$ or $y_i = \$$ or $x_i = y_i$. If π_1 and π_2 are compatible, then $\pi_1 \lhd \pi_2 = \pi_2 \lhd \pi_1$; and sometimes this will have special utility.

Definition 18. *If π_1 and π_2 are two compatible states, then their* join *is defined as $\pi_1 \otimes \pi_2 := \pi_1 \lhd \pi_2$. The* join *of two incompatible states is not defined.*

Again, we extend the join operation to sets as $\Pi_1 \otimes \Pi_2 = \{\pi_1 \otimes \pi_2 \mid \pi_1 \in \Pi_1, \pi_2 \in \Pi_2, \pi_1 \text{ and } \pi_2 \text{ compatible}\}$. Note that each partial state π that $\mathfrak{T}$ acquires during a run can be extracted by referring to the labels of the events in partial frontiers, i.e. if H_1 and H_2 are two partial frontiers of a trace then $\Lambda(H_1 \cup H_2) = \Lambda(H_1) \otimes \Lambda(H_2)$. Additionally, according to Lemma 11 it suffices that during an infinite run, processes p repeatedly compute only the partial frontiers that correspond to maximal degrees of p-synchronizations.

Lemma 7 indicates that it is possible that at any event $e \in \rho$, the processes in $\mathsf{dom}(e)$ can compute the partial states for all partial frontiers H of prefixes $\rho' \sqsubseteq \rho[e],\ \rho' \not\sqsubseteq \rho_\sqcap,\ \mathsf{dom}(H) = \mathsf{ds}(e)$. For that purpose, we define the projection operation w.r.t partial frontiers and events in ρ.

Definition 19. *Let ρ be a finite run of an ATS $\mathfrak{T}$, let H be a partial frontier of ρ, and let $e \in \rho$ be an event such that $H \cap \top_\rho(e) \neq \emptyset$. If one exists, then let $e_p = \max_p(\rho[H])$ be the maximal p-event with $e < e_p$. Now, the* projection of H on e in ρ *is defined as a state $\rho[e \lhd H] := (x_p)_{p \in \mathcal{P}}$, where the local p-states*

$$x_p := \begin{cases} \Lambda(e_p)_{|p} & \textit{if there exists } e_p = \max_p(\rho[H]), e \lneq e_p, \\ \$ & \textit{otherwise.} \end{cases}$$

Note that if $H = \top_\rho(e)$ then $\Lambda(H) = \Lambda(e) \lhd \rho[e \lhd H]$. The following proposition generalizes this observation and tightly couples Lemma 7 with Def. 19.

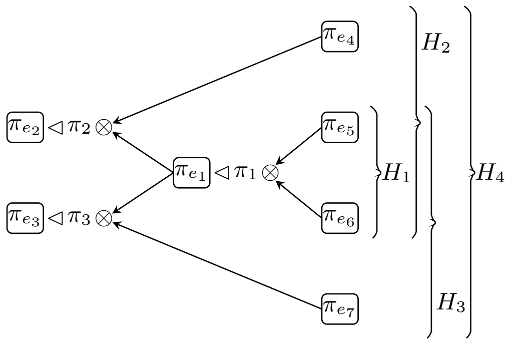
Proposition 20. *Let H be a partial frontier of ρ, and let $e_1, \ldots, e_n$ be pairwise concurrent events such that, for $1 \leq i \leq n, H \cap \top_\rho(e_i) \neq \emptyset$.*

1. *The states $\Lambda(e_i) \lhd \rho[e_i \lhd H],\ 1 \leq i \leq n$, are all pairwise compatible; and therefore,*
2. *if the frontier $H = \bigcup_{i=1}^{n} \top_\rho(e_i)$, then the partial $\mathsf{dom}(H)$-state in ρ can be computed as $\Lambda(H) = \bigotimes_{i=1}^{n}(\Lambda(e_i) \lhd \rho[e_i \lhd H])$.*

Prop. 20 follows immediately from the definitions, and it shows that events $e_i \in \mathsf{Sec}(H)$ can be used to compute $\Lambda(H)$ inductively. Fig. 7 illustrates this idea, where e.g. $\pi_1 = \pi_{e_5} \otimes \pi_{e_6}$, $\pi_2 = \pi_{e_4} \otimes (\pi_{e_1} \lhd \pi_1)$, etc.

Although Prop. 20 demonstrates how the cumulative secondary information $\bigcup_{p \in \mathcal{P}} \mathsf{Sec}_p(\rho)$ can help in computing $\Lambda(H)$, all of these secondary events may not be immediately available with any single process to perform this computation. We therefore need a way to "remember" certain partial states long enough so as to enable this computation at a later synchronization as per Lemma 7. Clearly, we require secondary information capable of storing partial states.

Definition 21. *Let ρ be a run of $\mathfrak{T}$. At any event $e \in \rho$, with reference to $\mathsf{Sec}(e)$, the* augmented secondary information *is defined as $\overline{\mathsf{Sec}}(e) := \{(f, \Pi, \pi) \mid f \in \mathsf{Sec}(e), \Pi \subseteq X_{2^\mathcal{P}}, \pi = \Lambda(f)\}$. Moreover, for each event $f \in \mathsf{Sec}(e)$ there exists exactly one augmented event $\overline{f} = (f, \Pi, \pi) \in \overline{\mathsf{Sec}}(e)$.*



π_{e_4}
H_2
$\pi_{e_2} \lhd \pi_2 \otimes$
π_{e_5}
$\pi_{e_1} \lhd \pi_1 \otimes$
H_1
H_4
$\pi_{e_3} \lhd \pi_3 \otimes$
π_{e_6}
π_{e_7}
H_3

Fig. 7: Using states $\pi_{e_i} \coloneqq \Lambda(e_i)$ to inductively reconstruct the partial states $\Lambda(H_i) = \pi_{e_i} \lhd \pi_i$ of partial frontiers H_i. Finally, $\Lambda(H_4) = (\pi_{e_2} \lhd \pi_2) \otimes (\pi_{e_3} \lhd \pi_3)$.

The motivation behind the above definition, and the invariant property that we wish to maintain in the augmented secondary set $\overline{\mathsf{Sec}}(e)$ at every event $e \in \rho$, is a certain modularity of the augmentations Π within each $\overline{f} \in \overline{\mathsf{Sec}}(e)$. This property is formalized as follows.

For each event $\overline{f} = (f, \Pi, \pi) \in \overline{\mathsf{Sec}}(e)$, there exists a partial state $\pi' \in \Pi$ iff there exists a prefix $\rho' \sqsubseteq \rho[e]$ and a partial frontier H in ρ' such that 1. $H = \top_{\rho'}(f)$; 2. $\pi' = \rho'[f \lhd H]$ or $\pi' = (\$, \dots \$)$; and 3. if $g \in \mathsf{Sec}(e)$ is an event such that $g > f$ then $H \cap \top_{\rho'}(g) = \emptyset$. $(*)$

Example 22. Consider a run prefix shown in Fig. 8 below.

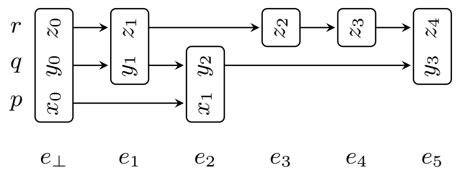


Fig. 8: A run ρ where events are labeled with local states. $\mathsf{Sec}(e_5) = \{e_1, e_2, e_5\}$.

As per property $(*)$, we have $\overline{e}_1 = (e_1, \Pi_1, \pi_1) \in \overline{\mathsf{Sec}}(e_5)$ with $\pi_1 = (\$, y_1, z_1)$ and the set Π_1 contains $\pi' = (\$, \$, z_2)$, because for $\rho' = \rho[e_3]$ and $H = \{e_1, e_3\}$ we have $\pi' = \rho'[e_1 \lhd H]$; and similarly for $\pi' = (\$, \$, z_3)$. For all the frontiers H, the states $\Lambda(H) = \Lambda(e_1) \lhd \rho[e_1 \lhd H]$ can now be found in the set $\pi_1 \lhd \Pi_1$. ⊠

We show later how this invariant is maintained in successive synchronizations. At present, assuming it holds, we show how we can exploit it to formalize the intuition presented in Fig. 7. The following procedure takes as an event e as input, and using the information $\overline{\mathsf{Sec}}(f_p)$, $f_p \lessdot_p e$ for each $p \in \mathsf{dom}(e)$, returns a data structure that forms the basis for computing the partial states for all partial frontiers H à la Lemma 7.

```
PartialStates($e$)
1. Create replicas $\iota$-$\overline{\mathsf{Sec}}(f_p)$ of each set $\overline{\mathsf{Sec}}(f_p)$; initialize $\iota$-$\overline{\mathsf{Sec}}(e) = \emptyset$.
2. To each set $\iota$-$\overline{\mathsf{Sec}}(f_p)$, append $\iota$-$\overline{e} = \big(e, \{(\$, \dots \$)\}, \Lambda(e)\big)$ as the greatest event.
```

```
3. Let $\{e\} \cup \bigcup_{p \in \mathsf{dom}(e)} \overline{\mathsf{Sec}}(f_p) = \{e_1, \ldots, e_n\}$ be the topologically ordered set of
   all secondary events, with $e_n = e$,
   for $i = n$ down to 1 do
   (a) Initialize event $\iota\text{-}\overline{e}_i \coloneqq (e_i, \Pi_i, \pi_i)$, where $\Pi_i = \emptyset$ and $\pi_i = \Lambda(e_i)$
   (b) Consider $Q_i = \{q_{i,1}, q_{i,2}, \ldots\} \subseteq \mathsf{dom}(e)$, such that for each $q_{i,j} \in Q_i$ there
       exists an $e_i$-copy $\iota\text{-}\overline{e}_{i,j} = (e_i, \Pi_{i,j}, \pi_i)$ in $\iota\text{-}\overline{\mathsf{Sec}}(f_{q_{i,j}})$:
       for $j = 1$ up to $|Q_i|$ do
         i. Locally in $\iota\text{-}\overline{\mathsf{Sec}}(f_{q_{i,j}})$, let $\{\iota\text{-}\overline{g}_{i,j,1}, \iota\text{-}\overline{g}_{i,j,2} \ldots\}$ be the set of all events
            $\iota\text{-}\overline{g}_{i,j,k} \gtrdot \iota\text{-}\overline{e}_{i,j}$, with $\iota\text{-}\overline{g}_{i,j,k} = (g_{i,j,k}, \Pi_{i,j,k}, \pi_{i,j,k})$. Let $\Pi = \emptyset$.
            begin iteration over $k$
              Update $\Pi \coloneqq \Pi \cup (\Pi \otimes \Pi_{i,j,k})$, assuming $\emptyset \otimes \Pi_{i,j,1} = \Pi_{i,j,1}$
            end iteration
            Update $\iota\text{-}\overline{e}_{i,j}$ by assigning $\Pi_{i,j} \coloneqq \Pi_{i,j} \cup (\Pi_{i,j} \lhd \Pi)$.
        ii. Update $\iota\text{-}\Pi_i \coloneqq \iota\text{-}\Pi_i \cup \Pi_{i,j} \cup (\iota\text{-}\Pi_i \otimes \Pi_{i,j})$, assuming $\emptyset \otimes \Pi_{i,1} = \Pi_{i,1}$
       done
   (c) Insert $\iota\text{-}\overline{e}_i \coloneqq (e_i, \iota\text{-}\Pi_i, \pi_i)$ in $\iota\text{-}\overline{\mathsf{Sec}}(e)$ as per the partial order.
   done.
4. return $\iota\text{-}\overline{\mathsf{Sec}}(e)$
```

The significance of $\iota\text{-}\overline{\mathsf{Sec}}(e)$ is that it is a temporary data structure whose augmentations $\iota\text{-}\Pi$ w.r.t. $\iota\text{-}\overline{f}$ contain the sum total of all the information of all the secondary events greater than f. Further assuming that each $\overline{\mathsf{Sec}}(f_p)$ satisfies $(*)$, the following property holds for $\iota\text{-}\overline{\mathsf{Sec}}(e)$.

For each event $\iota\text{-}\overline{f} = (f, \iota\text{-}\Pi, \pi) \in \iota\text{-}\overline{\mathsf{Sec}}(e)$, there exists a partial state $\pi' \in \iota\text{-}\Pi$ iff there exists a prefix $\rho' \sqsubseteq \rho[e]$ and a partial frontier H in ρ' such that 1. $H = \top_{\rho'}(f)$; and 2. $\pi' = \rho'[f \lhd H]$ or $\pi' = (\$, \ldots \$)$. (†)

Proposition 23. *For an event $e \in \rho$, if for each process $p \in \mathsf{dom}(e)$ the set $\overline{\mathsf{Sec}}(f_p), f_p \lessdot_p e$ satisfies property $(*)$ then $\iota\text{-}\overline{\mathsf{Sec}}(e)$ satisfies property (†).*

Proof. We argue by induction over the set $\{e_1, \ldots, e_n\}$ of events as mentioned in step 3 of PartialStates procedure. The induction proceeds in the same order as the loop, i.e. from n down to 1. As the base case, it is trivial to see that for index n, $\iota\text{-}\overline{e}_n = \iota\text{-}\overline{e} = (e, \{(\$, \ldots \$)\}, \Lambda(e))$, and there exists only one choice of prefix $\rho' \sqsubseteq \rho[e]$, which is $\rho[e]$ itself. Then we have $H = \top_{\rho'}(e) = \{e\}$ and $\rho[e \lhd H] = (\$, \ldots \$)$ (cf. Def. 19).

Now, assuming that all events up to index $\ell + 1$ satisfy (†), we consider $\iota\text{-}\overline{e}_\ell$ as obtained in step 3c. Let π' be a partial state in $\iota\text{-}\Pi_\ell$. If $\pi' \in \Pi_{\ell,j}$ already for some $\overline{e}_{\ell,j} \in \overline{\mathsf{Sec}}(f_{q_{\ell,j}})$, $q_{\ell,j} \in Q_\ell$ then we have nothing to show since condition $(*)$ already holds, implying that condition (†) holds. Otherwise, because the successor events $g_{\ell,j,k}$ chosen in Step 3(b)i are all pairwise concurrent, there must exist some prefixes $\rho_1, \ldots, \rho_m \sqsubseteq \rho[e]$ and partial frontiers $H_1, \ldots, H_m$ therein such that $\pi' = \rho_1[e_\ell \lhd H_1] \otimes \ldots \otimes \rho_m[e_\ell \lhd H_m]$. Firstly, we can set $\rho' \coloneqq \bigsqcup_{k=1}^m \rho_k \sqsubseteq \rho[e]$. Secondly, if $Q = \mathsf{dom}(\pi')$ then the set $H' \subseteq \bigcup_{k=1}^m H_k$ consisting of the maximal q-events from H_k's, $q \in Q$, is a partial frontier in ρ'. Hence it immediately follows that $\pi' = \rho'[e_\ell \lhd H']$.

For the reverse direction, consider a prefix $\rho' \sqsubseteq \rho[e]$ with a partial frontier H' such that $H' = \top_{\rho'}(e_\ell)$, and the set $G = \{g \in \rho' \mid g \neq e_\ell \Rightarrow g$ is an immediate successor of e_ℓ in $\{e\} \cup \bigcup_{p \in \mathsf{dom}(e)} \mathsf{Sec}(f_p)\}$. Then there are two possibilities.

If $G = \{e_\ell\}$ then either $H' = \{e_\ell\}$ in which case, by definition, $\rho'[e_\ell \lhd H'] = (\$, \dots \$)$ and is already a member of $\iota\text{-}\Pi_\ell$ by the initialization step 2. Otherwise, it must be the case that $\rho' \sqsubseteq \bigsqcup_{q_{\ell,j} \in Q_\ell} \rho[f_{q_{\ell,j}}]$. And then, H' may be expressed as $H' = H_{\ell,j_1} \cup H_{\ell,j_2} \cup \dots$ where that $H_{\ell,j_k} = H' \cap \rho[f_{q_{\ell,j_k}}]$ is a partial frontier in $\rho_{\ell,j,k} = \rho'[f_{\ell,j_k}]$. And therefore, $\rho'[e_\ell \lhd H'] = \rho_{\ell,j,1}[e_\ell \lhd H_{\ell,j,1}] \otimes \rho_{\ell,j,2}[e_\ell \lhd H_{\ell,j,2}] \dots$ is an element of $\iota\text{-}\Pi_\ell$ as a consequence of condition $(*)$ holding over $\overline{\mathsf{Sec}}(f_{\ell,j_k})$ and the join operation in Step 3(b)ii.

If $|G| > 1$, then denote the elements of G as $e_\ell = g_0, g_1, \dots, g_m$. For $k > 1$, let $H_k = \top_{\rho'}(g_k)$. Since these events g_k are are pairwise concurrent, by Prop. 20, the states $\Lambda(H_k) = \Lambda(g_k) \lhd \rho'[g_k \lhd H_k]$ are mutually compatible, and by induction hypotheses $\rho'[g_k \lhd H_k] \in \iota\text{-}\Pi_k$. From the compatibility of states, the order in which events g_k are joined in the local updates on $\iota\text{-}\overline{e}_{\ell,j}$ in step 3(b)i is immaterial. Finally, if $\bigcup_k H_k = H'$ then nothing more remains to be shown, since the local updates already ensure that $\Lambda(H') \in \iota\text{-}\Pi_\ell$. Otherwise, since $(*)$ holds on all sets $\overline{\mathsf{Sec}}(f_{q_{\ell,j}})$, using an argument similar to the case of $G = \{e_\ell\}$ we conclude that there must exist states $\pi_{\ell,j_1} \in \overline{\mathsf{Sec}}(f_{q_{\ell,j_1}})$ and so on, such that at the end of step 3(b)i for event e_ℓ, we have $\rho'[e_\ell \lhd H'] = (\pi_{\ell,j_1} \lhd \pi_M) \otimes (\pi_{\ell,j_2} \lhd \pi_M) \dots$, thereby implying that (†) holds for index ℓ as well. ■

Remark 24. Property (†) and Corr. 10 imply that corresponding to the set $\mathcal{M}_e \subseteq \mathcal{U}_e$ of the minimal updated secondary events at e, if we take the set $\overline{\mathcal{M}}_e = \{\iota\text{-}\overline{e}_1, \dots, \iota\text{-}\overline{e}_\ell\} \subseteq \iota\text{-}\overline{\mathsf{Sec}}(e)$ of the minimal iota-events, then processes $p \in \mathsf{dom}(e)$ can compute the yield $Y_e := \{\pi \in \bigotimes_{i=1}^{\ell} (\pi_i \lhd \iota\text{-}\Pi_i) \mid \mathsf{dom}(\pi) = \mathsf{ds}(e)\}$ which contains the partial states for all the partial frontiers H as in Lemma 7. ⊠

Finally, during the update of secondary information at e, some events from $\mathcal{U}_e$ may be deleted by the gossip algorithm. The processes must appropriately update the augmentations to ensure that the property $(*)$ is maintained. Referring to the partially ordered sets $\overline{\mathsf{Sec}}(f_p)$, $f_p \lessdot_p e$ for each $p \in \mathsf{dom}(e)$, and the set $\partial_e \subseteq \mathcal{U}_e$ of events to be deleted, we describe this update using the following procedure.

```
SecondaryUpdate($e$)
1. Initialize $\overline{\mathsf{Sec}}(e) = \emptyset$; append $\overline{e} = (e, \{(\$, \dots \$)\}, \Lambda(e))$ as its greatest event.
2. Let $\{e\} \cup \bigcup_{p \in \mathsf{dom}(e)} \mathsf{Sec}(f_p) = \{e_1, \dots, e_n\}$ be the topologically ordered set of
   all secondary events, with $e_n = e$,
   for $i = n$ down to 1 do
   (a) Initialize event $\overline{e}_i := (e_i, \Pi_i, \pi_i)$, where $\Pi_i = \emptyset$ and $\pi_i = \Lambda(e_i)$
   (b) Consider $Q_i = \{q_{i,1}, q_{i,2}, \dots\} \subseteq \mathsf{dom}(e)$, such that for each $q_{i,j} \in Q_i$ there
       exists an $e_i$-copy $\overline{e}_{i,j} = (e_i, \Pi_{i,j}, \pi_i)$ in $\overline{\mathsf{Sec}}(f_{q_{i,j}})$:
       for $j = 1$ up to $|Q_i|$ do
        i. Locally in $\iota\text{-}\overline{\mathsf{Sec}}(f_{q_{i,j}})$, let $\{\iota\text{-}\overline{g}_{i,j,1}, \iota\text{-}\overline{g}_{i,j,2} \dots\}$ be the set of all events
           $\iota\text{-}\overline{g}_{i,j,k} \gtrdot \iota\text{-}\overline{e}_{i,j}$ with $\iota\text{-}\overline{g}_{i,j,k} \in \partial_e$, and let $\iota\text{-}\overline{g}_{i,j,k} = (g_{i,j,k}, \Pi_{i,j,k}, \pi_{i,j,k})$.
           Initialize a temporary variable $\Pi = \emptyset$.
           begin iteration over $k$
               Update $\Pi := \Pi \cup (\Pi \otimes \Pi_{i,j,k})$, assuming $\emptyset \otimes \Pi_{i,j,1} = \Pi_{i,j,1}$
           end iteration
           Update $\iota\text{-}\overline{e}_{i,j}$ by assigning $\Pi_{i,j} := \Pi_{i,j} \cup (\Pi_{i,j} \lhd \Pi)$.
       ii. Update $\Pi_i := \Pi_i \cup \Pi_{i,j} \cup (\Pi_i \otimes \Pi_{i,j})$, assuming $\emptyset \otimes \Pi_{i,1} = \emptyset$
       done
   (c) If $e_i \notin \partial_e$ then add $\overline{e}_i := (e_i, \mathit{\Pi}_i, \pi_i)$ in $\overline{\mathsf{Sec}}(e)$ as per the partial order.
   done.
3. return $\overline{\mathsf{Sec}}(e)$
```

Proposition 25. *For an event $e \in \rho$, if for each process $p \in \mathsf{dom}(e)$ the set $\overline{\mathsf{Sec}}(f_p), f_p \lessdot_p e$, satisfies condition $(*)$ then, after the* SecUpdate *procedure, $\overline{\mathsf{Sec}}(e)$ satisfies condition $(*)$.*

Proof. As the basis, observe that $\overline{\mathsf{Sec}}(e_\perp) = (e_\perp, \{\pi_0\}, \pi_0)$ satisfies $(*)$. Hereafter, applying the induction hypothesis, the proof of this proposition is very similar to that of Prop. 23. The only difference between this procedure and PartialStates procedure is that, while in the latter the newly computed augmentations ι-$\mathit{\Pi}_i$ from step 3(b)i are always projected down to immediate predecessors, in SecondaryUpdate procedure the newly computed augmentations $\mathit{\Pi}_i$ from step 2(b)i are projected down to immediate predecessors only if the events $e_i \notin \mathsf{Sec}(e)$. Therefore, the modularity of the augmentations via the additional condition (3) of $(*)$ is preserved. ■

For processes $p \in \mathcal{P}$, let $\overline{\mathsf{SEC}}_p$ be the family of all augmented secondary sets whose unique maximal events are p-events. Starting from an ATS $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$, we construct an SATS $\overline{\mathfrak{T}} = \Big(\big((\overline{X}_p)_{p\in\mathcal{P}}, (\overline{\delta}_a)_{a\in\Sigma}, \overline{\pi}_0\big), \mathcal{D}\Big)$ where:

- $\overline{X}_p \subseteq \bigcup_{Q\subseteq\mathcal{P}} X_p \times \overline{\mathsf{SEC}}_p \times X_Q$, where sets Q are such that $p \in Q$;
- $\overline{\pi}_{0|p} = (x_0, \{(e_\perp, \{\pi_0\}, \pi_0)\}, \{\pi_0\})$ where $x_0 = \pi_{0|p}$ is the initial p-state of $\mathfrak{T}$;
- for letters $a \in \Sigma$, $\mathsf{dom}(a)$-states $\overline{\pi} = (x_p, \overline{\mathsf{Sec}}_p, Y_p)_{p\in\mathsf{dom}(a)} \in \overline{X}_{\mathsf{dom}(a)}$, and $q \in \mathsf{dom}(a)$, define the transition $\overline{\delta}_a(\overline{\pi})_{|q} := (y_q, \overline{\mathsf{Sec}}_e, Y_e)$, where
  - $y_q = \delta_a((x_p)_{p\in\mathsf{dom}(a)})_{|q}$ is obtained from $\mathfrak{T}$;
  - $\overline{\mathsf{Sec}}_e$ is obtained from SecondaryUpdate(e), assuming e is the current event in the run with $\lambda(e) = a$, and for $f_p \lessdot_p e, \overline{\mathsf{Sec}}(f_p) := \overline{\mathsf{Sec}}_p$; and
  - Y_e is the yield at e obtained from the PartialStates(e) procedure.
- For each local p-state $\overline{x} = (x, \overline{\mathsf{Sec}}, Y) \in \overline{X}_p$, if for any $\pi \in Y$, $\mathsf{dom}(\pi) = Q$, then assign $\mathcal{D}_p(\overline{x}) = Q$.

To summarize, at any event e, the yield Y_e consists of precisely the partial states $\pi \in X_{\mathsf{ds}(e)}$ of $\mathfrak{T}$ that had been missing from the view of one or the other process until they witnessed the event e. Over the infinite run ρ, the processes q belonging to the part $\lceil \mathsf{ds}_q(\rho) \rceil = P_i \in \mathit{\Psi}$ either will eventually halt at states $\pi_i \in X_{\{q\}}$ or will necessarily observe, at least in hindsight, all the partial states $\pi_i \in X_{P_i}$ that appear infinitely often in the run. Therefore, a global state $\pi \in X_{\mathcal{P}}$ of $\mathfrak{T}$ appears infinitely often in the run if and only if it can be inferred as the join $\bigotimes_i \pi_i$ of partial states corresponding to different parts in $\mathit{\Psi}$.

We now concretize this idea in order to describe the Büchi acceptance condition for the SATS constructed above. Fix a global accepting state $\pi_i \in F$ of $\mathfrak{A}$, a set $Q \subseteq \mathcal{P}$ of processes that make infinitely many transitions, and a partition $\mathit{\Psi}_j = \{P_{j,1}, \dots P_{j,n_j}\}$ of $\mathcal{P}$ compatible with Q, i.e. $P_{j,k} \cap Q \neq \emptyset \Rightarrow P_{j,k} \subseteq Q$ and $P_{j,k} \cap Q = \emptyset \Rightarrow |P_{j,k}| = 1$.

Firstly, for each $p \in Q$, the acceptance condition looks out for local p-states $\overline{x} = (x, \overline{\mathsf{Sec}}, Y) \in \overline{X}_p$ where the yield Y contains the partial state $\pi_{i|P_{j,k}}$, where $P_{j,k} \in \mathit{\Psi}_j$ is the part containing p. This is necessary and sufficient because, we know from our discussion above that, if one process in the part $P_{j,k}$ witnesses the partial state $\pi_{i|P_{j,k}}$ at some frontier, then sooner or later every other process in $P_{j,k}$ witnesses the same same partial frontier and hence the same state.

Secondly, let $S = \mathcal{P} \setminus Q \subsetneq \mathcal{P}$ be the set of processes that eventually stop. Clearly, for a fixed partition Ψ_j, S ranges over possible unions of singletons in it. For each such S, we consider a set $\mathcal{S} = (\mathcal{S}_p)_{p \in S}$, $\mathcal{S}_p \subseteq \mathcal{P}$. $\mathcal{S}$ ranges over the possible sets of degrees of synchronizations corresponding to the last transition of processes $p \in S$ during a run, and necessarily $p \in \mathcal{S}_p$. That is, for $p \in S$, the acceptance tuples looks out for local p-states $\overline{x} = (x, \overline{\mathsf{Sec}}, Y)$ where $x = \pi_{i|p}$ <u>and</u> $\mathcal{D}_p(\overline{x}) = \mathsf{dom}(Y) = \mathcal{S}_p$. We have the former requirement because process p stops at the state $\overline{x}$, and we have the latter requirement because each local acceptance set $\overline{F}^p$ can only contain states that have the same image under the mapping $\mathcal{D}_p$, which in this case is the same as $\mathsf{dom}(Y)$. Note that $\mathcal{S}$ ranges over the possibilities that correspond to the choice of Q, and the range of Q itself depends upon Ψ_j. That is, $\mathcal{P} \setminus Q$ can be non-empty only if Ψ_j contains singletons. And in particular, if $Q = \mathcal{P}$ then $\mathcal{S}$ becomes redundant.

Given $\pi_i \in F$, a non-empty set $Q \subseteq \mathcal{P}$, a partition Ψ_j compatible with Q, and $\mathcal{S}$ as above, we construct a local acceptance set for each process $p \in \mathcal{P}$ assuming $p \in P_{j,k}$: $\overline{F}^p_{i,Q,j,\mathcal{S}} = \begin{cases} \{(x, \overline{\mathsf{Sec}}, Y) \in \overline{X}_p \mid p \in P_{j,k} \wedge \pi_{i|P_{j,k}} \in Y\} & \text{if } p \in Q \\ \{(x, \overline{\mathsf{Sec}}, Y) \in \overline{X}_p \mid x = \pi_{i|p} \wedge \mathsf{dom}(Y) = \mathcal{S}_p\} & \text{if } p \notin Q \end{cases}$, which results in an acceptance pair $\big(Q, \overline{F}_{i,Q,j,\mathcal{S}}\big)$ with $\overline{F}_{i,Q,j,\mathcal{S}} = (\overline{F}^p_{i,Q,j,\mathcal{S}})_{p \in \mathcal{P}}$.

Proof (Lemma 16). Let $\mathfrak{A} = (\mathfrak{T}, F)$ be a deterministic asynchronous automaton recognizing a language $T \subseteq \mathbb{M}(\Sigma, I)$. We use the above construction to obtain from $\mathfrak{T}$ the corresponding SATS $(\overline{\mathfrak{T}}, \mathcal{D})$ with $\overline{\mathfrak{T}} = ((\overline{X}_p)_{p \in \mathcal{P}}, (\overline{\delta})_a, \overline{\pi}_0)$.

Defining the Büchi acceptance condition $\mathcal{F} \coloneqq \bigcup_i \bigcup_Q \bigcup_j \bigcup_{\mathcal{S}} \{(Q, \overline{F}_{i,Q,j,\mathcal{S}})\}$, we claim that $\overline{\mathfrak{A}} \coloneqq (\overline{\mathfrak{T}}, \mathcal{D}, \mathcal{F})$ is a deterministic, synchronization-aware Büchi automaton that recognizes the language $\Theta = \mathsf{lim}(T)$.

Clearly, a trace $\theta \in \mathbb{R}(\Sigma, I)$ is accepted by $\overline{\mathfrak{A}}$

iff there exists a pair $\big(Q, (\overline{F}^p)_{p \in \mathcal{P}}\big) \in \mathcal{F}$ such that $\mathsf{dom}(\mathsf{alphinf}(\theta)) = Q$ and in the run $\overline{\rho}$ of $\overline{\mathfrak{A}}$ over θ, for each $p \in \mathcal{P}$, $\overline{F}^p \cap \lceil \mathsf{Inf}_p(\overline{\rho}) \rceil \neq \emptyset$; and this holds

iff for the partition $\Psi = \{P_1, \ldots P_n\}$ of $\mathcal{P}$ induced by $\overline{\rho}$ and all $1 \leq k \leq n$, either

1. $P_k \subseteq Q$, and in this case processes $p \in P_k$ participate in infinitely many events in $\overline{\rho}$, $\lceil \mathsf{ds}_p(\overline{\rho}) \rceil = P_k$, and p witnesses some p-state $(x_p, \overline{\mathsf{Sec}}_p, Y_p) \in \overline{F}^p$ infinitely often; or
2. $P_k \not\subseteq Q$, and in this case process $q \in P_k$ participates in only finitely many events, and there exists some state $(x_q, \overline{\mathsf{Sec}}_q, Y_q) \in \overline{F}^q$ that is the last q-state;

and this holds

iff in the run ρ of $\mathfrak{A}$ over θ, there exists an infinite sequence of run prefixes $(\rho_i)_{i \in \mathbb{N}}$, with $\rho_i \sqsubset \rho_{i+1}$ and $\bigsqcup_{i \in \mathbb{N}} \rho_i = \rho$, where each ρ_i ends in a global state accepting state $\pi \in F$ for $\mathfrak{A}$ such that $\pi_{|Q} \in \bigotimes_{p \in Q} Y_p$ and $\pi_{|\mathcal{P} \setminus Q} = (x_q)_{q \notin Q}$; and this holds

iff there exists and infinite sequence of trace prefixes $(t_i)_{i \in \mathbb{N}}$ such that $t_i \in T, t_i \sqsubset t_{i+1}$ and $\bigsqcup_{i \in \mathbb{N}} t_i = \theta$; and this holds

iff $\theta \in \mathsf{lim}(T)$.

Note that the big join operation above is valid because for some $1 \leq k \leq n$ if $p_1, p_2 \in Q \cap P_k$ then $\pi_{|P_k} \in Y_{p_1} \cap Y_{p_2}$; and since $\pi_{|P_k}$ is compatible with itself, we have $\pi_{|P_k} \in Y_{p_1} \otimes Y_{p_2}$. On the other hand, if $p_1 \in P_{k_1}$ and $p_2 \in P_{k_2}, k_1 \neq k_2$, then Y_{p_1} is by construction compatible with Y_{p_2}. ■

Lemma 26. *If $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$ is a D-SABA with $|\mathcal{F}| = 1$, recognizing language $\Theta \subseteq \mathbb{R}(\Sigma, I)$, then there exists a set $A \subseteq \Sigma$ and a regular language $T \subseteq \mathbb{M}(\Sigma, I)$ such that $\Theta = \mathsf{lim}_A(T)$.*

Given $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$ recognizing language $\Theta \subseteq \mathbb{R}(\Sigma, I)$, let $\mathcal{F} = \{(Q, F)\}$. Then we need to present a (non-deterministic) asynchronous automaton $\mathfrak{B}$ recognizing $T \subseteq \mathbb{M}(\Sigma, I)$ s.t. for some $A \subseteq \Sigma$, $\Theta = \mathsf{lim}_A(T)$. Without loss of generality, we assume that (Q, F) is realizable. That is, for all $p, q \in Q$: $q \in \mathcal{D}_p(F^p) \Leftrightarrow \mathcal{D}_p(F^p) = \mathcal{D}_q(F^q)$. Further, note that although the acceptance tuple imposes that Q is precisely the set of processes that make infinitely many transitions, this does not imply that for some $\theta \in L(\mathfrak{A})$: $\mathsf{alphinf}(\theta) = \mathsf{dom}^{-1}(Q)$. The set of letters that may occur infinitely often in any $\theta \in L(\mathfrak{A})$ is precisely the set we fix as parameter A, and is given by $A \coloneqq \mathsf{dom}^{-1}(Q) \setminus \mathsf{dom}^{-1}(\mathcal{P} \setminus Q)$.

Next, we exploit non-determinism in asynchronous automata for languages of finite traces as well as a simple memory structure that processes p use to remember local q-states of other processes. The acceptance pair (Q, F) already indicates that if $p \in Q$ then, then p needs to look out only for infinitely occurring local q-states where $q \in \mathcal{D}_p(F^p)$. If $p \notin Q$ then p need not remember anything.

A local p-state of $\mathfrak{B}$ is of the form $\big(x, (X_{p\to q})_{q\in\mathcal{P}}, i\big)$ where $i \in \{0, 1\}$ is referred to as a *context*, x is a local p-state from $\mathfrak{A}$ and the memory-set $X_{p\to q}$ is a set of local q-states from $\mathfrak{A}$. Intuitively, the processes of $\mathfrak{B}$ begin in initial states where sets $X_{p\to q}$ are empty and the context $i = 0$. At first $\mathfrak{B}$ simply mimics $\mathfrak{A}$ – making transitions within context 0 and keeping sets $X_{p\to q}$ empty – until it non-deterministically decides that processes $p \in \mathcal{P} \setminus Q$ have all halted. At this point, processes $p \in Q$ make a "cross-over" transition to states with context $i = 1$. It is now that these processes start populating their memory-sets; and only those memory-sets that are relevant according to the Büchi acceptance condition (Q, F). At appropriate times, processes "reset" their memories and start accumulating again. The global final states of $\mathfrak{B}$ are defined around these points of reset.

Given a D-SABA $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$ with $\mathfrak{T} = \big((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0\big)$ and $\mathcal{F} = \{(Q, F)\}$, we construct a nondeterministic asynchronous automaton $\mathfrak{B} = (\overline{\mathfrak{T}}, \overline{F})$ by first constructing its underlying ATS $\overline{\mathfrak{T}} = \big((\overline{X}_p)_{p\in\mathcal{P}}, (\Delta_a)_{a\in\Sigma}, \overline{\pi}_0\big)$, and then describing $\overline{F}$ as follows:

– if $p \in Q$, then $\overline{X}_p \subseteq X_p \times (2^{X_p})_{p\in\mathcal{P}} \times \{0, 1\}$,
– otherwise, if $p \notin Q$, then $\overline{X}_p \subseteq X_p \times (2^{X_p})_{p\in\mathcal{P}} \times \{0\}$;
– $\overline{\pi}_{0|p} = (x_0, (\emptyset, \dots \emptyset), 0)$ where $x_0 = \pi_{0|p}$ is the initial p-state of $\mathfrak{T}$;
– for $a \in A$ the transition relation $\Delta_a \coloneqq \delta_{a,\times} \cup \bigcup_{i\in\{0,1\}} \delta_{a,i}$, where
  - for $\mathsf{dom}(a)$-states $\overline{\pi} = \big((x_p, (X_{p\to q})_{q\in\mathcal{P}}, i)\big)_{p\in\mathsf{dom}(a)}$, define the mapping $\delta_{a,i}(\overline{\pi})_{|p} \coloneqq \big(y_p, (Y_{p\to q})_{q\in\mathcal{P}}, i\big)$ such that
    - $y_p = \delta_a\big((x_q)_{q\in\mathsf{dom}(a)}\big)_{|p}$ is obtained from $\mathfrak{A}$; and
    - $Y_{p\to q}$ is computed[1] as follows:

[1] For the case when $i = 0$, this computation is redundant since all the memory-sets $Y_{p\to q}$ are empty to begin with.

1. First define $Z_{p\to q}$ as
   (a) if $q \in \mathcal{D}_p(F^p) \cap \mathsf{dom}(e)$, then $Z_{p\to q} \coloneqq \{y_q\} \cup \bigcup_{r\in\mathsf{dom}(e)} X_{r\to q}$;
   (b) else if $q \in \mathcal{D}_p(F^p) \setminus \mathsf{dom}(e)$, then $Z_{p\to q} \coloneqq \bigcup_{r\in\mathsf{dom}(e)} X_{r\to q}$.
2. Now, for each $q \in \mathcal{D}_p(F^p)$ if $Z_{p\to q} \cap F^q \neq \emptyset$ then define $Y_{p\to q} \coloneqq \emptyset$ else define $Y_{p\to q} \coloneqq Z_{p\to q}$.

- for $\mathsf{dom}(a)$-states $\overline{\pi} = \big((x_p, (X_{p\to q})_{q\in\mathcal{P}}, i)\big)_{p\in\mathsf{dom}(a)}$, define the mapping $\delta_{a,\times}(\overline{\pi})_{|p} \coloneqq \big(y_p, (Y_{p\to q})_{q\in\mathcal{P}}, i\big)$ such that
  * $y_p = \delta_a\big((x_q)_{q\in\mathsf{dom}(a)}\big)_{|p}$ is obtained from $\mathfrak{A}$; and
  * for $q \in \mathsf{dom}(a)$, $Y_{p\to q} \coloneqq \{y_q\}$.

– for $a \in \Sigma \setminus A$ the transition relation $\Delta_a \coloneqq \delta_{a,0}$ is defined similarly to the case $i = 0$ above.

– $\overline{F} \subseteq \overline{X}_{\mathcal{P}}$ is a set containing all global states $\big((x_p, (X_{p\to q})_{q\in\mathcal{P}}, i_p)\big)_{p\in\mathcal{P}}$ of $\mathfrak{T}$ where $\big[$for each $p \notin Q$ the $x_p \in F^p \wedge i_p = 0\big]$ <u>and</u> $\big[$for each $p \in Q, i_p = 1\big]$ <u>and</u> $\big[$for each set $F^p, p \in Q$, there exists $r \in F^p$ whose state $\big(x_r, (Y_{r\to q})_{q\in\mathcal{P}}, 1\big)$ is such that for each $q \in \mathcal{P}\colon Y_{r\to q} = \emptyset\big]$.

The acceptance set $\overline{F}$ of $\mathfrak{B}$ can be understood in the light of the transition function $\delta_{a,1}$ as it resets the memory-sets $Y_{p\to q}$ to empty, and the fact that the partition Ψ induced by every $\theta \in L(\mathfrak{A})$ can be inferred from (Q, F). The reset takes place if, during synchronization on an event, the collective memory-sets of the processes (along with the currently acquired local states) have non-empty intersections with respective sets in the Büchi acceptance condition. For each part $P \in \Psi,\ P \subseteq Q$, it is necessary and sufficient that at least one processes $p \in P$ repeatedly arrives at a local-state where the memory-sets are all empty. Then this local-state may constitute a global accepting state irrespective of local-state of other processes q in the same part P. On the other hand, processes $p \notin Q$ must come to a halt in one of their respective Büchi states, while making transitions solely within context $i = 0$.

Proof (Lemma 26). Let $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$ be a D-SABA recognizing a language $\Theta \subseteq \mathbb{R}(\Sigma, I)$. We claim that, if $\mathcal{F} = \{(Q, F)\}$ and $A = \bigcup_{p\in Q} \Sigma_p$ is computed as shown previously, then the NAA $\mathfrak{B} = ((\overline{X}_p)_{p\in\mathcal{P}}, (\Delta_a)_{a\in\Sigma}, \overline{\pi}_0, \overline{F})$ as constructed above recognizes $T \subseteq \mathbb{M}(\Sigma, I)$ such that $\Theta = \lim_A(T)$.

To show that $\Theta \subseteq \lim_A(T)$, let $\theta \in \Theta$ be an infinite trace accepted by $\mathfrak{A}$, and let Ψ be the corresponding maximal partition of processes from Q (i.e. we ignore the processes that halt). We show that θ can be broken down into a sequence of strictly increasing prefixes, each of which produce an accepting run in $\mathfrak{B}$.

First, we identify a prefix $t_\theta \sqsubset \theta$ such that $\forall q \in \mathcal{P} \setminus Q\colon \max_q(\theta) \in t_\theta$ <u>and</u> $\forall e \in \theta \setminus t_\theta, \exists P \in \Psi\colon \mathsf{ds}(e) \subseteq P$. The first condition says that all processes q that make only finitely many transition in the run over θ already process their maximal events in t_θ. The second condition says that while processing events e in any suffix of t_θ, the secondary information update of processes is restricted exclusively to the maximally interacting parts to which the processes belong.

Next, we consider a sequence $(t_i)_{i\in\mathbb{N}}$ s.t. $t_0 \coloneqq t_\theta$, $t_i \sqsubset t_{i+1}$, and $\bigsqcup_i t_i = \theta$. In order to describe this sequence, we make use of intermediate variables $t_{p\to q}$ where p, q belong to the same part $P \in \Psi$. Initially, we set $t_{p\to q} \coloneqq t_0$. Then, for each $i \geq 0$, t_{i+1} is a smallest trace satisfying:

1. there exists at least one part $P \in \Psi\colon P \cap \mathsf{dom}(t_{i+1} \setminus t_i) \neq \emptyset$;

2. for each part $P \in \Psi$ as in 1 above, there exists a maximal event e_P of t_{i+1} such that if $f_p \lessdot_p e_P, p \in \mathsf{dom}(e_P)$ are immediate predecessors of e_P then for each $q \in P, \exists p \in \mathsf{dom}(e_P), \exists e_q \in \{e_P\} \cup t[f_p] \setminus t_{p\to q} \colon \Lambda(e_q)_{|q} \in F^q$;
3. for each $P \in \Psi$ as in 1 above, for each $p \in \mathsf{dom}(e_P)$ and each $q \in P$, reassign $t_{p\to q} \coloneqq t[e_P]$.
4. As a special case, for t_1, the three conditions above must be fulfilled by <u>each</u> part $P \in \Psi$.

Given an input $t_i, i > 0$, the NAA $\mathfrak{B}$ first guesses $t_0 = t_\theta$, and starting in its initial state $\overline{\pi}_0$, it processes all the events of t_0 using the transition functions within context 0, i.e. transition functions of the form $\delta_{a,0}, a \in \Sigma$. So for all the processes $p \notin Q$ that halt, the final local p-states $\big(x_p, (\emptyset, \dots \emptyset), 0\big)$ in the run of $\mathfrak{B}$ are such that $x_p \in F^p$, because so far $\mathfrak{B}$ had been blindly mimicking $\mathfrak{A}$.

The remaining processes that are still active then move to local states with context 1 by virtue of the cross-over transition function $\delta_{a,\times}$, and start populating their memory-sets corresponding to their maximally interacting sets from Ψ.

The acceptance of t_1 by $\mathfrak{B}$ can be understood by looking at any one part $P \in \Psi$ individually and then repeating the analysis for every other part. Firstly, we argue why t_1 must exist. Since P is a maximally interacting set of θ, and since the processes $p \in P$ all visit their Büchi sets F^p infinitely often, it follows from Lemma 11 that there must exist at least one event e_P such that $\theta[e_P] \setminus t_\theta$ contains events favorable events e_p, $\Lambda(e_p)_{|p} \in F^p$ for each $p \in P$.

Next, since t_1 is a minimal such prefix, $e_P \in \theta \setminus t_\theta$ is a minimal such event. Since $\mathfrak{B}$ switches to the transition function $\delta_{a,1}$ after t_θ, as described in step 1a in the construction of $\delta_{a,i}$ above, individual processes $q \in \mathsf{dom}(e_P)$ maintain and update the r-states in their memory-sets $Y_{q\to r}$ much like the primary information update of the gossip algorithm. Only in this case, the retention of local states is not dependent upon the longevity of the primary event in the primary graph. Since there exist favorable events $e_r \in \theta[e_P] \setminus t_\theta$ for each $r \in P$, construction step 1a implies that $\Lambda(e_r)_{|r} \in Z_{q\to r}$. By construction step 1b, we have it that upon processing e_P, all processes $q \in \mathsf{dom}(e_P)$ arrive in local q-states with empty memory-sets. Therefore, we have that $\Lambda(t_1) = \bigotimes_{P\in\Psi} \Lambda(t_1)_{|P} \in \overline{F}$ since for each part $P \in \Psi$, and each $q \in \mathsf{dom}(e_P), \Lambda(t_1)_{|q}$ is a reset state.

Now, assuming t_i exists, the existence of t_{i+1} can again be established using the same arguments as for t_1. However, it must be pointed out that if for some part $P \in \Psi$, $P \cap \mathsf{dom}(t_{i+1} \setminus t_i)$, then it does not imply that for all processes $p \in P$ there exist states $x_p \in F^p$ in run over the factor $t_{i+1} \setminus t_i$. It only implies that some processes $q \in P$ witness these states x_p (although they may appear in $t_{i'}, i' \leq i$) for the first time. And clearly, $D(\mathsf{alphinf}(\theta)) = D(A)$ because θ must realize the acceptance pair (Q, F). Hence, we have established that θ can be decomposed in a sequence $(t_i)_{i\geq 1}$ of strictly increasing prefixes belonging to $L(\mathfrak{B})$.

Now for the reverse direction, consider an infinite trace $\theta \in \mathbb{R}(\Sigma, I)$ with $D(\mathsf{alphinf}(\theta)) = D(A)$, which can be broken into a sequence $(t_i)_{i\in\mathbb{N}}$ of strictly increasing prefixes, where each t_i induces an accepting run in $\mathfrak{B}$. By construction of the transition relation of $\mathfrak{B}$, it is evident that $\mathsf{alph}(t_{i+1} \setminus t_i) \subseteq A$ for all $i \in \mathbb{N}$. So we can claim that $Q = \mathsf{dom}(A)$ is precisely the set of processes that witness infinitely many transitions, and we can choose as the first element in the sequence a prefix t_n for large enough n such that the processes $\mathcal{P} \setminus Q$ have stopped. We can also infer that the final local states of processes $p \in \mathcal{P} \setminus Q$ are of the form $\big(x_p, (\emptyset, \dots \emptyset), 0\big)$ for $x_p \in F^p$.

Further, since $D(\mathsf{alphinf}(\theta)) = D(A)$ it must be the case that the run ρ of $\mathfrak{A}$ on θ induces precisely the partition Ψ as one deduces from the Büchi acceptance pair (Q, F). Therefore we obtain a sequence $(t'_j)_{j\in\mathbb{N}}$, with $t'_j := t_{n+j}$, which satisfies the four conditions mentioned above. And using a similar analysis as above, we conclude that each process $p \in Q$ visits its Büchi set F^p infinitely often during the run of θ over $\mathfrak{A}$. Hence, $\theta \in L(\mathfrak{A})$. ■

Proposition 27. *The family of D-SABA-recognizable languages is closed under finite unions.*

Proof (Sketch). Without loss of generality, we assume that the D-SABAs in question have all the same set $\mathcal{P}$ of processes and the same mapping dom.

Given two D-SABAs $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$ and $\mathfrak{B} = (\mathfrak{T}', \mathcal{D}', \mathcal{F}')$ we construct a synchronization-aware product automaton $\mathfrak{C}$, i.e. one where the sets Y_p of local p-states are constructed as $Y_p := \{(x, x') \in X_p \times X'_p \mid \mathcal{D}_p(x) = \mathcal{D}'_p(x')\}$. Over this product, for each acceptance pair $(Q, F) \in \mathcal{F}$ the new Büchi table $\mathcal{F}_\cup$ contains a pair $(Q, F_\cup)$ where $F^p_\cup := \{(x, x') \in Y_p \mid x \in F^p\}$. Similarly for each acceptance pair $(Q', F') \in \mathcal{F}'$ the new Büchi table $\mathcal{F}_\cup$ contains a pair $(Q', F'_\cup)$ where $F'^p_\cup := \{(x, x') \in Y_p \mid x' \in F'^p\}$. Now it is straightforward to show that $L(\mathfrak{C}) = L(\mathfrak{A}) \cup L(\mathfrak{B})$. ■

From Lemmas 16 and 26, and Prop. 27, we obtain the following result.

Theorem 28. *A language $\Theta \subseteq \mathbb{R}(\Sigma, I)$ is recognized by a D-SABA iff Θ is a deterministic trace language, i.e. Θ can be expressed as a finite union of languages of the form $\mathsf{lim}_A(T)$ for regular languages $T \subseteq \mathbb{M}(\Sigma, I)$ and sets $A \subseteq \Sigma$.*

Lastly, the following result has been established for the class of deterministic trace languages in [5]. We can now state the same result in terms of the family of D-SABA recognizable languages.

Proposition 29. *The family of D-SABA-recognizable languages is closed under finite intersections.*

3.3 Synchronization-aware Asynchronous Muller Automata

Over synchronization-aware asynchronous transition systems, we define the variant of deterministic Muller automata for languages of infinite traces. Then we show that this family of automata accept precisely the family of ω-regular trace languages by comparing them with DAMAs.

Definition 30. *A* deterministic synchronization-aware asynchronous Muller automaton *(a D-SAMA) is a tuple $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$, where $(\mathfrak{T}, \mathcal{D})$ is an SATS and the acceptance table $\mathcal{F} = \{F_1, \dots F_k\}$ is s.t. $F_i = (F^p_i)_{p\in\mathcal{P}}$ are tuples of homosynchronous sets F^p_i. A D-SAMA $\mathfrak{A}$* accepts *a trace $\theta \in \mathbb{R}(\Sigma, I)$ if, for the run ρ of $\mathfrak{A}$ on θ, there exists a tuple $F_i \in \mathcal{F}$ s.t. for each process $p \in \mathcal{P}$: $\lceil \mathsf{Inf}_p(\rho) \rceil = F^p_i$.*

Theorem 31. *Any language $\Theta \subseteq \mathbb{R}(\Sigma, I)$ of infinite traces is recognized by a D-SAMA if and only if Θ is recognized by a DAMA.*

Proof (Theorem 31(⇒)). Starting with a D-SAMA $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$ construct a DAMA $\mathfrak{A}' = (\mathfrak{T}, \mathcal{F}')$ over the same transition structure but a bigger acceptance condition defined as follows. For each $F_i = (F_i^p)_{p \in \mathcal{P}} \in \mathcal{F}$, construct $F'_{i,j} = (F_i^p \cup X_j^p)_{p \in \mathcal{P}}$ where the set $X_j^p \subseteq X_p$ is such that $\forall x \in X_j^p \colon \mathcal{D}_p(x) \subsetneq \mathcal{D}_p(F_i^p)$. The index j refers to the different ways of choosing the suitable subsets of X^p. This construction additionally ensures that in the DAMA, a local infinity set contains all of the original local p-states as well as those that correspond to strictly lesser degrees of synchronization. The required equivalence of automata is now straightforward. ■

For the reverse direction, Given a DAMA $\mathfrak{A} = (\mathfrak{T}, \mathcal{F})$, using the same augmented secondary information data structure as we introduced in Sec. 3.2, we first construct an SATS $(\overline{\mathfrak{T}}, \mathcal{D})$ corresponding to $\mathfrak{T}$. In order to proceed with the proof, we need to handle the case where some processes may halt. Unlike the case of D-SABA however, owing to the nature of Muller acceptance conditions, we cannot explicitly mark an acceptance set with the set of processes that halt. This situation must be handled implicitly within the acceptance condition.

We choose a symbol $\mathbb{I}$ to denote the index over a suitable range, which we will use to define the Muller condition $\overline{\mathcal{F}} = \bigcup_{\mathbb{I}} (\overline{F}_{\mathbb{I}}^p)_{p \in \mathcal{P}}$. Clearly, $\mathbb{I}$ must at least be of the form $(i, Q, j, \mathcal{S})$ where

- i ranges over the Muller sets $F_i \in \mathcal{F}$ of the input DAMA $\mathfrak{A}$;
- $Q \subseteq \mathcal{P}$ is the set of processes consistent with F_i that may make infinitely many transitions, that is $\forall p \in \mathcal{P} \colon \big(p \notin Q \Rightarrow |F_i^p| = 1\big) \wedge \big(|F_i^p| > 1 \Rightarrow p \in Q\big)$;
- j ranges over possible partitions Ψ_j which are consistent with the choice of Q, that is $\forall P_{j,k} \in \Psi_j \colon \big(P_{j,k} \cap Q \neq \emptyset \Rightarrow P_{j,k} \subseteq Q\big) \wedge \big(P_{j,k} \cap Q = \emptyset \Rightarrow |P_{j,k}| = 1\big)$;
- For the set $S = \mathcal{P} \setminus Q$ that, as a consequence of Q, must participate in only finitely many transitions, $\mathcal{S} = (\mathcal{S}_p)_{p \in S}, \mathcal{S}_p \subseteq \mathcal{P}$ is a collection of the possible degrees of synchronization of the final states of processes p.

Now if a process $p \in S$ then, from our choice of Q above, it must be the case that $|F_i^p| = 1$. Let $F_i^p = \{x\}$. Then indeed, the contribution $\overline{F}_{\mathbb{I}}^p$ of p to the Muller set $\overline{F}_{\mathbb{I}}$ must be a singleton, which can be chosen to contain any $\overline{x} = (x, \overline{\mathsf{Sec}}, Y) \in \overline{X}_p$ for some $\overline{\mathsf{Sec}}$, and Y such that $\mathcal{D}(\overline{x}) = \mathsf{dom}(Y) = \mathcal{S}_p$. But, for the fixed $x \in F_i^p$, there may be many such states $\overline{x} \in \overline{X}_p$ for varying values of $\overline{\mathsf{Sec}}$ and Y; and so on for all $p \in S$. Therefore, given $\mathcal{S}$,

- we must additionally have an index $\ell = (\ell_p)_{p \in S}$ that ranges over the possible collections of singletons w.r.t. i, Q, j, and $\mathcal{S}$.

So $\mathbb{I}$ must at least be of the form $(i, Q, j, \mathcal{S}, \ell)$, where $\mathcal{S} = (\mathcal{S}_p)_{p \notin Q}$ and $\ell = (\ell_p)_{p \notin Q}$. Now we move on to the processes $p \in Q$ that never halt. It is clear that if the run of processes p from a part $P_{j,k}$ is confined to precisely the states F_i^p, then during the run of the corresponding SATS $\overline{\mathfrak{T}}$, processes yields Y of processes $p \in P_{j,k}$ must be confined to a suitable subset of $\bigotimes_{p \in P_{j,k}} F_i^p$ which covers each local q-state in $F_i^q, q \in P_{j,k}$. Formally, given i, Q, and j

- we consider a collection $\mathcal{Y} = (\mathcal{Y}_p)_{p \in Q}$ where each set $\mathcal{Y}_p \subseteq 2^{P_{j,k}}$ of partial $P_{j,k}$-states of $\mathfrak{A}$ satisfies the conditions:
  1. For each $Y \in \mathcal{Y}_p,\ Y \subseteq \bigotimes_{q \in P_{j,k}} F_i^q$; and

2. For each $q \in P_{j,k}$ and each $x \in F_i^q$, there exists a set $Y \in \mathcal{Y}_p$ such that for some $\pi \in Y$, $\pi_{|q} = x$.

This means that $\mathbb{I}$ must at least be of the form $(i, Q, j, \mathcal{S}, \ell, \mathcal{Y})$ with $\mathcal{Y} = (\mathcal{Y}_p)_{p \in Q}$. Now for the processes $p \in Q$, the Muller set $\overline{F}^p_{\mathbb{I}}$ must be such that for each local p-state $(x, \overline{\mathsf{Sec}}, Y) \in \overline{F}^p_{\mathbb{I}}$ the set $Y \in \mathcal{Y}_p$ and for each $Y \in \mathcal{Y}_p$ there exists a p-state $(x, \overline{\mathsf{Sec}}, Y) \in \overline{F}^p_{\mathbb{I}}$. The idea here is that the set $\mathcal{Y}_p$ as a possible set of yields so that if <u>any</u> process $p \in P_{j,k}$ of $\overline{\mathfrak{T}}$ were to visit every local state in $F^p_{\mathbb{I}}$ as mentioned here, then this would imply that in the corresponding run of $\mathfrak{T}$ <u>all</u> processes $q \in P_{j,k}$ visit every state from the local state sets F_i^q.

Now, similar to the case of halting processes above, upon fixing a choice tuple $(\mathcal{Y}_p)_{p \in \mathcal{P}}$, there may be many possible ways in which choices of local Muller sets can be combined together. For example, for a fixed $\mathcal{Y}_p$ and the same yield Y, we may have $(x, \overline{\mathsf{Sec}}, Y) \in \overline{F}^p_{\mathbb{I}}$ and $(x', \overline{\mathsf{Sec}}', Y) \in \overline{F}^p_{\mathbb{I}'}$. So lastly, given i, Q, j and $\mathcal{Y}$,

- we must have an index $m = (m_p)_{p \in Q}$ that ranges over the possible ways of choosing local sets $\overline{F}^p_{\mathbb{I}}$ such that
  1. for each local p-state $(x, \overline{\mathsf{Sec}}, Y) \in \overline{F}^p_{\mathbb{I}}$ it holds that $Y \in \mathcal{Y}_p$;
  2. for each $Y \in \mathcal{Y}_p$ there exists a p-state $(x, \overline{\mathsf{Sec}}, Y) \in \overline{F}^p_{\mathbb{I}}$; and
  3. for each $\pi \in \bigcup_{Y \in \mathcal{Y}_p} Y$ if $\pi_{|p} = x$, then for some $\overline{\mathsf{Sec}}$ and $Y' \in \mathcal{Y}_p$ there exists a p-state $(x, \overline{\mathsf{Sec}}, Y') \in \overline{F}^p_{\mathbb{I}}$.

The three conditions automatically imply that for each $p \in Q$: there exists $x \in F_i^p$ if and only if there exists a state $(x, \overline{\mathsf{Sec}}, Y) \in \overline{F}^p_{\mathbb{I}}$ for some $\overline{\mathsf{Sec}}, Y$. Specifically, the last condition above ensures that if during an infinite run a process p witnesses all the yields from $\mathcal{Y}_p$ infinitely often, then the p-states themselves are consistent with these yields. Clearly, a local p-state y of $\mathfrak{A}$ appears in the yields infinitely iff $\mathfrak{A}$ infinitely often witnesses p-state y iff $\overline{\mathfrak{A}}$ infinitely often visits witnesses p-states of the form $(y, \overline{\mathsf{Sec}}', Y')$.

Therefore, we set the index $\mathbb{I}$ to be of the form $(i, Q, j, \mathcal{S}, \ell, \mathcal{Y}, m)$, and define $\overline{\mathcal{F}} = \bigcup_{\mathbb{I}} (\overline{F}^p_{\mathbb{I}})_{p \in \mathcal{P}}$. We believe that computation of exact size of the range of $\mathbb{I}$ is superfluous to the present discussion.

Before we proceed to the proof, we make an observation. It is possible that $\overline{F}_{\mathbb{I}} = \overline{F}_{\mathbb{I}'}$ even if the corresponding sets Q and Q' are unequal. For example, consider $\mathbb{I} = (i, Q, j, \mathcal{S}, \ell, \mathcal{Y}, m)$ and $\mathbb{I}' = (i, Q', j, \mathcal{S}', \ell', \mathcal{Y}', m')$. Let's say there exists a process $p \in Q$ but $p \notin Q'$. In particular, $\overline{F}^p_{\mathbb{I}} = \overline{F}^p_{\mathbb{I}'} = \{\overline{x}\}$. The partition Ψ_j in both the cases is the same, and this implies that $\mathbb{I}'$ assumes that p eventually halts at $\overline{x}$, and $\mathbb{I}$ assumes that p makes infinitely many transitions from $\overline{x}$ to itself and hence its maximally interacting set of processes is $\{p\}$. The automaton cannot distinguish between these assumptions underlying $\overline{F}_{\mathbb{I}}$ and $\overline{F}_{\mathbb{I}'}$. In fact, it treats these sets as one, and may likewise accept traces with different sets of halting processes by referring to it.

There is however, an important property of the new Muller sets $\overline{F}_{\mathbb{I}}$ that will be useful in handling the case of accepted traces where some processes may halt.

Proposition 32. *For any index $\mathbb{I} = (i, Q, j, \mathcal{S}, \ell, \mathcal{Y}, m)$ and any process $p \in \mathcal{P}$, if $\overline{F}^p_{\mathbb{I}} = \{(x, \overline{\mathsf{Sec}}, Y)\}$ then it must be the case that $F_i^p = \{x\}$.*

Proof. From our construction, $\overline{F}^p_{\mathbb{I}}$ may be assigned a singleton in two scenarios. First is the assumption that $p \notin Q$ and therefore p is considered to be one of the processes that may halt. Referring to the discussion of indices $i, Q, j, \mathcal{S}$ above, this can happen only if, p belongs to a singleton part in the partition Ψ_j, which itself is chosen in consistence with Q. And Q in turn could be chosen to exclude p only if $|F_i^p| = 1$. And by construction, $\overline{F}_{\mathbb{I}}^p = \{(x, \overline{\mathsf{Sec}}, Y)\}$ only if $F_i^p = \{x\}$.

Second scenario is the case when $p \in Q$. In this case, the conditions governing the choice of $\overline{F}_{\mathbb{I}}^p$ are described in the discussion of index m. If $\overline{F}_{\mathbb{I}}^p = \{(x, \overline{\mathsf{Sec}}, Y)\}$ then the three conditions together mandate that $\mathcal{Y}_p = \{Y\}$; that $P_{j,k}$-states in Y cover all the q-states occurring in $F_i^q,\ q \in P_{j,k}$; and condition 3 especially mandates that for any $\pi \in Y$ there exists a state $(x, \overline{\mathsf{Sec}}, Y) \in \overline{F}_{\mathbb{I}}^p$ with $\pi_{|p} = x$. And since $\overline{F}_{\mathbb{I}}^p$ is a singleton, we can claim that $\forall \pi \in Y \colon \pi_{|p} = x$. And since Y is the only set in $\mathcal{Y}_p$, it alone covers all sets $F_i^q,\ q \in P_{j,k}$. In particular, Y covers F_i^p. Hence $F_i^p = \{x\}$. ■

Proof (Theorem 31($\Leftarrow$)). We claim that, given a DAMA $\mathfrak{A} = (\mathfrak{T}, \mathcal{F})$, the D-SAMA $\overline{\mathfrak{A}} = (\overline{\mathfrak{T}}, \mathcal{D}, \overline{\mathcal{F}})$ as constructed here recognizes precisely the language $L(\mathfrak{A})$.

A trace $\theta \in \mathbb{R}(\Sigma, I)$ is accepted by $\overline{\mathfrak{A}}$

iff there exists a component $(\overline{F}^p)_{p\in\mathcal{P}} \in \overline{\mathcal{F}}$ such that in the run $\overline{\rho}$ of $\overline{\mathfrak{A}}$, for each $p \in \mathcal{P}$, $\overline{F}^p = \lceil \mathsf{Inf}_p(\overline{\rho}) \rceil$; and this holds

iff there exists a Muller component $(G^p)_{p\in\mathcal{P}} \in \mathcal{F}$ of $\mathfrak{A}$ such that
- for processes p that make infinitely many transitions in $\overline{\rho}$, sets $\overline{F}^p$ contain states whose yields collectively cover the sets $G^q,\ q \in \lceil \mathsf{ds}_p(\overline{\rho}) \rceil$. And this can only happen if in the run ρ of $\mathfrak{A}$ over θ, $\mathsf{Inf}_p(\rho) = G^p$.
- for processes p that make only finitely many transition in $\overline{\rho}$, it must be the case that $\overline{F}^p = \{(x, \overline{\mathsf{Sec}}, Y)\}$. And, as per Prop. 32, this can only happen if the Muller component $(\overline{F}^q)_{q\in\mathcal{P}}$ is constructed from such a component $(G^q)_{q\in\mathcal{P}}$ of $\mathfrak{A}$ where $G^p = \{x\}$.

Therefore, by construction of the SATS $\overline{\mathfrak{T}}$, we can claim that $\lceil \mathsf{Inf}_p(\overline{\rho}) \rceil = \overline{F}^p \Leftrightarrow \mathsf{Inf}_p(\rho) = G^p$. And this holds

iff θ is accepted by $\mathfrak{A}$.

This demonstrates that the family of deterministic, synchronization-aware Muller automata recognizes precisely the family of ω-regular trace languages. ■

We can now provide a constructive proof to show that the family of D-SAMAs enjoy the same closure properties under finite Boolean operations as the family of Muller automata over infinite words.

Proposition 33. *If $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$ and $\mathfrak{A}' = (\mathfrak{T}', \mathcal{D}', \mathcal{F}')$ are two D-SAMAs recognizing languages $\Theta, \Theta' \subseteq \mathbb{R}(\Sigma, I)$ respectively, then*

1. *the language $\mathbb{R}(\Sigma, I) \setminus \Theta$ is recognized by a D-SAMA; and*
2. *the language $\Theta \cup \Theta'$ is recognized by a D-SAMA.*

Proof (Sketch). For each possible partition Ψ_i of $\mathcal{P}$, let $Q_i^p = P_{i,j} \in \Psi_i,\ p \in P_{i,j}$. Now, for each process $p \in \mathcal{P}$, consider the set of local states $X_i^p \coloneqq \{x \in X_p \mid \mathcal{D}_p(x) = Q_i^p\}$. Define $\mathcal{F}_\neg \coloneqq \bigcup_i \Big(\big(\prod_{p\in\mathcal{P}} 2^{X_i^p} \big) \setminus \{F_j \in \mathcal{F} \mid \forall p \in \mathcal{P} \colon \mathcal{D}_p(F_j^p) = Q_i^p\} \Big)$. Now it is routine to show that the D-SAMA $\mathfrak{B} = (\mathfrak{T}, \mathcal{D}, \mathcal{F}_\neg)$ recognizes exactly the language $\mathbb{R}(\Sigma, I) \setminus \Theta$.

The other part of the proposition follows by referring to new acceptance component $\mathcal{F}_\cup$ constructed similarly in a synchronization-aware product automaton, i.e. one where the sets Y_p of local p-states are constructed as $Y_p \coloneqq \{(x, x') \in X_p \times X'_p \mid \mathcal{D}_p(x) = \mathcal{D}'_p(x')\}$. ■

4 Discussion

We introduced synchronization-aware asynchronous transition systems (SATS), that allow us to define for the first time asynchronous Büchi automata that recognize precisely the family of finite unions of A-infinitary limit languages, where $A \subseteq \Sigma$ governs the letters with infinite occurrences. We suggest that this set of languages, which strictly includes the class of infinitary limits of regular trace languages, should be referred to as the set of *deterministically Büchi recognizable trace languages*. This is because not only can their definition be viewed as a generalization of that for the word case but, more importantly, they are closed under finite unions and intersections – analogous to the deterministically Büchi recognizable languages of infinite words.

The procedure that we present to construct an SATS $\overline{\overline{\mathfrak{T}}}$ from an arbitrary ATS $\mathfrak{T}$ essentially refines the state space of $\mathfrak{T}$, and from that point of view it may also find utility in exploring properties of languages of finite traces. Moreover, we can modify the algorithm slightly and apply it to "I-diamond" finite state automata to directly obtain an SATS without the intermediate ATS.

We also consider it important to explore a definition of synchronization-aware asynchronous cellular automata, where, although a set of processes synchronize on a letter, exactly one of them changes its state. With the existing definitions, it is known that asynchronous cellular automata are equivalent to asynchronous automata for the case of languages of finite traces.

Some basic questions still remain open, e.g. whether the ω-regular trace languages are precisely those that can be obtained as finite Boolean combinations of deterministically Büchi recognizable trace languages. It is also open whether an ω-regular trace language is deterministically Büchi recognizable precisely when it is recognized by a deterministic Muller automaton whose acceptance table is closed under supersets. The corresponding results for the case of ω-regular word languages were established by McNaughton and Landweber respectively (cf. [6]). We believe that a well rounded theory of traces necessitates that these results extend to ω-regular traces languages as well, and we suspect that the definition of synchronization-aware transition systems is a step in that direction.

Analogous to the study of families of word automata and the corresponding families of ω-regular languages, our interest lies in contributing to the theory of asynchronous automata for ω-regular trace languages. Therefore, beyond addressing the classical theorems, our results also suggest definitions of classes of automata with weaker acceptance conditions, namely, Staiger-Wagner automata and automata recognizing reachability and safety acceptance conditions.

Acknowledgement I would like to thank Marcus Gelderie, Christof Löding, and Wolfgang Thomas for encouragement and numerous fruitful discussions. I also thank the anonymous referees of a conference to which a preliminary version of this report was submitted.

References

1. Volker Diekert and Anca Muscholl. Deterministic asynchronous automata for infinite traces. In P. Enjalbert, A. Finkel, and K. Wagner, editors, *STACS 93*, volume 665 of *Lecture Notes in Computer Science*, pages 617–628. Springer, 1993.

2. Paul Gastin and Antoine Petit. Asynchronous cellular automata for infinite traces. In W. Kuich, editor, *Automata, Languages and Programming*, volume 623 of *Lecture Notes in Computer Science*, pages 583–594. Springer, 1992.
3. Mukund Madhavan. Automata on distributed alphabets. In Deepak D'Souza and Preeti Shankar, editors, *Modern Applications of Automata Theory*, volume 2 of *IISc Research Monographs Series*, pages 257–288. World Scientific, May 2012.
4. Antoni Mazurkiewicz. Trace theory. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *LNCS*, pages 278–324. Springer-Verlag, 1987.
5. Anca Muscholl. *Über die Erkennbarkeit unendlicher Spuren.* PhD thesis, 1994.
6. Wolfgang Thomas. Automata on infinite objects. Technical Report AIB-88-17, RWTH Aachen University, 1988. Available online.
7. Wieslaw Zielonka. Notes on Finite Asynchronous Automata. *R.A.I.R.O. – Informatique théorique et applications*, 21:99–135, 1987.

Aachener Informatik-Berichte

This list contains all technical reports published during the past three years. A complete list of reports dating back to 1987 is available from:

http://aib.informatik.rwth-aachen.de/

To obtain copies please consult the above URL or send your request to:

Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

2010-01 * Fachgruppe Informatik: Jahresbericht 2010
2010-02 Daniel Neider, Christof Löding: Learning Visibly One-Counter Automata in Polynomial Time
2010-03 Holger Krahn: MontiCore: Agile Entwicklung von domänenspezifischen Sprachen im Software-Engineering
2010-04 René Wörzberger: Management dynamischer Geschäftsprozesse auf Basis statischer Prozessmanagementsysteme
2010-05 Daniel Retkowitz: Softwareunterstützung für adaptive eHome-Systeme
2010-06 Taolue Chen, Tingting Han, Joost-Pieter Katoen, Alexandru Mereacre: Computing maximum reachability probabilities in Markovian timed automata
2010-07 George B. Mertzios: A New Intersection Model for Multitolerance Graphs, Hierarchy, and Efficient Algorithms
2010-08 Carsten Otto, Marc Brockschmidt, Christian von Essen, Jürgen Giesl: Automated Termination Analysis of Java Bytecode by Term Rewriting
2010-09 George B. Mertzios, Shmuel Zaks: The Structure of the Intersection of Tolerance and Cocomparability Graphs
2010-10 Peter Schneider-Kamp, Jürgen Giesl, Thomas Ströder, Alexander Serebrenik, René Thiemann: Automated Termination Analysis for Logic Programs with Cut
2010-11 Martin Zimmermann: Parametric LTL Games
2010-12 Thomas Ströder, Peter Schneider-Kamp, Jürgen Giesl: Dependency Triples for Improving Termination Analysis of Logic Programs with Cut
2010-13 Ashraf Armoush: Design Patterns for Safety-Critical Embedded Systems
2010-14 Michael Codish, Carsten Fuhs, Jürgen Giesl, Peter Schneider-Kamp: Lazy Abstraction for Size-Change Termination
2010-15 Marc Brockschmidt, Carsten Otto, Christian von Essen, Jürgen Giesl: Termination Graphs for Java Bytecode
2010-16 Christian Berger: Automating Acceptance Tests for Sensor- and Actuator-based Systems on the Example of Autonomous Vehicles
2010-17 Hans Grönniger: Systemmodell-basierte Definition objektbasierter Modellierungssprachen mit semantischen Variationspunkten
2010-18 Ibrahim Armaç: Personalisierte eHomes: Mobilität, Privatsphäre und Sicherheit
2010-19 Felix Reidl: Experimental Evaluation of an Independent Set Algorithm
2010-20 Wladimir Fridman, Christof Löding, Martin Zimmermann: Degrees of Lookahead in Context-free Infinite Games

2011-01 *	Fachgruppe Informatik: Jahresbericht 2011
2011-02	Marc Brockschmidt, Carsten Otto, Jürgen Giesl: Modular Termination Proofs of Recursive Java Bytecode Programs by Term Rewriting
2011-03	Lars Noschinski, Fabian Emmes, Jürgen Giesl: A Dependency Pair Framework for Innermost Complexity Analysis of Term Rewrite Systems
2011-04	Christina Jansen, Jonathan Heinen, Joost-Pieter Katoen, Thomas Noll: A Local Greibach Normal Form for Hyperedge Replacement Grammars
2011-06	Johannes Lotz, Klaus Leppkes, and Uwe Naumann: dco/c++ - Derivative Code by Overloading in C++
2011-07	Shahar Maoz, Jan Oliver Ringert, Bernhard Rumpe: An Operational Semantics for Activity Diagrams using SMV
2011-08	Thomas Ströder, Fabian Emmes, Peter Schneider-Kamp, Jürgen Giesl, Carsten Fuhs: A Linear Operational Semantics for Termination and Complexity Analysis of ISO Prolog
2011-09	Markus Beckers, Johannes Lotz, Viktor Mosenkis, Uwe Naumann (Editors): Fifth SIAM Workshop on Combinatorial Scientific Computing
2011-10	Markus Beckers, Viktor Mosenkis, Michael Maier, Uwe Naumann: Adjoint Subgradient Calculation for McCormick Relaxations
2011-11	Nils Jansen, Erika Ábrahám, Jens Katelaan, Ralf Wimmer, Joost-Pieter Katoen, Bernd Becker: Hierarchical Counterexamples for Discrete-Time Markov Chains
2011-12	Ingo Felscher, Wolfgang Thomas: On Compositional Failure Detection in Structured Transition Systems
2011-13	Michael Förster, Uwe Naumann, Jean Utke: Toward Adjoint OpenMP
2011-14	Daniel Neider, Roman Rabinovich, Martin Zimmermann: Solving Muller Games via Safety Games
2011-16	Niloofar Safiran, Uwe Naumann: Toward Adjoint OpenFOAM
2011-17	Carsten Fuhs: SAT Encodings: From Constraint-Based Termination Analysis to Circuit Synthesis
2011-18	Kamal Barakat: Introducing Timers to pi-Calculus
2011-19	Marc Brockschmidt, Thomas Ströder, Carsten Otto, Jürgen Giesl: Automated Detection of Non-Termination and NullPointerExceptions for Java Bytecode
2011-24	Callum Corbett, Uwe Naumann, Alexander Mitsos: Demonstration of a Branch-and-Bound Algorithm for Global Optimization using McCormick Relaxations
2011-25	Callum Corbett, Michael Maier, Markus Beckers, Uwe Naumann, Amin Ghobeity, Alexander Mitsos: Compiler-Generated Subgradient Code for McCormick Relaxations
2011-26	Hongfei Fu: The Complexity of Deciding a Behavioural Pseudometric on Probabilistic Automata
2012-01	Fachgruppe Informatik: Annual Report 2012
2012-02	Thomas Heer: Controlling Development Processes
2012-03	Arne Haber, Jan Oliver Ringert, Bernhard Rumpe: MontiArc - Architectural Modeling of Interactive Distributed and Cyber-Physical Systems
2012-04	Marcus Gelderie: Strategy Machines and their Complexity

2012-05 Thomas Ströder, Fabian Emmes, Jürgen Giesl, Peter Schneider-Kamp, and Carsten Fuhs: Automated Complexity Analysis for Prolog by Term Rewriting

2012-06 Marc Brockschmidt, Richard Musiol, Carsten Otto, Jürgen Giesl: Automated Termination Proofs for Java Programs with Cyclic Data

2012-07 André Egners, Björn Marschollek, and Ulrike Meyer: Hackers in Your Pocket: A Survey of Smartphone Security Across Platforms

2012-08 Hongfei Fu: Computing Game Metrics on Markov Decision Processes

2012-09 Dennis Guck, Tingting Han, Joost-Pieter Katoen, and Martin R. Neuhäußer: Quantitative Timed Analysis of Interactive Markov Chains

2012-10 Uwe Naumann and Johannes Lotz: Algorithmic Differentiation of Numerical Methods: Tangent-Linear and Adjoint Direct Solvers for Systems of Linear Equations

2012-12 Jürgen Giesl, Thomas Ströder, Peter Schneider-Kamp, Fabian Emmes, and Carsten Fuhs: Symbolic Evaluation Graphs and Term Rewriting — A General Methodology for Analyzing Logic Programs

2012-15 Uwe Naumann, Johannes Lotz, Klaus Leppkes, and Markus Towara: Algorithmic Differentiation of Numerical Methods: Tangent-Linear and Adjoint Solvers for Systems of Nonlinear Equations

2012-16 Georg Neugebauer and Ulrike Meyer: SMC-MuSe: A Framework for Secure Multi-Party Computation on MultiSets

2012-17 Viet Yen Nguyen: Trustworthy Spacecraft Design Using Formal Methods

2013-01 * Fachgruppe Informatik: Annual Report 2013

2013-02 Michael Reke: Modellbasierte Entwicklung automobiler Steuerungssysteme in Klein- und mittelständischen Unternehmen

2013-03 Markus Towara and Uwe Naumann: A Discrete Adjoint Model for OpenFOAM

2013-04 Max Sagebaum, Nicolas R. Gauger, Uwe Naumann, Johannes Lotz, and Klaus Leppkes: Algorithmic Differentiation of a Complex C++ Code with Underlying Libraries

2013-05 Andreas Rausch and Marc Sihling: Software & Systems Engineering Essentials 2013

2013-06 Marc Brockschmidt, Byron Cook, and Carsten Fuhs: Better termination proving through cooperation

2013-07 André Stollenwerk: Ein modellbasiertes Sicherheitskonzept für die extrakorporale Lungenunterstützung

2013-08 Sebastian Junges, Ulrich Loup, Florian Corzilius and Erika Ábrahám: On Gröbner Bases in the Context of Satisfiability-Modulo-Theories Solving over the Real Numbers

2013-10 Joost-Pieter Katoen, Thomas Noll, Thomas Santen, Dirk Seifert, and Hao Wu: Performance Analysis of Computing Servers using Stochastic Petri Nets and Markov Automata

2013-12 Marc Brockschmidt, Fabian Emmes, Stephan Falke, Carsten Fuhs, and Jürgen Giesl: Alternating Runtime and Size Complexity Analysis of Integer Programs