Classifications of Recognizable Infinitary Trace Languages and the Distributed Synthesis Problem

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

M.Sc.
Namit Chaturvedi
aus Gwalior, Indien

Berichter: Universitätsprofessor Dr. Dr. h.c. Dr. h.c. Wolfgang Thomas
Professeur des universités Dr. Anca Muscholl

Tag der mündlichen Prüfung: 10.12.2014

Zusammenfassung

Die Klassifikation erkennbarer ω-Wortsprachen in Stufen der Borel-Hierarchie ist die Basis zahlreicher angepasster Lösungen in den Bereichen der formalen Verifikation und der algorithmischen Controller-Synthese. Jede dieser Stufen wird durch eine Klasse deterministischer ω-Automaten charakterisiert, nämlich durch deterministische schwache Automaten (Erreichbarkeits- und Sicherheitsbedingungen), deterministische Büchi-Automaten und deterministische Muller-Automaten. Die vorliegende Arbeit analysiert das allgemeinere Rahmenwerk der unendlichen Mazurkiewicz-Spuren, die als Modelle für nicht terminierende, nebenläufige Läufe verteilter Systeme dienen. Das Studium der Sprachen endlicher Spuren verallgemeinert das der Wortsprachen, und zahlreiche Ergebnisse haben auch eine Erweiterung auf Sprachen unendlicher Spuren erreicht (Gastin, Petit, Diekert, Muscholl und andere). Doch liefern die verfügbaren Definitionen von asynchronen ω-Automaten keine Klassifizierung der ω-Spursprachen, die mit der Borel-Hierarchie verträglich ist. Wir schließen diese seit den 1990'er Jahren bestehende Lücke durch die Einführung der Familie der „synchronisationsbewussten" Automaten.

Wir zeigen auch die Semi-Entscheidbarkeit des Problems, ob eine gegebene erkennbare ω-Spursprache durch einen deterministischen synchronisationsbewussten Büchi-Automaten erkannt werden kann.

Obwohl asynchrone Automaten in der Implementierung verteilter Monitore und verteilter Controller ihren Nutzen haben, ist ihre Konstruktion extrem aufwändig, sogar im Vergleich zu bekannten komplexen Verfahren für ω-Automaten. Dagegen finden „Linearisierungen" infinitärer Spursprachen, die auf Spur-abgeschlossenen ω-Wortsprachen beruhen, unmittelbare Anwendungen im Model-Checking und der formalen Verifikation verteilter Systeme. Dies hat seinen Grund darin, dass Wortautomaten, die Spur-abgeschlossene Sprachen erkennen, effizientere Analysen der für verteilte Systeme wesentlichen Eigenschaften ermöglichen. In diesem Rahmen präsentieren wir eine weitere Klassifikation von ω-Spursprachen in einer Borel-ähnlichen Hierarchie von Spur-abgeschlossenen ω-Wortsprachen.

Abschließend führen wir eine Version des Church'schen Synthese-Problems für verteilte Systeme ein und vergleichen es mit zwei bekannten Varianten der verteilten Controller-Synthese, nämlich der aktionsbasierten Kontrolle (Gastin, Lerman, Zeitoun) und der prozessbasierten Kontrolle (Madhusudan, Thiagarajan, Yang). Die algorithmische Lösung dieser Probleme bleibt zwar eine offene Frage, doch erweitern wir Ergebnisse von Muscholl, Walukiewicz und Zeitoun und vergleichen Ihre Problemklassen mit Varianten des Church'schen Syntheseproblems durch dem Nachweis von Reduktionen in dem Sinne, dass eine Lösung eines Problems in die Lösung eines anderen überführt werden kann.

Abstract

The classification of recognizable ω-word languages into Borel levels is the basis of many specialized solutions in the fields of formal verification and algorithmic controller synthesis. Each of these levels is characterized by a class of deterministic ω-automata, namely deterministic weak (reachability and safety), deterministic Büchi, and deterministic Muller automata. This thesis analyses the more general setting of infinitary Mazurkiewicz traces, which model nonterminating, concurrent computation of distributed systems. The study of finitary trace-languages generalizes that of word-languages, and numerous results have been extended to infinitary trace-languages (due to Gastin, Petit, Diekert, Muscholl, and others). However, the current definitions of asynchronous ω-automata fail to yield a classification of ω-trace languages that is compatible with the Borel hierarchy. We close this gap, which had been open since the 1990's, by introducing the family of "synchronization aware" automata.

We also demonstrate the semi-decidability of the problem to determine whether a given recognizable ω-trace language is also recognized by a deterministic synchronization aware Büchi automaton.

Although asynchronous automata are useful in implementing distributed monitors and distributed controllers, their constructions are prohibitively expensive even by automata-theoretic standards. On the other hand, "linearizations" of infinitary trace languages, which invoke the framework of "trace-closed" ω-word languages, can find immediate applications in model checking and formal verification of distributed systems. This is because word automata recognizing trace-closed languages support more efficient analyses of most of the interesting properties pertaining to distributed computations. In this setting, we present another classification of ω-trace languages in terms of a Borel-like hierarchy of trace-closed ω-word languages.

Finally, we introduce a distributed version of Church's synthesis problem and compare it with two well known variants of distributed controller synthesis, viz. action-based control (due to Gastin, Lerman, and Zeitoun) and process-based control (due to Madhusudan, Thiagarajan, and Yang). While the algorithmic solution of these problems remains an open area of investigation, we build upon the work of Muscholl, Walukiewicz, and Zeitoun, and compare their problem classes with variants of distributed Church's synthesis problem by demonstrating suitable reductions in the sense that a solution of any one problem can be converted into a solution of the other.

Acknowledgements

Above all, I would like to thank my wife Nidhi for her unflinching spirit, which in turn makes me strong. Without her encouragement, I would never have started my doctoral studies; without her sacrifices, her love and faith, I would never have completed them. Her constant support and motivation have held my morale and perseverance from ever ebbing.

I extend my sincere gratitude to my supervisor Wolfgang Thomas for offering me the opportunity to pursue this doctorate, and for the freedom in choosing my own dissertation topic. His guidance has always been invaluable toward improving my solution approaches. I am grateful to Marcus Gelderie, Jörg Olschewski, and Stefan Repke for being my first friends in a new country, and my ever helpful mentors in an equally novel field of theoretical research. I especially thank Christof Löding and Martin Lang for their patience in helping me become a better researcher and a better teacher.

I am deeply indebted to Anca Muscholl, not only for opening up the grounds for my present work, but also for her support and numerous constructive comments.

Contents

Introduction

Since the 1950's, mathematicians and computer scientists have studied and developed models of abstract machines with finite memory, the so called *finite-state automata*. A finite-state automaton derives its name from the finite number of internal "states" that it is comprised of. The concept of a state in a finite-state automaton corresponds directly to the concept of the abstract state that machine acquires during its computations. There are two different contexts in which the computations of a finite-state machine are considered. In the first setting, the computations are assumed to be finite. An automaton begins in one of the few states that are earmarked as *initial.* The notion of a "good" or an *acceptable* computation is defined in terms of whether or not the automaton reaches one of the *accepting* states at the end of the computation.

However, in many practical scenarios, it is important to consider nonterminating computations, for example, to analyze whether the system remains "always safe"(*safety* condition) or does "something good repeatedly" (*liveness* condition). In this setting, infinite computations are considered. A finite-state automaton evaluates such properties of infinite computations by observing whether or not the automaton always remains in the set of safe states or visits accepting states repeatedly.

Nonterminating Systems

The study of nonterminating systems is one of the central tasks of computer science, motivated by the ubiquitous use of such systems in all branches of information technology. In theoretical computer science, nonterminating state-based systems are both an established field and a very active research area. This theory has found applications in the two important fields of *formal verification* and *algorithmic synthesis* of computer systems. A large body of research exists on these topics which also extends the utility of this theory to the industrial setting.

This has been made possible owing to fundamental results on finite-state systems with nonterminating behavior, which were established already in the 1960s by J.R. Büchi, L. Landweber, R. McNaughton, M.O. Rabin, and others. They studied the close relationships between behaviors *recognized* by finite-state automata and behaviors that can be *described* in terms of various logical formalisms. They developed the theory of regular and recognizable sets of infinite words and infinite trees; and that of finite-state automata for the domains of infinite words and infinite trees [BE58, McN66, Rab69] (we refer the reader also to [Tho90a, Tho97, PP04] for surveys on these results).

These results offer a number of automata models to capture specific families of ω-*languages* – a convenient term for sets of nonterminating behaviors of computation systems. The families of ω-languages that can be described in monadic second order logic can be characterized in terms of deterministic Muller or, equivalently, in terms of nondeterministic Büchi automata. There exist characterizations of some important subclasses of languages by specific classes of automata, namely deterministic Büchi automata for ω-languages that capture "liveness" of finite properties. There also exists the family of weak automata that typifies ω-languages that describe Boolean combinations of "reachability" and "safety" languages.

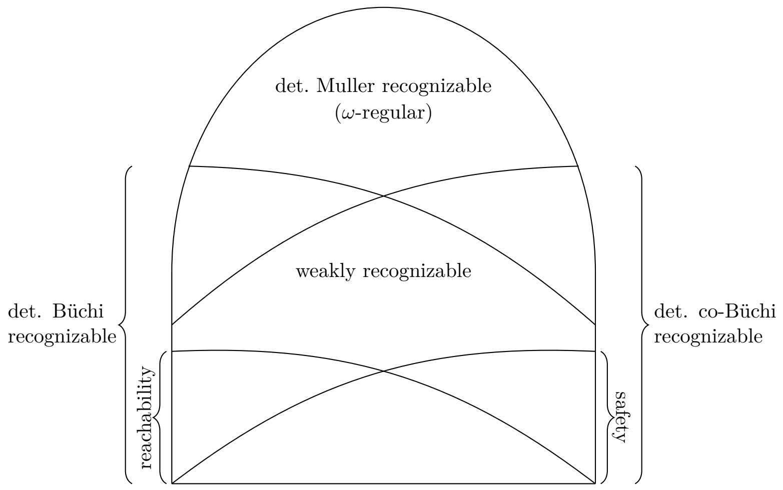


Illustration 1: Borel classification of recognizable ω-languages.

Primarily due to Landweber (cf. [PP04]), explorations into the classes of recognizable ω-languages, as shown in Illustration 1, have enriched the field by allowing for development of specialized and efficient techniques for solving verification and synthesis problems when restricted to the corresponding classes of languages.

Since the late 1980's, these concepts have also been studied in the broader setting of distributed systems. These systems represent situations where a number of independently functioning computation agents, called *processes*, achieve a common goal through repeated synchronizations and exchanges of information. This generalizes the above mentioned theory, because a sequential system is a special kind of distributed system which comprises of just one computation agent. In this thesis we study languages and automata for distributed systems, for which a number of the above-mentioned concepts are still missing.

A definition of finite behavior of distributed systems was first developed by A.W. Mazurkiewicz in 1984 and later formally presented in the form of the theory of Mazurkiewicz traces [Maz87], or simply *finite traces.* A finite trace can be viewed as a finite acyclic graph whose vertices are labeled by the actions of the system and where dependence between two occurrences of actions is detectable by the existence of a path between them. As indicated above, this is a generalization of the model of behaviors of sequential systems which can be viewed as a finite path graph, and where every pair of actions is ordered.

Through the late 1980's and the early 1990's, the theory of traces saw tremendous progress, spurred by a deep theorem by W. Zielonka [Zie87] that led to the development of the first model of automata for languages of finite traces, namely *deterministic asynchronous automata.* An asynchronous automaton captures the concurrent computations of a discrete, distributed system. It consists of a number of agents, each of which possesses a set of local states, and is responsible for processing a specific set of events. While processing certain events, a set of agents may come together, and jointly perform transitions to new local states. The concurrent nature of the computation is captured by the fact that certain events are declared to be *independent* of certain others. Two independent events are necessarily processed by disjoint sets of agents, and agents responsible for processing dependent events must necessarily perform the state transitions in cooperation with each other. Whether or not a finite trace is acceptable is decided by referring to the final local states of the agents at the end of processing the trace.

Zielonka also showed, using an intricate construction, that recognizable languages of finite traces have a nice correspondence with recognizable languages of finite words (cf. [Mad12] for a gentle introduction). This is captured by the twin notions of *trace-closed languages* of words and *I-diamond automata* over words. His results showed that a language of finite traces is recognized by an asynchronous automaton if and only if the corresponding trace-closed language is recognized by an I-diamond finite-state automaton.

Subsequently[1], Mazurkiewicz also provided a first definition of infinite traces, and Gastin, Petit, Diekert, Muscholl, and others, contributed to important results regarding recognizable ω-trace languages and models of deterministic asynchronous Muller automata and deterministic asynchronous Büchi automata [GP92a, DM94, Mus94]. Diekert and Muscholl also established the correspondence between recognizable ω-trace languages and trace-closed ω-word languages that are recognized by a specific subclass of I-diamond Muller automata [DM94, Mus94].

However, this theory has not yielded to a characterization of any specific subclass of recognizable ω-trace languages in the manner that the theory of automata for sequential languages has. The characterizations that we refer to are as shown in Illustration 1 above. For example, it is unknown what class of recognizable ω-trace languages is characterized deterministic asynchronous Büchi automata. Although Muscholl introduced the family of *deterministic trace languages* [DM94], which generalizes the deterministically Büchi recognizable word languages, there does not yet exist any matching family of deterministic asynchronous ω-automata.

A well-rounded theory of recognizable ω-trace languages– inasmuch as it is seen as a generalization of the theory of recognizable ω-languages– is the primary focus of the present thesis.

[1]A comprehensive survey of early results regarding languages of finite and infinite traces, and the corresponding asynchronous automata models can be found in [DR95].

Contributions of this Thesis

Some fundamental facts about recognizable ω-languages, as illustrated in the above figure, can be summarized as follows. This family is structured into a hierarchy where each level is characterized by a class of deterministic ω-automata. The lowest levels in this hierarchy are occupied by reachability and safety languages, whose finite Boolean combinations are exactly the languages that are recognized by deterministic weak automata. For this reason, these Boolean combinations are referred to as weakly recognizable languages. These languages can also be identified as those that are recognized by both, deterministic Büchi and deterministic co-Büchi automata.

Deterministically Büchi and deterministically co-Büchi recognizable languages occupy the next level in the hierarchy of recognizable ω-languages; and Boolean combinations of these languages yield the entire class of recognizable ω-languages. This latter class is ultimately characterized by deterministic Muller automata. Furthermore, an important result due to L. Landweber states that given any deterministic Muller automaton, it is decidable whether or not the language that it recognizes is also recognized by a deterministic Büchi automaton [PP04].

Through the main results of this thesis, we initiate a generalization of this classification to the case of recognizable infinitary trace languages.

Classification by Asynchronous Automata

When agents of an asynchronous automaton synchronize in order to process an event, it is possible for them to summarize and exchange their causal histories with one another. In this manner, one agent can obtain others' information and "learn" about the events that had been absent from its own causal view. Events in which the agents participate can be ordered according to their importance. For example, events where agents learn about other agents are more important that events where agents perform the processing in isolation. The more information that is exchanged at an event, the more important the event becomes.

In this thesis, we pursue the principle that during an infinite run of an asynchronous automaton, each agent must only concentrate on the most important events that it participates in, and ignore all events that are of lesser importance. Whether an infinite trace is acceptable must then be decided by only consulting the local states that appear at the most important events. In this vein, we introduce a new class of asynchronous automata called *synchronization aware automata*.

By equipping these automata with a Büchi acceptance condition, we obtain the definition of *deterministic synchronization aware Büchi automata*, in short *D-SABA*, which are able to characterize the class of *deterministic trace languages* due to Muscholl. The corresponding class of *deterministic synchronization aware Muller automata*, or *D-SAMA*, captures all the recognizable ω-trace languages. Not only is every recognizable ω-trace language expressible as a Boolean combination of D-SABA recognizable languages, but we also obtain a result which makes it

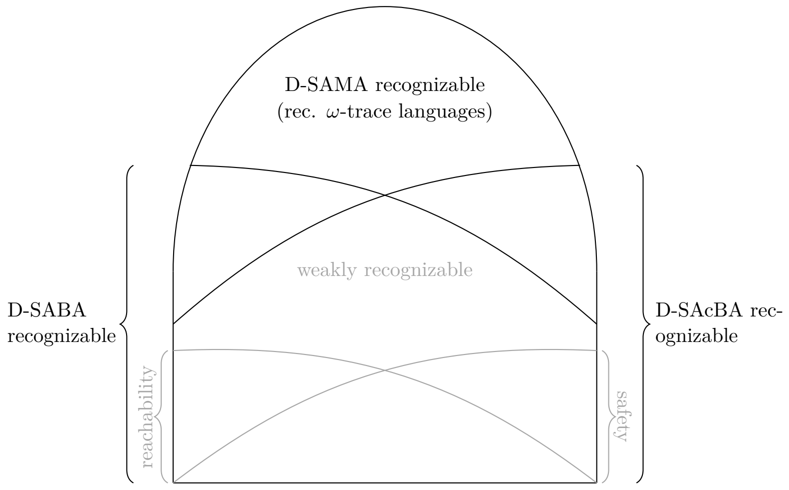


Illustration 2: A classification of recognizable ω-trace languages.

semi-decidable whether or not such a language is also D-SABA recognizable.

Consequently, as shown in Illustration 2, we are able to settle the question of describing the second level of the structure hierarchy of recognizable ω-trace languages in automata theoretic terms. We conjecture that the first level of reachability and safety trace languages cannot be characterized in terms of asynchronous automata. This is because the acceptance condition in an asynchronous "reachability" automaton must refer to local states reached by the processes, but recognition of an infinite trace would require checking for reachability of a global state.

On the other hand, synchronization aware automata generalize the definition of all finite-state automata. This is because an automaton over words can be viewed as a single-agent distributed system, where all events are equally important in terms of the above-mentioned criterion of information-exchange.

The results pertaining to this part of the thesis were first announced at the 41st International Colloquium on Automata, Languages, and Programming [Cha14b], and the algorithms were presented in detail in a technical report [Cha14a].

Classification by Word Automata

As mentioned previously, Zielonka demonstrated an intricate relationship between recognizable languages of finite traces and those of finite words. It must be noted that this relationship extends to the infinitary case as well [GP92b, DM94]. Zielonka showed that is possible to study a trace language by referring to the set of words that comprises of "linearizations" of the partially ordered events of the traces contained in the language. Two words are said to be *trace-equivalent* if they are the linearizations of the same trace.

Such a set of trace-equivalent words, containing linearizations of all traces of a trace language, is called a *trace-closed language*. The finite-state automata that recognize trace-closed languages have a basic structural property known as the

I-diamond property, which implies that if a, b are two independent letters, then starting at any state x, the automaton reaches the same state y upon processing the words ab and ba. Thus, the I-diamond property ensures that all trace-equivalent (finite) words induce equivalent finitary behaviors on the automaton.

In this sense, classes of I-diamond weak automata, I-diamond Büchi automata, and I-diamond Muller automata ensure that all trace-equivalent infinite words induce equivalent infinitary behaviors on the ω-automata. Subsequently, these automata offer an opportunity to explore the corresponding classes of trace-closed recognizable ω-languages.

In this setting, we characterize recognizable trace-closed ω-word languages at all Borel levels, which is nicely analogous to the case of word languages. At the lowest level, we have trace-closed reachability and safety languages whose Boolean combinations are recognized by I-diamond weak automata. At the next level we have trace-closed languages that are characterized by the class of the so called *F, I-cycle closed Büchi automata*. Boolean combinations of these languages are recognized by *$\mathcal{F}, I$-cycle closed Muller automata*, which also characterize all trace-closed recognizable ω-languages– the class of trace-closed recognizable ω-languages being in one-to-one correspondence with the class of recognizable ω-trace languages. This is as shown in Illustration 3.

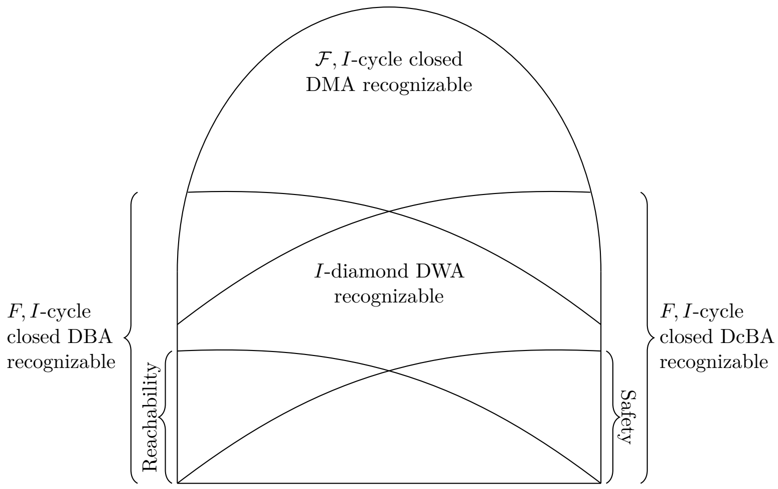

$\mathcal{F}, I$-cycle closed DMA recognizable
I-diamond DWA recognizable
F, I-cycle closed DBA recognizable
F, I-cycle closed DcBA recognizable
Reachability
Safety

Illustration 3: A classification of trace-closed recognizable ω-languages.

Observe that the level two in the above figure looks very similar to the level two in Illustration 2. However, it must be noted here that the latter strictly subsumes the former. That is, each language recognized by an F, I-cycle closed DBA is a linearization of a D-SABA recognizable trace language; but there exist D-SABA recognizable trace languages whose linearizations are not recognizable by any F, I-cycle closed DBA. It is still remarkable that any recognizable ω-trace language can be expressed as a finite Boolean combination of languages in either of these classes. In other words, as far as describing recognizable ω-trace languages is concerned, D-SABA recognizable trace languages are no more expressive than F, I-cycle closed DBA recognizable languages.

Most of the results regarding recognizable trace-closed ω-word languages were first announced in [CG14b]. They were further developed in a paper that appears in the 39th International Symposium on Mathematical Foundations of Computer Science [CG14a].

On the Distributed Synthesis Problem

One of the primary motivations for the investigations undertaken in this thesis was the problem of distributed synthesis. As mentioned in the beginning, a hierarchical characterization of recognizable ω-languages in terms of ω-automata has greatly helped in developing specialized algorithms for verification and synthesis. It still remains a challenge to develop similar algorithmic solutions for synthesis problems specified in terms of classes of recognizable ω-trace languages.

The synthesis problem for recognizable ω-languages is studied in two equivalent variants, namely, the Church's synthesis problem [Chu57] (see also [Tho08] for a modern exposition) and the Ramadge-Wonham controller synthesis problem [RW89]. Underlying both variants is a *system* with a well defined behavior. The controller synthesis problem asks whether it is possible to algorithmically construct a *controller* for the system that satisfies a given *specification* in the presence of an adversarial *environment*. A fundamental result by Büchi and Landweber states that for systems defined as finite-state machines and specifications provided in terms of recognizable ω-languages, the controller synthesis problem is decidable, and if there exists a controller then there exists one with finite memory [BL69, Tho08].

The Ramadge-Wonham setting has been naturally extended by considering systems defined as asynchronous automata and specifications provided as recognizable ω-trace languages. The concept of a controller, however, has been extended into two distributed variants, namely, *process-based controllers*[MTY05] and *action-based controllers*[GLZ04]. It is generally believed that action-based controllers are stronger than the process-based controllers in the sense that for certain problem instances it seems possible to construct the former but not the latter. the problem of deciding whether or not there exists a distributed controller remains open (cf. [MWZ09] and references therein). Under suitably restricting either the system or the specification or both, a number of decidability results exist in the literature [MT01, GLZ04, MTY05, GLZ05] (see [MWZ09] for further pointers).

Apart from the distributed Ramadge-Wonham setting, there exists an additional body of literature regarding distributed controller synthesis that lays special emphasis on "distributed architectures" of systems. In these investigations, communication channels between various components of the system often do not correlate with the mutual independence of events that constitute the specifications. This is a primary reason why the synthesis problem for a large class of such architectures remains undecidable [PR90, FS13, FP14, Sch14].

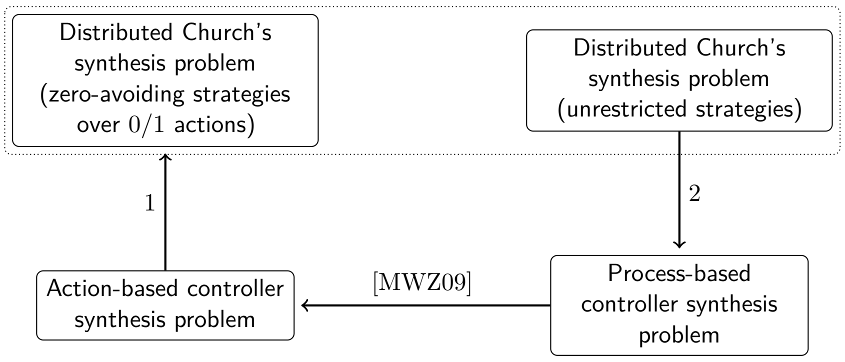

Distributed Church's synthesis problem (zero-avoiding strategies over $0/1$ actions)
Distributed Church's synthesis problem (unrestricted strategies)
1
2
Action-based controller synthesis problem
[MWZ09]
Process-based controller synthesis problem

Illustration 4: Reductions between different variants of the distributed controller synthesis problem.

In this thesis, we concentrate on the variants where system architectures and trace specifications agree on independent events. First, we introduce a definition of "distributed Church's synthesis problem", a variant that has as yes not received attention in the literature. Second, while we are presently unable to provide any solutions for the general or restricted version, we do demonstrate that the distributed Church's synthesis problem is the weakest of them all. Moreover, if we restrict the distributed Church's problem to finding "zero-avoiding" strategies over $0/1$ actions, then this variant subsumes all other problems.

These relationships are proved by demonstrating the reductions as shown in Illustration 4. Here we say that a problem Φ can be reduced to a problem Ψ if for every instance φ of Φ one can construct an instance ψ of Ψ, such that there exists a solution for φ if and only if there exists a solution for ψ. In this manner, a reduction implies that the task of developing solution techniques for Φ is at most as hard as the task of developing solution techniques for Ψ.

One of these reductions was already established in [MWZ09], and we complete the picture by providing reductions 1 and 2. This immediately lays a groundwork for collating all the existing results under a single context. Additionally, this equivalence opens up the question of comparing these variants at the more refined Borel levels, for example, by considering only specifications that can be described as deterministic trace languages.

Chapter 1

Word Languages and Recognizability

In this chapter we provide an overview of the main results pertaining to recognizable ω-languages of words, their connections with recognizable languages of (finite) words, and their characterization in terms of families of deterministic automata. We restrict ourselves to language and automata theoretic concepts, and we particularly avoid a discussion of the corresponding results in terms of logic. We refer the reader to [Tho97, Tho90a] for a detailed exposition on the relationship between languages, automata, and logic.

We expect the reader to be familiar with the basic definitions and concepts regarding recognizable languages of finite words. Here, we first fix a notation and then present the important results on languages of infinite words. While we cover certain algebraic aspects, our emphasis remains in the theory of deterministic automata for these languages – an aspect that we generalize later to the case of languages of infinite traces.

1.1 Words, Operators, and Languages

Given a finite alphabet Σ, a *finite word* is a sequence $w = a_1a_2 \cdots a_n$ of letters $n \in \mathbb{N}, a_i \in \Sigma$. The number n is referred to as the *length* of the word w, commonly denoted $|w| = n$. A word with length 0 is called the *empty word*, denoted ε. It is the empty sequence which contains no letters. We denote the set of all finite words (including the empty word) as Σ^*.

On the other hand, an infinite sequence $\alpha = a_1a_2 \cdots$ of letters $a_i \in \mathbb{N}$ is called an *infinite word* or equivalently an *ω-word*. An ω-word is denoted with lower case Greek letters α and β; and the set of all infinite words is denoted Σ^ω.

Finite words may be alternatively looked upon as mappings $w\colon |w| \to \Sigma$, and at infinite words as mappings $\alpha\colon \mathbb{N} \to \Sigma$, which map each position of the word to a letter from Σ. In this sense, it is possible to refer to the i^{th} letter $w[i]$ of the word $w \in \Sigma^*$, assuming $1 \le i \le |w|$; and similarly we can refer to the i^{th} letter $\alpha[i]$ of the infinite word $\alpha \in \Sigma^\omega$. For simplicity, whenever we refer to the letters $w[i]$, we assume that $1 \le i \le |w|$.

There also exist notions of subsequences appearing within a word. A word $v \in \Sigma^*$ is a *prefix* of $w \in \Sigma^*$, denoted $v \sqsubseteq w$ or $w \sqsupseteq v$, if $v = w[1, \ell]$, that is v is the sequence of the first ℓ letters of w. Similarly, v is a *suffix* of w if $v = w[\ell, |w|]$, that is v is the sequence of the last letters – position ℓ onwards – of w. Finally, a word v is an *infix* of w if $v = w[i, j]$ for some $1 \leq i, j \leq |w|$. Analogously, for ω-words $\alpha \in \Sigma^\omega$, we define prefixes $v \in \Sigma^*$, infixes $v \in \Sigma^*$, and suffixes $\beta \in \Sigma^\omega$.

The *concatenation* of two words v and w is a word u constructed by appending the sequence of letters comprising v with the sequence of letters comprising w, and is denoted as $u = v \cdot w$. We analogously define the concatenation $v \cdot \alpha$ of a finite word v and an infinite α. However, a concatenation $\alpha \cdot \beta$ of two ω-words α and β is not an ω-word. In favor of convenience, we usually denote the concatenation $u \cdot v$ simply as uv.

A set $K \subseteq \Sigma^*$ is called a *language over* Σ, and a set $L \subseteq \Sigma^\omega$ is referred to as an *ω-language over* Σ. We can now generalize the concatenation operator to languages. That is, given languages $K_1, K_2 \subseteq \Sigma^*$ and an ω-language $L \subseteq \Sigma^\omega$, the language concatenation $K_1 \cdot K_2 := \{w_1 w_2 \in \Sigma^* \mid w_1 \in K_1, w_2 \in K_2\}$ is a language, and the concatenation $K_1 \cdot L := \{w\alpha \in \ \Sigma^\omega | w \in K_1, \alpha \in L\}$ is an ω-language.

Starting with with individual letters, the concatenation operator is one way of inductively describing languages and ω-languages. However, there exist more operators to make this task easier and intuitive. These operators are as follows:

- *Kleene star* This operator may be applied to sets $A \subseteq \Sigma$ or languages $K \subseteq \Sigma^*$ to define
  - $A^* := \{w \in \Sigma^* \mid w = \varepsilon$ or w consists only of letters from $A\}$, and
  - $K^* := \{w_1 w_2 \dots w_n \in \Sigma^* \mid n \in \mathbb{N}, w_i \in K$ or $w_i = \varepsilon\}$.
- *Kleene plus*: Like the Kleene star, this operator may be applied to sets $A \subseteq \Sigma$ or languages $K \subseteq \Sigma^*$, only that languages A^+ and K^+ exclude the empty word ε.
- *Bounded iteration*: In contrast to the above operators that consider unbounded sequences of concatenations, we define, for a fixed $\ell \in \mathbb{N}$,
  - $A^\ell := \{a_1 a_2 \dots a_\ell \in \Sigma^* \mid$ for $1 \leq i \leq \ell, a_i \in A\}$, and
  - $K^\ell := \{w_1 w_2 \dots w_\ell \in \Sigma^* \mid$ for $1 \leq i \leq \ell, w_i \in K\}$.
- *ω-iteration*: Finally, we consider the operator that induces infinitely many iterations and, for $A \subseteq \Sigma$ and $K \subseteq \Sigma^*$, define
  - $A^\omega := \{a_1 a_2 \dots \in \Sigma^\omega \mid$ for $i \in \mathbb{N}, a_i \in A\}$, and
  - $K^\omega := \{w_1 w_2 \dots \in \Sigma^\omega \mid$ for $i \in \mathbb{N}, w_i \in K\}$.
- *Union*: For languages $K_1, K_2 \subseteq \Sigma^*$, define the union $K_1 \cup K_2 := \{w \in \Sigma^* \mid w \in K_1$ or $w \in K_2\}$. Similarly, we define the union $L_1 \cup L_2$ for $L_1, L_2 \subseteq \Sigma^\omega$.
- *Complement*: For a language $K \subseteq \Sigma^*$, define $\overline{K} := \{w \in \Sigma^* \mid w \notin K\}$. For a language $L \subseteq \Sigma^\omega$, define $\overline{L} := \{\alpha \in \Sigma^\omega \mid \alpha \notin L\}$.

Apart from the operators defined above, we introduce two more. And although these may be obtained by combining the ones above, we often use them for the sake of succinctness.

- *Intersection*: For languages $K_1, K_2 \subseteq \Sigma^*$, define the intersection $K_1 \cap K_2 \coloneqq \{w \in \Sigma^* \mid w \in K_1 \text{ and } w \in K_2\}$. Similarly, we define the intersection $L_1 \cap L_2$ for $L_1, L_2 \subseteq \Sigma^\omega$.

- *Difference*: For languages $K_1, K_2 \subseteq \Sigma^*$, define the difference $K_1 \setminus K_2 \coloneqq \{w \in \Sigma^* \mid w \in K_1 \text{ and } w \notin K_2\}$. Similarly, we define the intersection $L_1 \setminus L_2$ for $L_1, L_2 \subseteq \Sigma^\omega$.

Example 1.1 *Let $\Sigma = \{a, b, c\}$ be a finite alphabet.*

1. $K_1 \coloneqq \bigcup_{\ell \in \mathbb{N}} \{a\}^\ell \{b\}^\ell \setminus \{\varepsilon\}$ *is a language containing non-empty words that comprise of equal number of letters a and b appearing in that order.*

2. $L_1 \coloneqq K_1 \cdot \{b, c\}^\omega$ *is an ω-language containing words α such that α has a prefix $w \in K_1$, followed by infinitely many b's or c's.*

3. $K_2 \coloneqq \left(\{b,c\}^* \cdot \{a\} \cdot \{b,c\}^* \cdot \{a\} \cdot \{b,c\}^*\right)^+ \cap \left(\{a,c\}^* \cdot \{b\} \cdot \{a,c\}^* \cdot \{b\} \cdot \{a,c\}^*\right)^+$ *is a language containing all the words that have an even (non-zero) number of a's and an even (non-zero) number of b's.*

4. $K_3 = \overline{K_2}$ *is the languages whose non-empty words either contain an odd number of a's or an odd number of b's or both.*

5. $L_2 \coloneqq K_2^\omega$ *is the ω-language containing infinitely many infixes (or, as it turns out in this particular case, infinitely many prefixes) with even number of a's and even number of b's.*

In this thesis, we will only concentrate on languages that can be obtained as a result of a finite combination of operators defined above. For example, language K_1, and consequently the language L_1, requires a union of infinitely many languages. In fact, these languages cannot be described otherwise. Such languages are beyond the scope of this thesis.

Only languages of the former kind, those which can be expressed with the help of finitely many operators, can be characterized in terms of abstract machines that have finitely many states. We formalize this notion in the following section.

1.2 Languages and Automata

For both, the class of languages and the class of ω-languages, there exist notions of abstract machines, or finite state automata, which can acquire a fixed number of states, and can move from one state to another upon witnessing different letters from the alphabet. A finite state automaton consists of a labeled transition system and an acceptance condition. We present these separately.

A *deterministic, transition system* (a *DTS* in short), is a tuple $\mathfrak{T} = (Q, \Sigma, \delta, q_0)$ comprising of a set Q of states, a finite alphabet Σ, a transition function $\delta \colon Q \times \Sigma \to Q$, and an initial state q_0. A DTS $\mathfrak{T}$ "processes" a word $w = a_1 a_2 \ldots a_n$

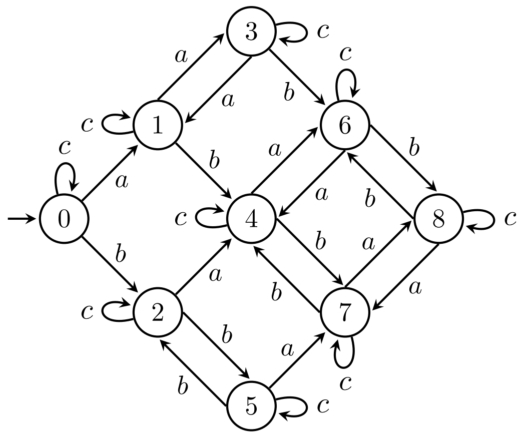


Figure 1.1: The deterministic transition system $\mathfrak{T}$ underlying the automata that accept languages K_2, K_3, and L_2 described in Example 1.1

by starting in its initial state q_0 and performing subsequent state transitions at letters a_i by referring to the transition function δ. This gives us a view of a *run* $\rho := q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$ of $\mathfrak{A}$ on the word w. The state q_n acquired by the DTS at the end of a finite run ρ is called the *final state*. The notion of an *infinite run* of a DTS on an ω-word $\alpha = a_1a_2\ldots$ is exactly the same, except that there does not exist any final state.

A *deterministic, finite state automaton*, referred to as a *DFA* in short, is a pair $\mathfrak{A} = (\mathfrak{T}, F)$ where $\mathfrak{T}$ is a DTS and the set $F \subseteq Q$ is referred to as the set of *accepting states*. By the run of a DFA, we mean the run of the underlying DTS. The acceptance of a word $w \in \Sigma^*$ is defined by referring to the run ρ of $\mathfrak{A}$ on w. The word w is *accepted* by $\mathfrak{A}$ if $q_n \in F$. The language $L(\mathfrak{A})$ *recognized* by a DFA $\mathfrak{A}$ is defined as $L(\mathfrak{A}) := \{w \in \Sigma^* \mid w \text{ is accepted by } \mathfrak{A}\}$.

Definition 1.2 *A language $K \subseteq \Sigma^*$ is called a* recognizable language *if there exists a DFA $\mathfrak{A}$ such that $L(\mathfrak{A}) = K$.*

Example 1.3 *Over the alphabet $\Sigma = \{a, b, c\}$, let $\mathfrak{T} = (Q, \Sigma, \delta, 0)$ be the DTS shown in Figure 1.1. With respect to the languages K_2 and K_3 described in Example 1.1, we have that:*

1. *the DFA $\mathfrak{A}_2 = (\mathfrak{T}, F_2)$, with $F_2 = \{8\}$, recognizes the language K_2; and*

2. *the DFA $\mathfrak{A}_3 = (\mathfrak{T}, F_2)$, with $F_3 = Q \setminus F_2$, recognizes the language K_3.*

In the context of ω-languages, we again refer to an automaton by referring to its underlying transition systems and an associated acceptance condition. However, since infinite words induce infinite runs, the acceptance of ω-words is defined in terms of the (sets of) states that are visited infinitely often during the runs. Given a word $\alpha = a_1a_2\ldots \in \Sigma^\omega$ and a DTS $\mathfrak{T}$, consider the *infinite run* $\rho = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2}$

$\cdots \xrightarrow{a_i} q_i \xrightarrow{a_{i+1}} \cdots$ of $\mathfrak{T}$ over α. We define the *infinitary set* $\mathsf{Inf}(\rho) := \{q \in Q \mid$ there exist infinitely many $i\colon \delta(q_{i-1}, a_i) = q\}$.

A *deterministic Muller automaton*, referred to as a *DMA* in short, is a pair $\mathfrak{M} = (\mathfrak{T}, \mathcal{F})$ comprising of a DTS $\mathfrak{T}$, and the *acceptance component* $\mathcal{F} \subseteq 2^Q$ of sets of accepting states. An ω-word α is *accepted* by a DMA $\mathfrak{M}$ if there exists a set $F \in \mathcal{F}$ such that, for the run ρ of $\mathfrak{M}$ over α, $\mathsf{Inf}(\rho) = F$. The language $L(\mathfrak{M})$ *recognized* by a DMA $\mathfrak{M}$ is defined as $L(\mathfrak{M}) := \{\alpha \in \Sigma^\omega \mid \alpha$ is accepted by $\mathfrak{M}\}$.

Definition 1.4 *A language $L \subseteq \Sigma^\omega$ is called a* recognizable ω-language *if there exists a DMA $\mathfrak{M}$ such that $L(\mathfrak{A}) = K$.*

Example 1.5 *Over the alphabet $\Sigma = \{a, b, c\}$, let $\mathfrak{T} = (Q, \Sigma, \delta, 0)$ be the DTS shown in Figure 1.1. With respect to the language L_2 described in Example 1.1, the DMA $\mathfrak{M}_2 = (\mathfrak{T}, \mathcal{F}_2)$, with $\mathcal{F}_2 = \{\{4,6,8\}, \{4,7,8\}, \{6,7,8\}, \{4,6,7,8\}\}$, recognizes the language L_2.*

Note that the acceptance component $\mathcal{F}_2$ in the example above may have also included more sets, for example $\{3,4,6,8\}$ or $\{1,2,6,7,8\}$. However, here we consider only those acceptance sets $F \in \mathcal{F}_2$ that are *realizable*, i.e. there exists an infinite run ρ of $\mathfrak{M}$ such that $\mathsf{Inf}(\rho) = F$. Henceforth, unless stated explicitly, we assume that the acceptance component of a Muller automaton is defined only in terms of realizable sets of states.

1.3 From Finitary to Infinitary Languages

The ω-iteration, as defined in Section 1.1, is an operator that allows us to define ω-languages solely in terms of languages of finite words. A fundamental characterization of recognizable ω-languages states that every such language $L \subseteq \Sigma^\omega$ can be expressed as a finite union $\bigcup_{i=1}^{n} U_i \cdot V_i^\omega$ where $n \in \mathbb{N}$, and the languages $U_i, V_i \subseteq \Sigma^*$ are recognizable languages.

This characterization however do not always lends itself to intuitive descriptions of recognizable ω-languages. Here, we present two more operators that help us in defining recognizable ω-languages with the help of recognizable languages, although these definitions can in general be applied to all languages of finite words. Given languages $K \subseteq \Sigma^*$, we can define ω-languages:

- $\mathsf{ext}(K) := K \cdot \Sigma^\omega = \{\alpha \in \Sigma^\omega \mid \alpha \text{ has a prefix } v, \text{ such that } v \in K\}$; and
- $\mathsf{lim}(K) := \{\alpha \in \Sigma^\omega \mid \alpha$ has infinitely many prefixes in $K\}$.

The language $\mathsf{ext}(K)$ is called the *infinitary extension of K* and the language $\mathsf{lim}(K)$ is called the *infinitary limit of K*. Let $\mathcal{K} \subseteq 2^{\Sigma^*}$ be a family of languages, we often refer to families $\mathsf{ext}(\mathcal{K}) := \{\mathsf{ext}(K) \mid K \in \mathcal{K}\}$ and $\mathsf{lim}(\mathcal{K}) := \{\mathsf{lim}(K) \mid K \in \mathcal{K}\}$ of ω-languages. For the class $\mathsf{REG} \subseteq 2^{\Sigma^*}$ of all recognizable languages of finite words, we introduce two classes of ω-automata which recognize the families $\mathsf{ext}(\mathsf{REG})$ and $\mathsf{lim}(\mathsf{REG})$ of recognizable ω-languages.

A *deterministic Büchi automaton*, or a *DBA* in short, is a tuple $\mathfrak{B} = (\mathfrak{T}, F)$ where $\mathfrak{T}$ is a DTS and the *acceptance component* $F \subseteq Q$ is a set of accepting states. An ω-word α is *accepted* by a DBA $\mathfrak{B}$ if for the run ρ of $\mathfrak{B}$ over α, $\mathsf{Inf}(\rho) \cap F \neq \emptyset$. A language $L \subseteq \Sigma^\omega$ is said to be *deterministically Büchi recognizable* if there exists a DBA $\mathfrak{B}$ such that $L(\mathfrak{B}) = L$.

Example 1.6 *Referring to the language K_2 defined in Example 1.1, consider the language $L_3 := \mathsf{lim}(K_2)$. Now L_3 is recognized by the DBA $\mathfrak{B}_3 = (\mathfrak{T}, F)$, where $\mathfrak{T}$ is the DTS shown in Figure 1.1 and $F = \{8\}$.*

Indeed, the words $aabbc^\omega, (abc)^\omega, acbc(ab)^\omega \in \mathsf{lim}(K_2)$ since they all contain infinitely many prefixes with (non-zero) even numbers of a's and b's.

Looking at the DBA $\mathfrak{B}_3$ defined above and the DFA $\mathfrak{A}_2$ defined in Example 1.3, it is not a coincidence that their signatures $(\mathfrak{T}, F)$ are identical. This is in fact a well known result in the theory of recognizable ω-languages.

Theorem 1.7 *A language $L \subseteq \Sigma^\omega$ is recognized by a DBA $\mathfrak{A}$ if and only if it can be expressed as $\mathsf{lim}(K)$ for some $K \subseteq \Sigma^*$. Moreover, if a DFA $\mathfrak{B} = (\mathfrak{T}, F)$ recognizes K, then the DBA $\mathfrak{A} = (\mathfrak{T}, F)$ recognizes L.*

Remark 1.8 *The class $\mathsf{lim}(\mathsf{REG})$ of deterministically Büchi recognizable languages is closed under finite unions and finite intersections, but not under complements.*

The class of DBA recognizable languages is important for a number of reasons. Not only are these languages intuitive to define but, as the above theorem states, the deterministic automata recognizing these languages can also be obtained in a straightforward manner. Moreover, as the following theorems state respectively, the class of DBA recognizable languages is expressive enough to generate the entire class of recognizable ω-languages, and at the same time it is a decidable subclass of recognizable ω-languages. The following theorem is due to Büchi & McNaughton.

Theorem 1.9 (see [PP04]) *Each recognizable ω-language can be expressed as a finite Boolean combination of DBA recognizable languages.*

The decidability of DBA recognizable languages comes from their characterization in terms of a special class of DMAs. Given a DMA $\mathfrak{M} = (\mathfrak{T}, \mathcal{F})$, we say that the acceptance component $\mathcal{F}$ is *closed under supersets* if any two sets $F, F' \subseteq Q$: $F \in \mathcal{F} \wedge F \subseteq F' \Rightarrow F' \in \mathcal{F}$. Recall the assumption that the acceptance component $\mathcal{F}$ contains only realizable sets of states. Due to Landweber, we have the following result.

Theorem 1.10 (see [PP04]) *A language $L \subseteq \Sigma^\omega$ is recognized by a DBA if and only if for any DMA $\mathfrak{M}$ recognizing L, the acceptance component of $\mathfrak{M}$ is closed under supersets.*

Example 1.11 *Referring back to Example 1.1, we can claim that the language L_2 is DBA recognizable, since the acceptance component of the DMA $\mathfrak{M}_2$ shown in Example 1.5 is closed under supersets.*

We leave it as an exercise to the reader to construct a DBA recognizing L_2.

The lim operator therefore not only provides us a way of describing recognizable ω-languages, but also provides an intuitive correspondence between the DFA and the ω-automaton recognizing the ω-language so generated. This same essence is captured by the ext operator. As we will see, the class $\mathsf{ext}(\mathsf{REG})$ of infinitary extensions of recognizable languages is a strict subclass of $\mathsf{lim}(\mathsf{REG})$.

A *deterministic Weak automaton*, or a *DWA*, $\mathfrak{B} = (\mathfrak{T}, F)$ is essentially a DBA where every strongly connected component of the transition system $\mathfrak{T}$ has only accepting states or only rejecting states. Given the minimal DFA $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$ recognizing K, a DWA $\mathfrak{B} := (Q', \Sigma, q_0, \delta', F')$ recognizing $\mathsf{ext}(K)$, respectively $\overline{\mathsf{ext}(K)}$, can be constructed as follows:

1. For a symbol $\perp \notin Q$, define $Q' := (Q \setminus F) \cup \{\perp\}$.
2. For each $q \in Q', a \in \Sigma$, define $\delta'(q,a) := \begin{cases} \delta(q,a) & \text{if } q \neq \perp \text{ and } \delta(q,a) \notin F, \\ \perp & \text{otherwise.} \end{cases}$
3. Define $F' := \{\perp\}$, respectively $F' := Q' \setminus \{\perp\}$

We say that an ω-language is *weakly recognizable* if it is recognized by a DWA. It is not difficult to demonstrate that the family of DWA recognizable languages is closed under finite Boolean operations. That is, the set BC($\mathsf{ext}(\mathsf{REG})$) of finite Boolean combinations of languages in $\mathsf{ext}(\mathsf{REG})$ is exactly the set of weakly recognizable languages [Sta83].

Moreover, for an ω-language L, define a congruence $\sim_L \subseteq \Sigma^* \times \Sigma^*$ where $u \sim_L v \Leftrightarrow \forall \alpha \in \Sigma^\omega, u\alpha \in L$ iff $v\alpha \in L$. If L is recognized by a DWA then this congruence has a finite index. This observation is crucial in obtaining a unique minimal DWA for a weakly recognizable language.

Theorem 1.12 (The Minimal DWA [Löd01]) *For a weakly recognizable language L, if M is the index of the congruence defined above, then L is recognized by a DWA $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$ with $|Q| = M$. Also, for every $q \in Q$ there exists a word $u_q \in \Sigma^*$ such that for each $u \in \Sigma^*, \delta(q_0, u) = q$ iff $u \in [u_q]_{\sim_L}$.*

Similar to the decidability of languages in $\mathsf{lim}(\mathsf{REG})$, we can refer to structural properties of Muller automata and of subsequently obtained Büchi automata to decide whether or not a recognizable ω-language is weakly recognizable. But it is possible to go still finer and decide whether or not a recognizable ω-language can be expressed as an infinitary extension of some recognizable language.

For a Muller automaton $\mathfrak{M} = (\mathfrak{T}, \mathcal{F})$, we say that the acceptance component $\mathcal{F}$ is *closed under reachable sets* if for two realizable infinitary sets $F, F' \subseteq Q$, if $F \in \mathcal{F}$ and F' is reachable from F then $F' \in \mathcal{F}$. The following characterization is again due to Landweber.

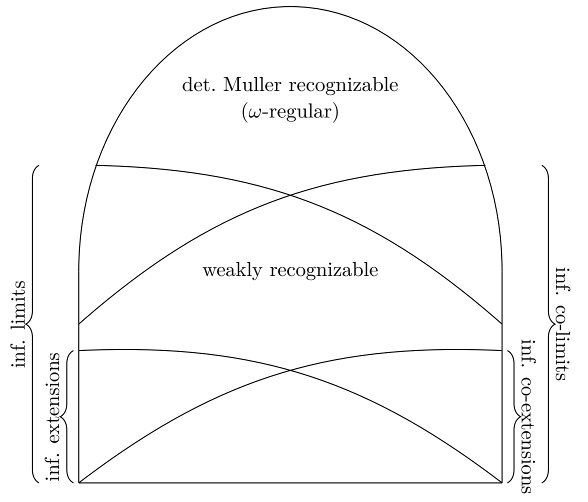


Figure 1.2: A classification of recognizable ω-languages in terms of logic and automata.

Theorem 1.13 (see [PP04]) *A language $L \subseteq \Sigma^\omega$ can be expressed in the form $L = \mathsf{ext}(K), K \subseteq \Sigma^*$ recognizable if and only if for any DMA $\mathfrak{M}$ recognizing L, the acceptance component of $\mathfrak{M}$ is closed under reachable sets.*

At this point, it is important to note that each of these language classes, viz. recognizable ω-languages, $\mathsf{lim}(\mathsf{REG})$, $\mathsf{co\text{-}lim}(\mathsf{REG}) := \{L \subseteq \Sigma^\omega \mid \exists L' \in \mathsf{lim}(\mathsf{REG})\colon L = \overline{L'}\}$, $\mathsf{ext}(\mathsf{REG})$, and $\mathsf{co\text{-}ext}(\mathsf{REG}) := \{L \subseteq \Sigma^\omega \mid \exists L' \in \mathsf{ext}(\mathsf{REG})\colon L = \overline{L'}\}$, can also be characterized in terms of appropriate fragments of monadic second order logic. We do not study these logics in more detail here but do mention the important consequence, the Borel hierarchy of recognizable ω-languages, which this logical characterization yields. Figure 1.2 exhibits a graphical representation of this hierarchy, and captures the recurring theme of this thesis.

The top-most level of this hierarchy for recognizable ω-languages is occupied by the class of all recognizable ω-languages. Every language in this class can be expressed as a finite combination of lim and $\mathsf{co\text{-}lim}$ languages, which occupy the next lower (incomparable) classes. In the intersection of these language classes lies the family of weakly recognizable languages, which itself can be constructed in terms of finite combinations of ext and $\mathsf{co\text{-}ext}$ languages.

Chapter 2

Trace Languages and Recognizability

In this chapter we look at the main results that attempt to generalize the study of words to that of Mazurkiewicz traces. We present the classical result due to Zielonka, which closely relates languages of traces to languages of words, and then move on to ω-regular trace languages and the corresponding automata models. We then highlight the research opportunities offered by the current set of results in this field. Once again, we only concentrate on languages and automata and refer the reader to [Tho90b, EM93] for connections between logic and trace languages.

2.1 Traces, Operators, and Languages

Over a finite alphabet Σ, let $I \subseteq \Sigma^2$ be a symmetric, irreflexive[1] *independence relation*. We also refer to the corresponding reflexive, symmetric *dependence relation* $D \coloneqq \Sigma^2 \setminus I$, and we denote the pair (Σ, I) as the *independence alphabet*. Given 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 events $e, e' \in V \colon \lambda(e) D \lambda(e') \Leftrightarrow (e \lessdot e'$ or $e' \lessdot e$ or $e = e')$. We denote the set of all finite traces over an alphabet (Σ, I) with $\mathbb{M}(\Sigma, I)$.

For convenience, we work with "simplified" traces $t = [V, \lessdot, \lambda]$ where we remove all edges that may be inferred from others, i.e. by $\lessdot$ we mean $\lessdot \setminus \lessdot^2$. We also refer to the partial order $<$ obtained from the transitive closure of this simplified edge relation; and define relations $\leq$, $\gtrdot$, $\geq$, and $>$ in the natural manner. We use the abbreviation $e \in t$ to convey $t = [V, \lessdot, \lambda]$ and $e \in V$, and if $a \in \Sigma$ then we also use the abbreviation $a \in t$ to convey that there exists $e \in t$ such that $\lambda(e) = a$. This simplified notation is shown in the example trace of Figure 2.1. Although a and b are mutually independent letters of the alphabet, $e_1 < e_4$ since they are both ordered with respect to e_2, i.e. $e_1 \lessdot e_2 \lessdot e_4$.

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.

[1]A relation $R \subseteq X \times X$ is called *irreflexive* only if for each $x \in X, (x, x) \notin R$.

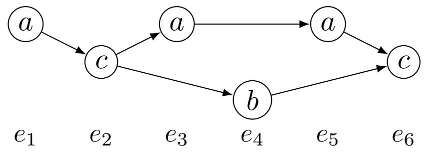


Figure 2.1: A finite trace $t = [V, \lessdot, \lambda]$ with simplified edge relation $\lessdot$.

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 $\lessdot' = \lessdot_1 \uplus \lessdot_2 \uplus \{(e_1, e_2) \in V_1 \times V_2 \mid \lambda_1(e_1) D \lambda_2(e_2)\}$. The *concatenation* $t \odot \theta$, with $t \in \mathbb{M}(\Sigma, I)$ and $\theta \in \mathbb{R}(\Sigma, I)$, is defined similarly. Unlike the case of words, the *concatenation* $\theta_1 \odot \theta_2$ of two infinite traces is defined, but only if $\mathsf{alphinf}(\theta_1) I \mathsf{alph}(\theta_2)$.

The concatenation operation helps us define the notion of trace prefixes. We say that t_1 is a *prefix* of t_2, denoted $t_1 \sqsubseteq t_2$ or $t_2 \sqsupseteq t_1$, if there exists $t' \in \mathbb{M}(\Sigma, I)$ such that $t_2 = t_1 \odot t'$. A prefix t_1 is a *strict prefix* of t_2, denoted $t_1 \sqsubset t_2$, if $t_1 \sqsubseteq t_2$ and $t_1 \neq t_2$. Similarly, we can define finite prefixes t if infinite traces $\theta \in \mathbb{R}(\Sigma, I)$. If $E \subseteq t$ is a set of events, then $t[E] = [V', \lessdot', \lambda']$ is a prefix of t with the set $V' := \{f \in t \mid f \leq e \text{ for some } e \in E\}$ and $\lessdot'$ and λ' are obtained by restricting the corresponding entities in t to V'. For finite sets $E \subsetneq \theta$, we define $\theta[E]$ in exactly the same manner.

As mentioned previously, traces are generalizations of words. In order to formalize this notion, the *trace morphism* $\Gamma \colon \Sigma^* \to \mathbb{M}(\Sigma, I)$ is defined inductively as follows:

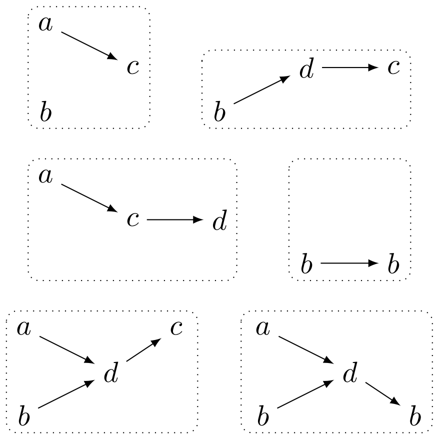
- For each $a \in \Sigma$, define $\Gamma(a) = [V, \lessdot, \lambda]$ as a trace comprising of a singleton set V, the empty edge relation $\lessdot$, and the mapping λ which maps the only element in V to a.
- For $u \in \Sigma^+, a \in \Sigma$, define $\Gamma(ua) := \Gamma(u) \odot \Gamma(a)$.

Similarly, we also refer to the inverse morphism $\Gamma^{-1} \colon \mathbb{M}(\Sigma, I) \to 2^{\Sigma^*}$ as the *linearization* of traces. This inverse morphism yields an equivalence class of words that “correspond” to the same trace. Consequently, we say that two words $u, v \in \Sigma^*$ are *trace-equivalent*, denoted $u \sim_I v$, if $\Gamma(u) = \Gamma(v)$. For a language $K \subseteq \Sigma^*$, define $\Gamma(K) = \{\Gamma(u) \in \mathbb{M}(\Sigma, I) \mid u \in K\}$. We note that for finite traces, the relation $\sim_I$ coincides with the reflexive, transitive closure of the relation $\{(uabv, ubav) \mid u, v \in \Sigma^* \wedge aIb\}$. For a word w, define the set $[w]_{\sim_I} := \Gamma^{-1}(\Gamma(w))$. Finally, we say that a word language K is *trace-closed* iff $K = [K]_{\sim_I}$, where $[K]_{\sim_I} := \bigcup_{u \in K} [u]_{\sim_I}$.

Example 2.1 *Let* (Σ, I) *be an independence alphabet with* $\Sigma = \{a, b, c\}$ *and the independence relation*[2] $I = \{(a, b)\}$. *For the trace* t *shown in Figure 2.1:*

1. $t = \Gamma(v)$, *where* $v = acaabc$; *and*

[2]Since I is symmetric, it suffices to define I with the help of only the representative pairs of independent letters.


(a) Three pairs t_1, t_2 of traces.

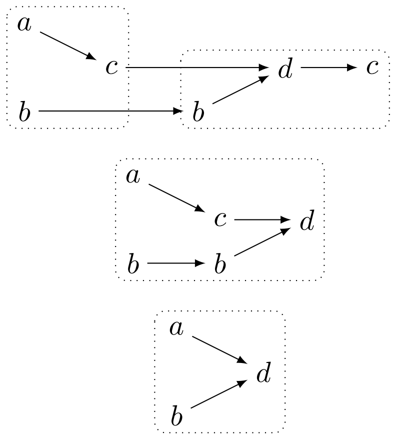

(b) Operations $t_1 \odot t_2$, $t_1 \sqcup t_2$, and $t_1 \sqcap t_2$.

Figure 2.2: $\Sigma = \{a, b, c, d\}$, with aIb, cIb.

2. $\Gamma^{-1}(t) = \{acaabc, acabac, acbaac\}$.

The trace morphism can be naturally extended to infinite structures, namely the morphism $\Gamma\colon \Sigma^\omega \to \mathbb{R}(\Sigma, I)$. And similarly, we obtain the notion of linearizations of infinite traces $\Gamma^{-1}\colon \mathbb{R}(\Sigma, I) \to 2^{\Sigma^\omega}$, and the notion of trace equivalence of infinite words where two infinite words α and β are trace equivalent, also denoted $\alpha \sim_I \beta$, if $\Gamma(\alpha) = \Gamma(\beta)$. For $L \subseteq \Sigma^\omega$, we define the ω-trace language $\Gamma(L)$ and trace-closed ω-language $[L]_{\sim_I}$ in the natural manner.

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 $s \sqsubseteq t_1 \wedge s \sqsubseteq t_2$. Figure 2.2 considers three pairs of traces, t_1 and t_2, and considers the three operations on these.

Over an independence alphabet (Σ, I), a set $T \subseteq \mathbb{M}(\Sigma, I)$ of finite traces is called a *trace language*, and a set $\Theta \subseteq \mathbb{R}(\Sigma, I)$ of infinite traces is called an *ω-trace language*. Similar to the case of word languages, the concatenation operator can be extended to languages of traces in natural ways, viz. $T_1 \odot T_2$ and $T \odot \Theta$. Unlike word languages however, we can also define the concatenation $\Theta_1 \odot \Theta_2$ of ω-trace languages, but[3] only if $\mathsf{alphinf}(\Theta_1) I \mathsf{alph}(\Theta_2)$.

For trace languages $T \subseteq \mathbb{M}(\Sigma, I)$ and $\Theta \subseteq \mathbb{R}(\Sigma, I)$, we also refer to the corresponding *trace-closed languages* $K = \Gamma^{-1}(T)$ and $L = \Gamma^{-1}(\Theta)$ respectively.

Similar to the case of language operators for word languages (cf. Section1.1), we have the usual operators for trace languages, viz. Kleene star, Kleene plus, bounded iteration, ω-iteration, union, complement, intersection, and difference. To avoid repetition, we do not present these definitions, but illustrate them with the help of an example.

[3]We overload the definitions and define $\mathsf{alph}(\Theta) := \bigcup_{\theta \in \Theta} \mathsf{alph}(\theta)$; similarly $\mathsf{alphinf}(\Theta)$.

Example 2.2 *Consider a simple alphabet* (Σ, I)*, with* $\Sigma = \{a, b\}$ *and* $I = \{(a, b)\}$*.*

1. $T_1 = \{\Gamma(ab)\}$*, is a language containing a single trace which comprises of two independent letters* a *and* b*.*
2. $K_1 = \Gamma^{-1}(T_1) = \{ab, ba\}$*.*
3. $T_2 = T_1^*$ *is a language containing all traces* $t \in \mathbb{M}(\Sigma, I)$ *with* $c \notin t$ *and an equal number of occurrences of* a *and* b *in* t*.*
4. $K_3 = [\{aa\}^+\{bb\}^+]_{\sim_I}$ *is a trace-closed language containing an even (non-zero) number of occurrences of both* a *and* b*.*
5. $T_3 = \Gamma(K_3)$ *is the trace language corresponding to* K_3*.*

In Section 1.1 and subsequently in Section 1.2 we mention the equivalence between languages that can be expressed with the help of finitely many operators and languages that are recognized by deterministic finite automata, or by deterministic Muller automata in case of ω-languages. In the case of traces however, this equivalence does not hold immediately.

In particular, observe that in Example 2.2 above, although the language T_2 can be defined in terms of finitely many operators, it consists of traces with equal number of occurrences of a and b. This implies that the language $\Gamma^{-1}(T_2)$ is not a recognizable word language (cf. language K_1 in Example 1.1), although T_2 can be expressed in terms of finitely many operators. We formalize these ideas in the next sections.

2.2 Trace Languages and Asynchronous Automata

In order to generalize the word model to distributed computations, we have a model of "distributed," finite state automata. Such automata consist of a number of "processes" which capture the independent nature of letters by ensuring concurrent transitions across multiple processes. Since there is no restriction on the number of events that may be concurrently executed, this automaton model is also referred to as *asynchronous automata* [Zie87].

A deterministic asynchronous automaton is 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 $\mathsf{dom}(\lambda(e))$; and to $\mathsf{dom}(E)$ 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 $P \subseteq \mathcal{P}$, the set X_P of *P-states* is a defined as $X_P := \{(x_p)_{p\in\mathcal{P}} \mid x_i \in X_{p_i}$ if $p_i \in$

P, 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*, or a *partial 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 $\pi = (x_p)_{p\in\mathcal{P}}$, define the *$q$-state in* π as $\pi_{|q} \coloneqq x_q \in X_q \cup \{\$\}$, 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 the states of the transitions system as $X_{2^\mathcal{P}} \coloneqq \bigcup_{P\subseteq\mathcal{P}} X_P$.

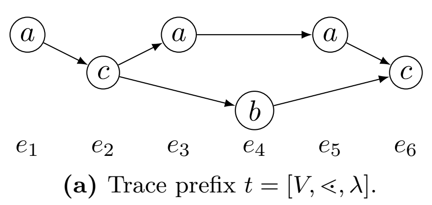
Over a fixed independence alphabet (Σ, I), a set $\mathcal{P}$ of processes, and a mapping dom, we now define a *deterministic asynchronous transition system* (an *ATS*) as a tuple $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$, where X_p are sets of local p-states; transition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

Over a trace $t = [V, \lessdot, \lambda] \in \mathbb{M}(\Sigma, I)$, we define *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. The labeling λ' is defined similarly except $\lambda'(e_\perp) \coloneqq \varepsilon$. Analogously, we define the *infinite run* $\rho = [V', \lessdot', \lambda', \Lambda]$ over an infinite trace $\theta = [V, \lessdot, \lambda] \in \mathbb{R}(\Sigma, I)$.

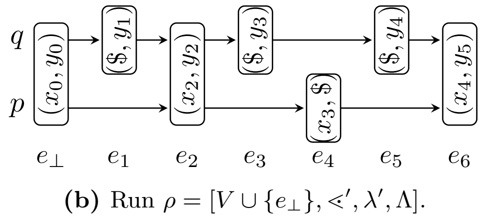
In order to help us formally define the labeling Λ, we make the following observations. During the run ρ of an ATS $\mathfrak{T}$ over a trace, each process p makes state transitions on events $e \in \mathsf{dom}^{-1}(p)$. Each such event may be called a *p-event* as well as a *P-event* where $P = \mathsf{dom}(e)$. 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 and the mapping dom. 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 $f \lessdot'_p e$. Now, $\Lambda\colon V' \to X_{2^\mathcal{P}}$ is defined inductively as:

- $\Lambda(e_\perp) \coloneqq (\pi_0)$,
- for any $e > e_\perp$, if
  1. $a = \lambda(e)$, and
  2. 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.



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



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

Figure 2.3: 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 $\mathsf{dom}(c) = \{p, q\}$.

Figure 2.3a shows the labeled events of a trace t and Figure 2.3b shows the corresponding run ρ; but λ' is omitted in ρ for readability. The processes are assumed to be ordered, hence the representation of states as tuples. Note that, in Figure 2.3b, 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 $e_\perp \lessdot' e_2$. Moreover, while $e_2 \lessdot'_p e_3$, e_5 does not have any p-predecessor; i.e., there does not exist any $e \in \rho$ such that $e \lessdot'_p e_5$.

Analogous to trace prefixes, we refer to run prefixes, and to prefixes $\rho[e], \rho[E]$ for $e \in \rho$ and $E \subseteq \rho$ respectively. For a finite run ρ, and a process p, we refer to the *causal view of* p in ρ, respectively in ρ, as the prefix $\rho[\max_p(\rho)]$ comprising of all the events of ρ that are below the maximum p-event in ρ. We can generalize this for sets $P \subseteq \mathcal{P}$ to define *collective causal view of* P in ρ as $\bigsqcup_{p \in P} \rho[\max_p(\rho)]$. Clearly, $\bigsqcup_{p \in \mathcal{P}} \rho[max_p(\rho)] = \rho$. With a slight abuse of notation, in the context of an SATS, we also define the *causal view of* p in a finite trace t as $t[\max_p(t)]$; and hence the *collective causal view of* P in t.

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

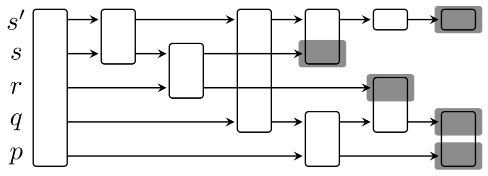
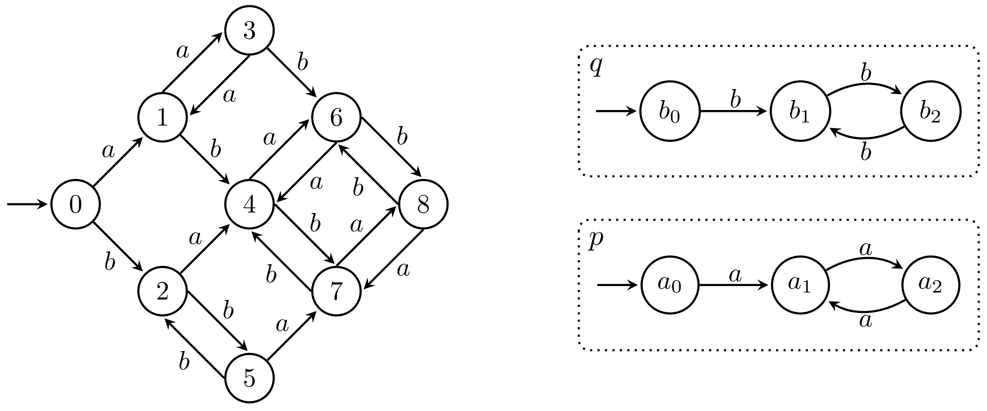


Figure 2.4: Acceptance by referring to the final local states acquired by the DFAA.

Although the acceptance component $\mathcal{F}$ of a DFAA consists of global states, it must be noted that this is just a convenient abstraction to collectively refer to all the local p-states that the DFAA acquires at the end of a finite run. As shown in Figure 2.4, the acceptance decision is therefore made in a “distributed” manner by referring to individual final states of the processes.

Definition 2.3 *A language* $T \subseteq \mathbb{M}(\Sigma, I)$ *is called* recognizable trace language *if there exists a DFAA* $\mathfrak{A}$ *such that* $L(\mathfrak{A}) = T$.

As we mentioned previously, trace languages are closely related to languages of words. A deep theorem by Wiesław Zielonka establishes the connection between recognizable trace languages T and their linearizations $\Gamma^{-1}(T)$ (cf. Section 2.1



(a) I-diamond DTS $\mathfrak{T}$ for languages K_3 and L_3. **(b)** ATS $\mathfrak{T}'$ for languages T_3 and Θ_3.

Figure 2.5: Transition systems for languages in Example 2.5 and Example 2.9.

for a definition of Γ). Referring to the independence relation I over the alphabet, a DFA $\mathfrak{A} = (\mathfrak{T}, F)$ with $\mathfrak{T} = (Q, \Sigma, \delta, q_0)$, is called an *$I$-diamond DFA* if for each pair $a, b \in \Sigma$ of independent letters and each state $q \in Q$, it holds that $\delta(q, ab) = \delta(q, ba)$. Zielonka provided the following seminal result [Zie87].

Theorem 2.4 (see also [DR95]) *Given any $T \subseteq \mathbb{M}(\Sigma, I)$ and its linearization $K = \Gamma^{-1}(T) \subseteq \Sigma^*$, T is recognizable if and only if K is recognizable. Moreover, if K is recognizable then the minimal DFA recognizing it is I-diamond closed.*

Example 2.5 *Consider the languages K_3 and T_3 mentioned in Example 2.2.*

With the DTS $\mathfrak{T}$ shown in Figure 2.5a and $F = \{8\}$, we obtain the minimal DFA $\mathfrak{A} = (\mathfrak{T}, F)$ recognizing K_3.

For the trace language T_3, the DFAA $\mathfrak{A}' = (\mathfrak{T}', \mathcal{F}')$ can be constructed where ATS $\mathfrak{T}'$ is shown in Figure 2.5b, with processes p and q, $\mathsf{dom}(a) = \{p\}$ and $\mathsf{dom}(b) = \{q\}$, and $\mathcal{F}' = \{(a_2, b_2)\}$.

The definition of recognizable languages of infinite traces was first provided by Gastin-Petit [GP92a] in terms of recognition by partially commutative monoids. However, we use as definition the characterization of the same family of languages in terms of deterministic (cellular) Muller automata [DM94, Mus94]. Although this definition was in terms of asynchronous cellular transition systems, they are equivalent to the ATS's that we have defined [DR95, Chapter 7].

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 $\mathcal{F} = \{F_1, F_2, \dots F_n\}$ 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 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, $F_i^p = \mathsf{Inf}_p(\rho)$ [DM94].

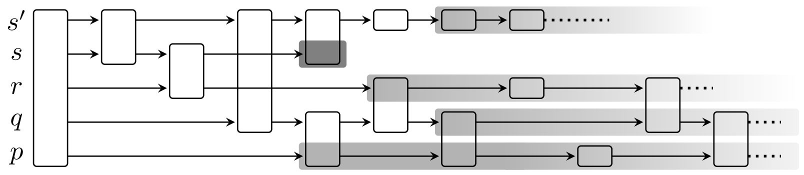


Figure 2.6: Acceptance by referring to the local infinity sets of the DAMA.

Figure 2.6 contrasts the finitary acceptance condition for DFAA. For a DAMA, the local infinitary sets are collectively compared against the tuples in the acceptance component $\mathcal{F}$.

Definition 2.6 *A language $\Theta \subseteq \mathbb{R}(\Sigma, I)$ is said to be an* recognizable ω-trace language *if it is recognized by a deterministic asynchronous Muller automaton.*

Similar to the statement of Theorem 2.4, in the case of ω-trace languages as well, we have a correspondence between recognizable trace languages Θ and word languages $\Gamma^{-1}(\Theta)$, which are recognized by I-diamond Muller automata with an additional restriction.

Definition 2.7 ([Mus94]) *Let $\mathfrak{M} = (\mathfrak{T}, \mathcal{F})$ be a deterministic Muller automaton (over words) with $\mathfrak{T} = (Q, \Sigma, \delta, q_0)$.*

1. *For a state $x \in Q$ and a word $u \in \Sigma^*$, if $\delta(x, u) = y \in Q$, then denote the set of states visited during the run from x to y, incl. x, y, as $\mathsf{Occ}(x \xrightarrow{u} y)$.*
2. *The acceptance table $\mathcal{F}$ is called* closed, *if for each $F \in \mathcal{F}$, each $x \in F$ and each $v \in \Sigma^*$ with $\delta(x, v) = x$ and $\mathsf{Occ}(x \xrightarrow{v} x) = F$, it holds that*

$$\forall w \in \Sigma^* \colon v \sim_I w \Rightarrow \mathsf{Occ}(x \xrightarrow{w} x) \in \mathcal{F}.$$

Theorem 2.8 (Muscholl [Mus94]) *Let $\Theta \subseteq \mathbb{R}(\Sigma, I)$ be an ω-trace language and $L = \Gamma^{-1}(\Theta)$ the corresponding trace-closed word language. Θ is recognized by a DAMA if and only if L is recognized by an I-diamond DMA whose acceptance table is closed as per Definition 2.7.*

Intuitively, the closure property of Definition 2.7 generalizes the I-diamond structure to arbitrarily long equivalent words $u, v \in \Sigma^*$ – the I-diamond structure refers only to words of length 2. The above definition goes further and includes all the states visited, since the final states at the end of u and v are identical owing to the I-diamond property.

Example 2.9 *Recall the language K_3 described in Example 2.2. The infinitary language $L_3 = [\mathsf{lim}(K_3)]_{\sim_I}$ consists of all words $u \in \Sigma^\omega$ where either (1) the number $|u|_a$ of a's appearing in u is even (non-zero) and the number $|u|_b$ of b's appearing in u is infinite; or (2) or the number $|u|_a$ of a's appearing in u is infinite and the number $|u|_b$ of b's appearing in u is even (non-zero); or (3) both $|u|_a = |u|_b = \infty$.*

Referring to the DTS in Figure 2.5a, we can define an I-diamond Muller automaton $\mathfrak{M} = (\mathfrak{T}, \mathcal{F})$ with $\mathcal{F} = \{6,8\}, \{7,8\}, \{4,6,7\}, \{4,6,8\}, \{4,7,8\}, \{6,7,8\}, \{4,6,7,8\}\}$. One can verify that $\mathfrak{M}$ recognizes L_3 and the acceptance table $\mathcal{F}$ is closed as per Definition 2.7.

To describe a DAMA recognizing $\Theta_3 = \Gamma(L_3)$, we refer to the ATS $\mathfrak{T}'$ shown in Figure 2.5b. Define the DAMA $\mathfrak{M}' = (\mathfrak{T}', \mathcal{F}')$, where $\mathcal{F}'$ contains the following collections.

1. $F_1 = (\{a_2\}, \{b_1, b_2\})$
2. $F_2 = (\{a_1, a_2\}, \{b_2\})$
3. $F_3 = (\{a_1, a_2\}, \{b_2, b_2\})$.

We would like to emphasize, as we did at the end of of Section 1.2, that within the acceptance tables $\mathcal{F}$ of DAMAs, we only consider those collections $F \in \mathcal{F}$ that are *realizable.* That is, $F \in \mathcal{F}$ only if there exists an infinite run ρ of the DAMA such that for each $p \in \mathcal{P}$: $\mathsf{Inf}_p(\rho) = F^p$.

2.3 From Finitary to Infinitary Trace Languages

In Chapter 1, we recalled the close relationships between recognizable languages and recognizable ω-languages. We saw various operators that help us describe ω-languages and the straight-forward correspondences between the DFAs and the various classes of ω-automata. In particular, recall that the family of recognizable languages of infinite words is structured into a hierarchy where each level is characterized by a class of deterministic ω-automata – the class of deterministic Büchi automata being the most prominent among them.

In this section, we will mention the main results in the theory of ω-automata for ω-trace languages, and we will see that the corresponding classifications are still missing. In particular, the simple examples that we saw in the previous sections – the straight-forward connection between the DFAA of Example 2.5 and the DAMA of Example 2.9 – do not generalize for the case of trace languages.

We begin with the case where a result for recognizable ω-languages does generalize to ω-regular trace languages, viz. every recognizable language $L \subseteq \Sigma^\omega$ can be expressed as a finite union $\bigcup_{i=1}^{n} U_i \cdot V_i$ with $n \in \mathbb{N}$ and $U_i, V_i \subseteq \Sigma^*$ recognizable. However, before we present the analogous result for trace languages, there is an additional case of the so called *connected trace languages* to be considered.

Definition 2.10 *Given an independence alphabet (Σ, I), a set $X \subseteq \Sigma$ is called a* connected alphabet *if the undirected dependence graph (X, D) is connected.*

- *A trace $t \in \mathbb{M}(\Sigma, I)$ is called a* connected trace *if the set* $\mathsf{alph}(t)$ *is a connected alphabet.*

- *A language $T \subseteq \mathbb{M}(\Sigma, I)$ is called a* connected trace language *if every trace $t \in T$ is a connected trace.*

Example 2.11 *The languages T_3 and Θ_3 considered in Examples 2.5 and 2.9 are not connected languages since the dependence graph (Σ, D) obtained from the independence alphabet (Σ, I) has no edges.*

The trace t shown in Figure 2.3a is connected, since $\mathsf{alph}(t) = \{a, b, c\}$*, and aDc and bDc.*

Theorem 2.12 (Gastin & Petit [GP92a]) *An ω-trace language $\Theta \subseteq \mathbb{R}(\Sigma, I)$ is recognizable if and only if Θ can be expressed as a finite union $\bigcup_{i=1}^{n} S_i \odot T_{i,1}^{\omega} \odot \cdots \odot T_{i,m_i}^{\omega}$, where*

- *$n \in \mathbb{N}$, and for all $i, 1 \leq i \leq n$: $m_i \in \mathbb{N}$;*

- *languages $S_i, T_{i,j} \subseteq \mathbb{M}(\Sigma, I)$ are recognizable languages of finite traces; and*

- *for all $1 \leq i \leq n$: languages $T_{i,j}$ are connected languages and* $\mathsf{alph}(T_{i,j_1}) \times \mathsf{alph}(T_{i,j_2}) \subseteq I$ *whenever $j_1 \neq j_2$.*

However, once again we seek more intuitive mechanisms of describing ω-regular trace languages with the help of recognizable trace languages. We see that these definitions generalize the case of word languages.

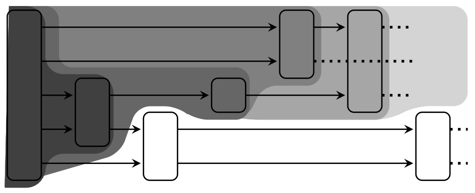
Definition 2.13 *Given a language $T \subseteq \mathbb{M}(\Sigma, I)$, the* infinitary extension of T *is the ω-language* $\mathsf{ext}(T) := T \odot \mathbb{R}(\Sigma, I)$.

Definition 2.14 ([DM94]) *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 $\bigsqcup_{i \in \mathbb{N}} t_i = \theta$.*

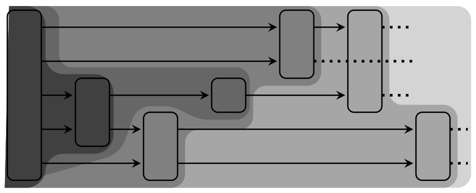
Figure 2.7 illustrates the definition of $\mathsf{lim}(T)$ with the help of a run of an asynchronous automaton recognizing T. Shaded prefixes end in global accepting states. Figure 2.7a illustrates an induced run if the trace $\theta \notin \mathsf{lim}(T)$, whereas Figure 2.7b illustrates the contrary.

A *deterministic asynchronous Büchi automaton*[GP92a, DM94] (a *DABA*) is a pair $\mathfrak{A} = (\mathfrak{T}, \mathcal{F})$, where $\mathcal{F} = \{F_1, F_2, \dots F_n\}$ is a table with $F_i = (F_i^p)_{p \in \mathcal{P}}$. 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, $F_i^p \subseteq \mathsf{Inf}_p(\rho)$.

Note that the Büchi acceptance condition defined above does not correspond to the classical condition that checks for non-empty intersection of the infinity sets with the acceptance sets. For a justification for the above-mentioned condition we direct the reader to Gastin & Petit's original definition of asynchronous cellular (non-deterministic) Büchi automata [GP92a].



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



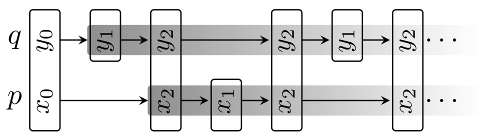
(b) $\theta \in \mathsf{lim}(T)$ if each event is eventually covered by one of the accepting runs over the sequence of strict prefixes.

Figure 2.7: Shaded regions constitute a sequence of accepting runs of an automaton recognizing T.

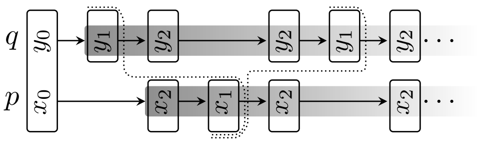
In the theory of ω-regular word languages, the family of deterministically Büchi recognizable languages can be characterized in terms of infinitary limit languages $\mathsf{lim}(K)$ for $K \subseteq \Sigma^*$ recognizable. However, with the current definitions, it is still open whether the family of deterministic asynchronous Büchi automata characterize any class of recognizable ω-trace languages. In fact, there exist deterministic trace languages that are not accepted by any DABA [Mus94]. In particular, a DFAA $\mathfrak{A} = (\mathfrak{T}, F)$ recognizing a language T does not directly yield a DABA $\mathfrak{B} = (\mathfrak{T}, \mathcal{F})$ recognizing $\mathsf{lim}(T)$. We consider a simple illustration of the reason behind this fact.

Example 2.15 *Consider a DFAA $\mathfrak{A} = (\mathfrak{T}, F)$ comprising of two processes, where $F = \{(x_1, y_1)\}$ is a singleton. Let $L(\mathfrak{A}) = T$. Suppose, analogously to the case of word automata, that we define a DABA $\mathfrak{A}' = (\mathfrak{T}, \mathcal{F})$ where $\mathcal{F} = \{(\{x_1\}, \{y_1\})\}$ contains a single collection of singleton local acceptance sets.*

The DABA $\mathfrak{A}'$ accepts a trace $\theta \in \mathbb{R}(\Sigma, I)$ if for the run ρ of θ: $\{x_1\} \subseteq \mathsf{Inf}_p(\rho)$ and $\{y_1\} \subseteq \mathsf{Inf}_q(\rho)$. Now consider two such infinite runs ρ_1 and ρ_2 as shown in Figure 2.8.



(a) Infinite run ρ_1: for no prefix $\rho \sqsubset \rho_1$, $\Lambda(\rho) = (x_1, y_1)$.



(b) Infinite run ρ_2: for prefixes ρ demarcated by dotted lines, $\Lambda(\rho) = (x_1, y_1)$.

Figure 2.8: Both the runs induce the same local infinity sets, $\mathsf{Inf}_p(\rho_1) = \mathsf{Inf}_p(\rho_2) = \{x_1, x_2\}$, and $\mathsf{Inf}_q(\rho_1) = \mathsf{Inf}_q(\rho_2) = \{y_1, y_2\}$.

By referring only to the local infinity sets, the DABA $\mathfrak{A}'$ cannot distinguish between the runs. Although the same local states appear infinitely often, only in run ρ_2 does the global state (x_1, y_1) appear at all.

On the other hand, it is 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 recognizable languages $T \subseteq \mathbb{M}(\Sigma, I)$ such that $\mathsf{lim}(T)$ is not recognized by any $(\mathfrak{T}, \mathcal{F})$ (see e.g. [Mus94]).

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 2.16 ([Mus94]) *For $T \subseteq \mathbb{M}(\Sigma, I)$ and some $A \subseteq \Sigma$, the* A-infinitary limit of T *is defined*[4] *as* $\mathsf{lim}_A(T) := \{\theta \in \mathsf{lim}(T) \mid D(\mathsf{alphinf}(\theta)) = D(A)\}$.

Definition 2.17 ([Mus94]) *An ω-regular trace language is called a* deterministic trace language *if it can be expressed as a finite union $\bigcup_{i=1}^{n} \mathsf{lim}_{A_i}(T_i)$ of A-infinitary limits of recognizable trace languages T_i.*

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 component $\mathcal{F}$ 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 $(F, A) \in \mathcal{F}$ where for each $p \in \mathcal{P}, F^p \subseteq \mathsf{Inf}_p(\rho)$, and $A \subseteq \mathsf{alphinf}(\theta)$.

Although these variants of automata fail to yield a characterization of deterministic trace languages or of infinitary limit trace languages, the definition of the class of deterministic trace languages is well motivated.

Theorem 2.18 (Diekert & Muscholl [DM94]) *A language $\Theta \subseteq \mathbb{R}(\Sigma, I)$ is recognizable, that is it is recognized by a DAMA, if and only if Θ can be expressed as a finite Boolean combination of deterministic trace languages.*

The definition of deterministic trace languages generalizes that of infinitary limits of word languages, i.e. DBA recognizable languages, since in the case of words the dependence graph over the alphabet is complete and therefore it is always the case that $D(\mathsf{alphinf}(\alpha)) = \Sigma$. Finally, we observe one more property that is also a hallmark of DBA recognizable languages.

Remark 2.19 ([DM94]) *The class of deterministic trace languages is closed under finite unions and intersections.*

2.4 Unanswered Questions

As we have stressed previously, the study of languages of Mazurkiewicz traces aims to generalize the study of languages of words. However, it has so far not been

[4] Referring to the dependence relation $D \subseteq \Sigma^2$, for $A \subseteq \Sigma$, define $D(A) := \{a \in \Sigma \mid \exists b \in A \colon bDa\}$.

possible to generalize a number of well known results from the theory of ω-regular word languages. We believe that a well rounded theory of traces necessitates that these results extend to ω-regular traces languages as well.

In particular, 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 $\mathsf{lim}(K) := \{\alpha \in \Sigma^\omega \mid \alpha \text{ has infinitely many prefixes in } K\}$ for some recognizable language $K \subseteq \Sigma^*$ (see Theorem 1.7). In Definition 2.14, Diekert-Muscholl generalize the definition of the operator lim for the case of trace languages. Muscholl also provides a definition of the A-infinitary limit lim_A, where the parameter A reveals information about the set of letters that are allowed to appear infinitely often (see Definition 2.16). Every infinitary limit language can be expressed as a finite union of A-infinitary limit languages, whereas the reverse in not necessarily true (cf. Theorem 2.18 and [Mus94]). 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? Is the class of deterministic trace languages a decidable subclass of recognizable ω-trace languages?*

We suspect that these questions remain open owing to the current definitions of models of asynchronous (cellular) automata. As illustrated in Example 2.15, 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 account for this partition while accepting or rejecting a run.

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 [Mus94]. This concern is redundant for the case of word automata since there is only one "process".

Exploiting this intuition, we present a new class of asynchronous automata that enables us to answer the above-mentioned question in the affirmative.

Another aspect of generalization of language theory may apply to the investigation of a hierarchy of ω-trace languages vis-á-vis the Borel hierarchy of recognizable

ω-languages as illustrated in Figure 1.2. In this direction, we provide a solution in the framework of ω-automata over infinite words – which is invoked via the sets of linearizations of infinitary trace languages. We identify trace languages whose linearizations are recognized by deterministic I-diamond weak or deterministic I-diamond Büchi (word) automata. We present a characterization of the class of linearizations of all ω-regular trace languages in terms of I-diamond Muller (word) automata. Finally, we show that the language classes characterized by these automata arrange themselves into a Borel-like hierarchy.

Chapter 3

Synchronization Aware Automata

In the theory of ω-regular word languages, a natural classification is induced by various forms of deterministic ω-automata. The three fundamental cases are given by (a) deterministic Muller automata, capturing the class of ω-regular word languages; (b) deterministic Büchi automata, capturing infinite recurrences of finitary properties; and (c) weak automata, capturing reachability of finitary properties. In this chapter, we concentrate on the first two automata models, on which fundamental facts can be summarized as follows[1] (see e.g. [PP04]):

1. A language is deterministically Büchi recognizable if and only if it can be expressed as $\mathsf{lim}(K) := \{\alpha \in \Sigma^\omega \mid \alpha \text{ has infinitely many prefixes in } K\}$ for some regular language $K \subseteq \Sigma^*$.

2. An ω-regular language is deterministically Büchi recognizable if and only if this language is recognized by a Muller automaton whose acceptance component is closed under supersets.

3. The class of Boolean combinations of deterministically Büchi recognizable languages coincides with the class of Muller recognizable languages.

We consider the problem of defining automata and language classes in the framework of Mazurkiewicz traces [Maz87] – modeling infinite, concurrent behaviors of a finite set of interacting processes – vis-à-vis the above mentioned facts.

The concept of recognizable ω-trace languages has been studied in close correspondence to the case of recognizable ω-languages, for example[2], in terms of finite partially-commutative monoids, asynchronous automata, concurrent regular expressions, or monadic second order logic. However, it is remarkable that there does not yet exist a definition of Büchi automaton over traces that allows for results analogous to any of the items 1–3 above. Our objective is to fill this gap, while making sure at all times that the corresponding definitions and results for word languages emerge from our study as special cases.

Muscholl [Mus94] took a major step toward establishing such structural results by introducing a parameterized lim operator for trace languages (see Definition 2.16).

[1] Recall the assumption from Section 1.2 that any Muller acceptance component contains only realizable state sets.

[2] We refer the reader to [DR95] for a comprehensive survey of early results.

She showed that the class of Boolean combinations of parameterized lim-languages is precisely the class of ω-regular trace languages [DM94, Mus94], and also characterized the class of linearizations of these parameterized languages in terms of "I-diamond" Büchi (word) automata with "extended" acceptance condition. However, I-diamond word automata do not offer a proper modeling of concurrency as realized over traces.

We introduce a new concept of asynchronous automata, viz. *synchronization aware automata* (over traces rather than their linearizations). These, when equipped with Büchi and Muller acceptance conditions, establish not only item 1, but also items 2 and 3 above. At the same time, the synchronization aware Muller automata are equivalent in expressive power to the standard deterministic asynchronous Muller automata for infinitary trace languages. Thus we provide a new framework that prepares – at least in important parts – a structure theory for ω-regular trace languages that is compatible with that of deterministic ω-automata over words.

Synchronization aware automata are "aware" of the fact that during a run over an infinite trace, the set of processes may be partitioned in a manner that after a finite prefix, a process belonging to one part repeatedly interacts only with processes in the same part and never with a process outside of this part. The processes infer this partition by observing their infinitely recurring interactions. Although infinite traces induce such partitions in all asynchronous automata, current models cannot perform such inferencing. At the same time, synchronization aware automata generalize the definition of ω-automata over words since word automata consist of only one process.

Another aspect of infinite runs is that while some processes may remain *live* ad infinitum, others may halt after finitely many steps. However, the set of live processes can be explicitly coded in the Büchi acceptance condition since this directly corresponds to the parameters $A \subseteq \Sigma$ in Muscholl's parameterized lim operation mentioned above.

By combining both these aspects, we obtain the family of synchronization aware Büchi automata corresponding to item 1 above (see Theorem 3.27). We also introduce synchronization aware Muller automata recognizing precisely the class of ω-regular trace languages (see Theorem 3.32). Finally, Theorems 3.36 and 3.37 respectively demonstrate a characterization à la item 2 and the equivalence result of item 3. We conclude with a discussion of a number of open problems.

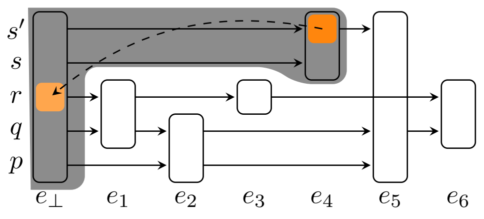
3.1 Secondaries and Frontiers

During a run ρ of an ATS, the processes can be thought of as "possessing and updating information" regarding other processes. 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

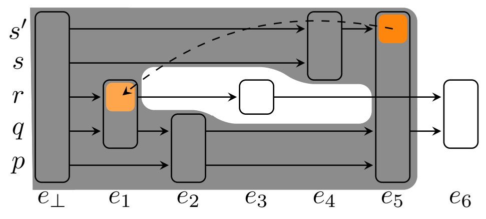
ρ, denoted by $\mathsf{latest}_{p\to q\to r}(\rho)$, is the maximal r-event in the prefix $\rho[\mathsf{latest}_{p\to q}(\rho)]$. Trivially, $\mathsf{latest}_{p\to q}(\rho) = \mathsf{latest}_{p\to p\to q}(\rho)$.

The *primary information* of p at ρ is defined as the partially ordered set $\mathsf{Pri}_p(\rho) := \{\mathsf{latest}_{p\to q}(\rho) \mid q \in \mathcal{P}\}$. The *secondary information* of p at ρ is given by the partially ordered set $\mathsf{Sec}_p(\rho) := \{\mathsf{latest}_{p\to q\to r}(\rho) \mid q, r \in \mathcal{P}\}$. 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 partial orders of these sets may both be defined in terms of 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 also $|\mathsf{Sec}_p(\rho)| \le |\mathcal{P}|^2$.

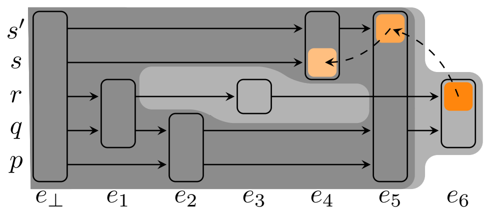
Throughout our discussions, we are mainly interested in secondary information of the form $\mathsf{Sec}_p(\rho[e])$ for $p \in \mathsf{dom}(e)$. Since, $\mathsf{Sec}_p(\rho[e]) = \mathsf{Sec}_q(\rho[e])$ for all $p, q \in \mathsf{dom}(e)$, for convenience we denote this information simply as $\mathsf{Sec}(e)$.



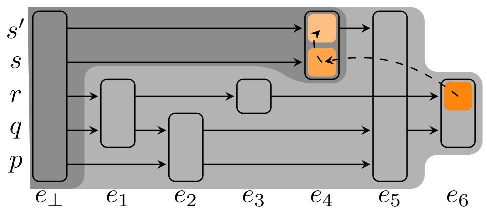
(a) $\mathsf{latest}_{s'\to r}(e_4)$.



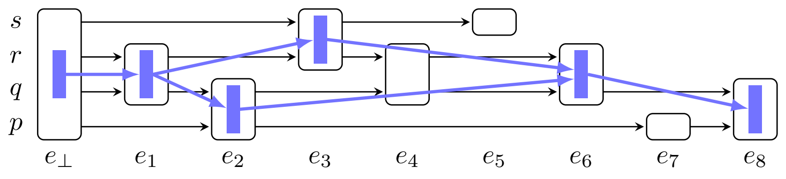
(b) $\mathsf{latest}_{s'\to r}(e_5)$.



(c) $\mathsf{latest}_{r\to s'\to s}(e_6)$.



(d) $\mathsf{latest}_{r\to s\to s'}(e_6)$.

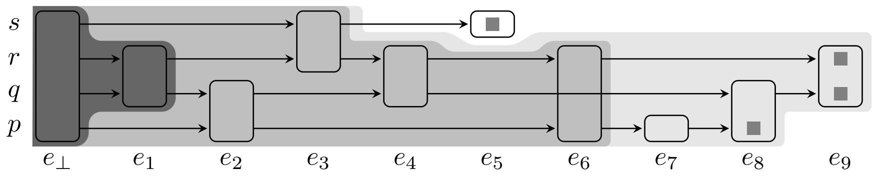


(e) Secondary information $\mathsf{Sec}(e_8)$ as a graph.

Figure 3.1: Illustration of first hand and second hand information.

Example 3.1 *Figure 3.1 illustrates these definitions.*

It also illustrates that first hand information may be updated as a result of indirect interaction. From Figure 3.1a and Figure 3.1b we see that the first hand information that process s' has about r changes in subsequent s'-events e_4 and e_5 although s' never synchronizes with r.



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 $e_5 \notin \rho[e_9]$.

Figure 3.2: Partial frontiers of a finite run prefix (see Section 3.1); illustration of Lemma 3.7 (see Exercise 3.8).

Furthermore, the information may be asymmetric. As shown in Figure 3.1c and Figure 3.1d, in the trace prefix $\rho[e_6]$*,* s' *is aware of the latest s-event, but s possesses "older" information regarding* s'*.*

Figure 3.1e illustrates secondary information that processes p, q *possess at the end of the run* $\rho[e_8]$*.*

There exists a distributed algorithm, implemented by way of the so-called *gossip algorithm* [Mad12], which enables processes to update their secondary graphs at the points of synchronization with the help of $|\mathcal{P}|^3$ system labels. 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 $\mathsf{dom}(e)$. We use this algorithm as a black-box for our constructions.

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 $H_\rho := \{e \in \rho \mid \exists p \in \mathcal{P}, e = \max_p(\rho)\}$. Any subset $H \subseteq H_\rho$ that is closed under the $\geq$ relation is called a *partial frontier*. We also refer to such a set as *upward closed*. E.g., the set $\{e_5, e_8\}$ in Figure 3.2 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 $\top_\rho(e) := \{f \in \rho \mid e \leq f \wedge \exists p \in \mathcal{P} \colon f = \max_p(\rho)\}$. Note that for any $e_1, \ldots, e_n \in \rho$, $\bigcup_{i=1}^n \top_\rho(e_i)$ 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 the (partial) state $\Lambda(H) := \Lambda(\rho)_{|\mathsf{dom}(H)}$. 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 finite run.

3.2 Synchronization Aware Transition Systems

Any infinite run ρ of an ATS $\mathfrak{T}$ over a trace $\theta \in \mathbb{R}(\Sigma, I)$ yields a partition $\Psi = (P_1, \ldots, P_n)$ 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 $p' \in P_j, i \neq j$. We informally refer to the parts P_i as "maximally interacting sets" induced by a run ρ over a θ. A process p may infer that it belongs to part P_i by observing that it infinitely often updates its first-hand or second-hand information w.r.t each $q \in P_i$. Our primary aim is to obtain a family of deterministic asynchronous transition systems where such infinitary inferencing can be performed by each process.

One way to enable the processes in computing their maximally interacting sets in the limit is to equip them with the ability to make relevant "local computations" at each event e that they participate in. Of course, each process $q \in \mathsf{dom}(e)$ contributes to the computation at e. The outcome of such a computation is a set P of processes, which indicates the "sum total of individual interactions" that are witnessed by the participants $q \in \mathsf{dom}(e)$ at e. Clearly, some events e can result in cumulative interaction P which is strictly more than cumulative interaction Q at other events f, i.e., $P \supsetneq Q$. In this manner, "maximal interactions" may be inferred by performing local computations. Now, if these local computations can be implicitly captured in the local states of the processes, then the task of inferring maximally interacting sets reduces to the task of observing which infinitely recurring local states correspond to "maximal local computations".

In order to enable this, 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 cumulative interaction witnessed by the processes in $\mathsf{dom}(e)$; and as explained above, sets $P \subseteq \mathcal{P}$ act as the unit of this measure. To define such a measure for events e of a run ρ, we first define the *secondary update* $\mathcal{U}_e$ comprising of the events in the secondary information that have been "reassigned" or "removed".

Definition 3.2 *For a run ρ of an ATS and an event $e \in \rho$, the* secondary update *at e is the set $\mathcal{U}_e := \{g \in \rho[e] \mid \exists p, q, r \in \mathcal{P}, \exists f_p \lessdot_p e : g = \mathsf{latest}_{p \to q \to r}(f_p) \neq \mathsf{latest}_{p \to q \to r}(e)\}$.*

For each displaced/updated secondary event $g \in \mathcal{U}_e$, we look in the future $\top_{\rho[e]}(g)$ of g and extract the processes that appear in this future. This set of processes is the contribution of g (and hence the contribution of the process $q \in \mathsf{dom}(e)$ from whose secondary information g comes) in the local computation at e. We result of this local computation, the "cumulative interaction" mentioned above, is referred to as the *degree of synchronization at e*.

Definition 3.3 *In a run ρ of an ATS, the* degree of synchronization *at an event $e \in \rho$ is defined as $\mathsf{ds}(e) := \bigcup_{g \in \mathcal{U}_e} \mathsf{dom}(\top_{\rho[e]}(g))$. By default, $\mathsf{ds}(e_\perp) := \mathcal{P}$.*

The subset relation over $\mathcal{P}$ provides a natural order for comparing various degrees of synchronizations. The following lemma demonstrates that Definition 3.3 is well behaved over infinite runs in the sense that it helps each process p in identifying the maximally interacting set P of processes for p in the run ρ. In turn, this means that $q \in P$ makes the same inference. For an infinite run ρ of an ATS, we denote with $\mathsf{alphinf}(\rho)$ the set $\mathsf{alphinf}(\theta)$ where θ is the trace inducing the run.

Lemma 3.4 *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 $P := \{q \in \mathcal{P} \mid \forall e \in \rho, \exists e' \in \rho : e < e' \wedge \mathsf{latest}_{p \to p \to q}(e) \neq \mathsf{latest}_{p \to p \to q}(e')\}$. 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 maximal 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 $\forall e \in \rho$: $e' < e \wedge p \in \mathsf{dom}(e) \Rightarrow \neg(P_1 \subsetneq \mathsf{ds}(e) \vee P_2 \subsetneq \mathsf{ds}(e))$, and consequently
4. there exists a process $r \in P_2 \setminus P_1$ but no p-event $e > e'$ with $P_1 \cup \{r\} \subseteq \mathsf{ds}(e)$.

For event e' as in item 3 and 4 above, 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 Definition 3.2. Since $r \notin \mathsf{ds}(e_1)$, it follows from Definition 3.3 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 $g'_{q_1,r}$ where q_1 will update its primary w.r.t. r, i.e., $g_{q_1,r} < \mathsf{latest}_{q_1 \to q_1 \to r}(g'_{q_1,r})$. Naturally, $g'_{q_1,r} \notin \rho[e_1]$. It further follows that there must exist a minimum p-event $e_2 > g'_{q_1,r}$ 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. This contradiction is illustrated in Figure 3.3. ■

We call the set P from Lemma 3.4 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 3.4, demonstrates the "symmetric" nature of max-degree of synchronizations.

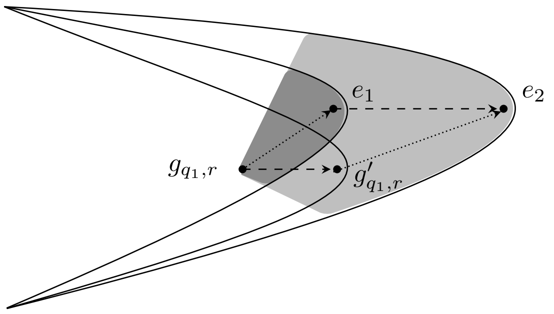


Figure 3.3: The contradiction of Lemma 3.4: $g_{q_1,r} <_r g'_{q_1,r}$, $e_1 <_p e_2$, $g_{q_1,r} < e_1$, and $g'_{q_1,r} < e_2$. The darker region comprises of event set $E_1 = \top_{\rho[e_1]}(g_{q_1,r})$, and $\mathsf{dom}(E_1) \supseteq P_1$. The lighter region, subsuming the darker region, comprises of event set $E_2 = \top_{\rho[e_2]}(g_{q_1,r})$, and $\mathsf{dom}(E_2) \supseteq P_1 \cup \{r\}$.

Corollary 3.5 *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 $P_i \in \Psi$: $q \in P_i \Leftrightarrow \lceil \mathsf{ds}_q(\rho) \rceil = P_i$. 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.

Now we are ready to define the family of ATS where for any run ρ and any event $e \in \rho$, each process $p \in \mathsf{dom}(e)$ is capable of computing $\mathsf{ds}(e)$ simply by looking at the local state $\pi_{|p}$ that p acquires after processing the event e.

Definition 3.6 *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 $\mathcal{D}_p(\pi_{|p}) = \mathsf{ds}(e)$.*

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. In this manner, a process p can "compute" the degree of synchronization of any p-event by observing the local p-state it acquires at that event. It is easy to see that condition 2 in Definition 3.6 is in fact decidable, whence the definition is "syntactic".

In the following section, we present a procedure to construct a SATS that is equivalent to a given ATS. We do this by equipping the local states of each process p of the input ATS with finite memory so that it can compute $\mathsf{ds}(e)$ at each event e it participates in. The reader may skip to Section 3.4 for ω-automata classes.

3.3 From ATS to SATS

For every ATS, the events of its runs have degrees of synchronization associated with them. However, not every ATS is capable of computing these via its local states. In this section, we present an algorithm that takes an arbitrary asynchronous transition system $\mathfrak{T}$ and constructs a synchronization aware transition system $\overline{\mathfrak{T}}$ that mimics the $\mathfrak{T}$ while being capable of computing the degrees of synchronization of events in the local states.

The natural idea in this construction is that the local states of $\overline{\mathfrak{T}}$ hold additional information regarding partial states acquired by $\mathfrak{T}$ in its run. Now the key question is, "what partial states are stored by the processes of $\overline{\mathfrak{T}}$, and how do the degrees of synchronization of various events help?"

We claim that at any event e in the run ρ of $\mathfrak{T}$, the set $P = \mathsf{ds}(e)$ indicates that there exist prefixes $\rho' \sqsubseteq \rho[e]$ with partial frontiers H, $\mathsf{dom}(H) = P$, such that for some process $p \in \mathsf{dom}(e)$ with a predecessor $f_p \lessdot_p e$, $H \not\subseteq \rho[f_p]$. That is to say that the frontier H is one of the largest frontiers that had been missing from the causal view of process p until p synchronized at event e. The following lemma illustrates this point, and demonstrates the importance of the set $\mathcal{U}_e$.

Lemma 3.7 *For $e \in \rho$, $e > e_\perp$, let $\rho_\sqcap := \bigsqcap_{f_p \lessdot_p e} \rho[f_p]$ be the greatest lower bound of all its p-prefixes, $p \in \mathsf{dom}(e)$. 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 Figure 3.2.

Example 3.8 *Referring to Figure 3.2, 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 3.7) 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 $U \coloneqq \{f_p\}$ for any $p \in \mathsf{dom}(e)$.

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

$$\begin{aligned}&\forall f \in H,\ \exists p \in \mathsf{dom}(e),\ \exists q, r \in \mathsf{ds}(e)\colon \\ &\qquad \mathsf{latest}_{p\to q\to r}(f_p) \leq f \ \wedge\ \mathsf{latest}_{p\to q\to r}(f_p) \neq \mathsf{latest}_{p\to q\to r}(e)\end{aligned} \tag{3.1}$$

Condition (3.1) is simply a restatement of the claim above, because if it is true then for each $f \in H$ we obtain $g_f = latest_{p\to q\to r}(f_p)$ 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 $U \coloneqq \bigcup_{f\in H}\{g_f\}$. Towards a contradiction, assume that condition (3.1) is false. Then its negation must be true, which is:

$$\begin{aligned}&\exists f \in H,\ \forall p \in \mathsf{dom}(e),\ \forall q, r \in \mathsf{ds}(e)\colon \\ &\qquad \mathsf{latest}_{p\to q\to r}(f_p) \leq f \ \Rightarrow\ \mathsf{latest}_{p\to q\to r}(f_p) = \mathsf{latest}_{p\to q\to r}(e)\end{aligned} \tag{3.2}$$

In particular, $\forall h \in \rho[e]\colon h \leq f \Rightarrow h \notin \mathcal{U}_e$. Since $H \nsubseteq \rho_\sqcap$, 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. (A.) $\mathsf{ds}(e) = \mathsf{dom}(H) \Rightarrow r \in \mathsf{ds}(e)$, 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 $h \in \mathcal{U}_e$; (B.) from condition (3.2) 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 (C.) $q' \notin \mathsf{dom}(e)$, 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 Figure 3.4.

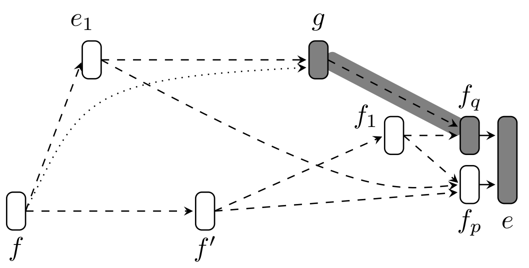


Figure 3.4: From the proof of Lemma 3.7: the shaded region lies outside of $\rho[f_p]$; $e_1, f_1 \in \mathsf{Pri}(f_p)$; there exist no r-events along any (dotted or dashed) paths from f to g.

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 finite 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 $f = \mathsf{latest}_{p\to q'\to r}(f_p) \neq \mathsf{latest}_{p\to q'\to r}(e) \geq f'$, implying that $f \in \mathcal{U}_e$ and contradicting the assumption that $f \notin \mathcal{U}_e$.

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 Figure 3.4, and proceed similarly to conclude 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 3.9 *The claim of Lemma 3.7 would not hold if $\mathcal{U}_e$ were defined as the set of "primary updates", i.e., if we define $\mathcal{U}_e := \{g \in \rho[e] \mid \exists p, q \in \mathcal{P}, \exists f \lessdot_p e\colon g = \mathsf{latest}_{p\to p\to q}(f_p) \neq \mathsf{latest}_{p\to p\to q}(e)\}$.*

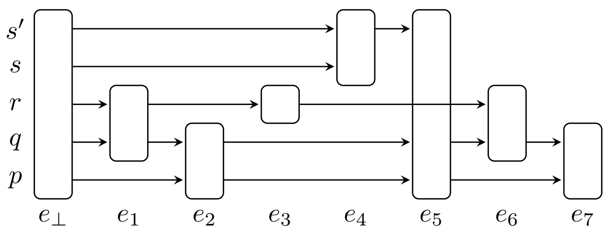


Figure 3.5: Considering only primary updates is insufficient.

Proof Consider the run $\rho := \rho[e_7]$ as shown in the Figure 3.5, 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, inserting $\mathcal{U}_e$ in Definition 3.3, we get $\mathsf{ds}(e_7) = \{p, q, r, s'\}$.

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 $\mathsf{dom}(e_4) = \{s, s'\} \nsubseteq \mathsf{ds}(e_7)$. Therefore, Lemma 3.7 breaks if we consider only the set of primary updates. ■

Corollary 3.10 *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 3.7.*

Lemma 3.7 illustrates that the degree of synchronization at event e corresponds precisely to the largest 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 becomes clear by juxtaposing Lemma 3.7 alongside with Lemma 3.4 and Corollary 3.5. We illustrate this with the help of Figure 3.6.

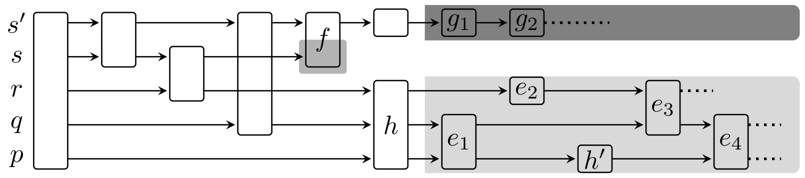


Figure 3.6: Different shades demarcate the different "maximally interacting sets".

After the infinite run partitions itself into maximally interacting sets, Lemma 3.4 guarantees that for processes that make infinitely many transitions, there will be infinitely many events where degrees of synchronization will correspond to these maximal sets. Events e_i and g_j are such events. At each of the events e_i, for example, the degrees of synchronization $\mathsf{ds}(e_i) = \{p, q, r\}$; although there can exist events, for example, h' where $\mathsf{ds}(h') \subsetneq \{p, q, r\}$. In the shaded regions, since the processes (except s) compute partial frontiers that correspond strictly to their maximally interacting sets, in the limit, the global frontiers of ρ can be obtained by straightforward "products" of these partial frontiers. These partial frontiers are precisely the ones that are revealed by Lemma 3.7. In Figure 3.6 above, s' computes partial frontiers $G_1 = \{g_1\}, G_2 = \{g_2\}, \ldots$; p, q, r compute[3] partial frontiers $H_1 = \{h\}, H'_1 = \{h, e_1\}, H_2 = \{h, e_2\}, H_3 = \{e_1, e_2\}, H_4 = \{h, e_1, h'\}$, and so on. The contribution of s remains fixed at $\{f\}$. Now it is clear that for each $i, j, H_i \cup G_j \cup \{f\}$ and $H'_i \cup G_j \cup \{f\}$ are global frontiers in ρ.

Presently, since we can compute global frontiers in the above manner, it should also be possible to compute global states[4] $\Lambda(H_i \cup G_j \cup \{f\})$, by referring to the partial states $\Lambda(H_i)$ and $\Lambda(G_j)$, and smartly "combining" them with $\Lambda(f)$.

We introduce a new term with respect to the partial frontiers H that are revealed by Lemma 3.7 at an event e. We refer to the set Y_e of all partial states $\Lambda(H)$ as the *yield at e*. For example, in Figure 3.6, $Y_{e_3} = \{\Lambda(\{h, e_1\}), \Lambda(\{h, e_2\}), \Lambda(\{e_1, e_2\}), \Lambda(\{e_1, e_3\})\}$. Notice that this is because $\{h, e_1\}$ is a partial frontier that r observes for the first time at e_3 (although q had already observed it at e_1); $\{h, e_2\}$ is one that q observes for the first time at e_3 (although r had already observed it at e_2); $\{e_1, e_2\}$ and $\{e_1, e_3\}$ are observed by both q, r for the first time at e_3. Clearly, for all partial states $\pi, \pi' \in Y_e$: $\mathsf{dom}(\pi) = \mathsf{dom}(\pi') = \mathsf{ds}(e)$. We say that a yield Y_e is *bigger* than yield Y_f if $\mathsf{ds}(f) \subsetneq \mathsf{ds}(e)$.

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 3.11 *Let $\pi_1 = (x_1, x_2, \ldots x_N)$ and $\pi_2 = (y_1, y_2, \ldots y_N)$ be two elements of $X_{2^{\mathcal{P}}}$. The* projection of π_2 on π_1 *is defined as $\pi_1 \lhd \pi_2 = (z_1, z_2, \ldots z_N)$ 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 3.12 *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 $\Lambda(H_1 \cup H_2) = \Lambda(H_1) \otimes \Lambda(H_2)$. To see these operations at work, we look at a new example.

[3]The partial frontiers H_i or H'_i are revealed by Lemma 3.7 at events e_i.

[4]Recall from near the end of Sec. 3.1, the definition of partial states $\Lambda(H)$ for partial frontiers H.

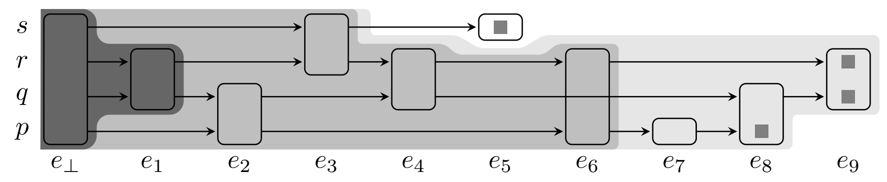


Figure 3.7: Partial frontiers for ρ: $\{e_5\}$, $\{e_9\}$, $\{e_5, e_9\}$, $\{e_8, e_9\}$, and $\{e_5, e_8, e_9\}$.

Example 3.13 *Refer to Figure 3.7, and observe that the global state of the ATS after the finite run prefix ρ may be computed as $\Lambda(\rho) = \Lambda(e_5) \otimes (\Lambda(e_8) \lhd \Lambda(e_9))$.*

For the partial frontier $\{e_8, e_9\}$, the projection $(\Lambda(e_8) \lhd \Lambda(e_9))$ computes the partial state $\Lambda(\{e_8, e_9\})$ as indicated by the gray boxes. This partial state is then joined with the partial $\{s\}$-state obtained from $\Lambda(e_5)$ to obtain the global state.

According to Lemma 3.4 it suffices that during an infinite run, processes p repeatedly compute only the partial frontiers that correspond to maximal degrees of p-synchronizations.

Lemma 3.7 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 3.14 *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*

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

Note that if $H = \top_\rho(e)$ then $\Lambda(H) = \Lambda(e) \lhd \rho[e \lhd H]$. Let us see the situation from Example 3.13 again in terms of projections of (partial) frontiers.

Example 3.15 *Referring once again to Figure 3.7, let the partial frontier $H_1 := \{e_8, e_9\}$. Note that $\rho[e_9 \lhd H_1] = (\$, \$, \$, \$)$.*

Now for H_1, consider events e_4 and e_8 for which we have $H_1 \cap \top_\rho(e_4) \neq \emptyset$ and $H_1 \cap \top_\rho(e_8) \neq \emptyset$. Note that since $\top_\rho(e_4) = H_1$ and $e_4 \notin H_1$, we have that $\Lambda(H_1) = \rho[e_4 \lhd H_1] = \Lambda(e_4) \lhd \rho[e_4 \lhd H_1]$.

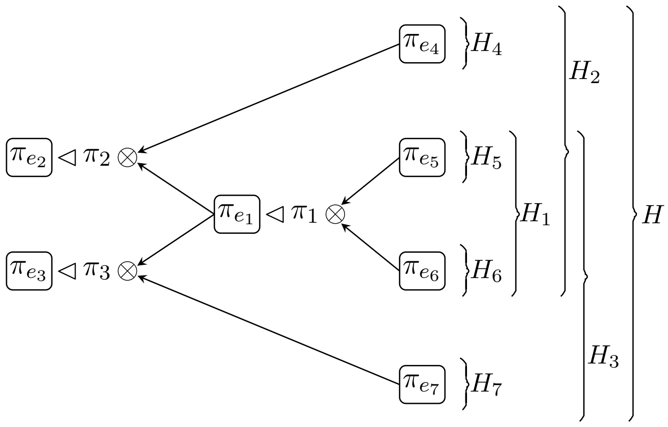
So also it holds that $\top_\rho(e_8) = H_1$. But since $e_8 \in H_1$, we have that $\rho[e_8 \lhd H_1] = \Lambda(e_9) \neq \Lambda(H_1) = \Lambda(e_8) \lhd \rho[e_8 \lhd H_1]$.

The following proposition formalizes the intuition presented above, and tightly couples Lemma 3.7 with Definition 3.14.

Proposition 3.16 *Let H be a partial frontier of ρ, and let $e_1, \ldots, e_n$ be pairwise concurrent events such that, for each $i, 1 \leq i \leq n$: $H \cap \top_\rho(e_i) \neq \emptyset$, and for each $i, 1 \leq i \leq n$: there exist partial frontiers $G, G' \subseteq H$ such that either $\top_\rho(e_i) = G$ or e_i is a maximal event in $\rho[G] \sqcap \rho[G']$.*

1. *The states* $\Lambda(e_i) \lhd \rho[e_i \lhd H]$, $1 \le i \le n$, *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* $\Lambda(H) = \bigotimes_{i=1}^{n}(\Lambda(e_i) \lhd \rho[e_i \lhd H])$.

Proposition 3.16 follows immediately from the definitions, and it shows that events $e_i \in \mathsf{Sec}(H)$ can be used to compute $\Lambda(H)$ inductively. Figure 3.8 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.



π_{e_4}
H_4
H_2
$\pi_{e_2} \lhd \pi_2 \otimes$
π_{e_5}
H_5
$\pi_{e_1} \lhd \pi_1 \otimes$
H_1
H
$\pi_{e_3} \lhd \pi_3 \otimes$
π_{e_6}
H_6
H_3
π_{e_7}
H_7

Figure 3.8: Using states $\pi_{e_i} := \Lambda(e_i)$ and projections $\pi_i := \rho[e_i \lhd H_i]$ to inductively reconstruct the partial states $\Lambda(H_i) = \pi_{e_i} \lhd \pi_i$ of partial frontiers H_i. For $i \in \{4,5,6,7\}, \pi_i = (\$, \ldots \$)$. Finally, $\Lambda(H) = (\pi_{e_2} \lhd \pi_2) \otimes (\pi_{e_3} \lhd \pi_3)$.

In Figure 3.8, the sets $\{e_2, e_3\}, \{e_1, e_4, e_7\}$, and $\{e_4, e_5, e_6, e_7\}$, satisfy both the conditions of Proposition 3.16. The state $\Lambda(H)$ may therefore be computed using any of these sets, using the inductive operations described in the figure.

Although Proposition 3.16 and Corollary 3.10 together demonstrate 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. The local states of $\overline{\mathfrak{T}}$ therefore need a way to "remember" certain partial states long enough so as to enable this computation at a later synchronization as per Lemma 3.7. Clearly, we require secondary information capable of storing partial states.

Definition 3.17 *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* $\overline{f} = (f, \Pi, \pi) \in \overline{\mathsf{Sec}}(e)$.

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 3.18 *Consider a run prefix ρ shown in Figure 3.9 below.*

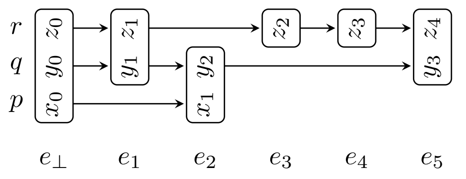


Figure 3.9: A run ρ where events are labeled with local states. $\mathsf{Sec}(e_5) = \{e_1, e_2, e_5\}$.

Event $e_1 \in \mathsf{Sec}(e_5)$ since $e_1 = \mathsf{latest}_{r\to q\to r}(e_5)$. 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 $\Pi_1 = \{\Lambda(e_3), \Lambda(e_4)\}$. This is because for $\rho' = \rho[e_3]$ whose frontier is $H = \{e_1, e_3\}$, we have $\Lambda(e_3) = \rho'[e_1 \lhd H]$; and for $\rho' = \rho[e_4]$ whose frontier is $H = \{e_1, e_4\}$, we have $\Lambda(e_4) = \rho'[e_1 \lhd H]$. For both these frontiers H, the states $\Lambda(H) = \Lambda(e_1) \lhd \rho[e_1 \lhd H]$ can now be computed locally as $\pi_1 \lhd \Pi_1$, with the help of π_1 and Π_1.

On the other hand, for $\rho' = \rho[e_2]$ we have $\top_{\rho'}(e_1) = \{e_1, e_2\} = H$, which is the frontier of ρ', and $\rho'[e_1 \lhd H] = (x_1, y_2, \$)$. But since $H \cap \top_{\rho'}(e_2) \neq \emptyset$ and $e_2 > e_1$ is such that $e_2 \in \mathsf{Sec}(e_5)$, we have that $(x_1, y_2, \$) \notin \Pi_1$.

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 Figure 3.8. Algorithm 3.1 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 an intermediate graph ι-$\overline{\mathsf{Sec}}(e)$. We would like to clarify that the set of events in graph $\overline{\mathsf{Sec}}(e)$ is not in one-to-one correspondence with the set of events in graph ι-$\overline{\mathsf{Sec}}(e)$. The former graph directly resembles the secondary graph $\mathsf{Sec}(e)$ at event $e \in \rho$. The latter graph, however, resembles the "union" of all secondary graphs $\mathsf{Sec}(f_p), f_p \lessdot_p e$; therefore possibly contains more events than $\overline{\mathsf{Sec}}(e)$.

The significance of ι-$\overline{\mathsf{Sec}}(e)$ is that it is a temporary data structure which will assist in computation of yields Y_e at e by allowing the computations of partial states for all partial frontiers H à la Lemma 3.7. Its nodes ι-$\overline{f}$ contain augmentations ι-Π, which in turn contain the "sum total" of all the information of all the secondary events greater than f. Formally, assuming that each $\overline{\mathsf{Sec}}(f_p)$ satisfies (∗), the following property holds for ι-$\overline{\mathsf{Sec}}(e)$.

For each event ι-$\overline{f} = (f, \iota\text{-}\Pi, \pi) \in \iota$-$\overline{\mathsf{Sec}}(e)$, there exists a partial state $\pi' \in \iota$-Π if and only if 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 $\pi' = (\$, \dots \$)$. (†)

Algorithm 3.1: SECINTERIM – computes intermediate secondary information

```
input  : the latest event $e$ and secondary graphs $\overline{\mathsf{Sec}}(f_p), f_p \lessdot_p e$
output : intermediate secondary graph $\iota\text{-}\overline{\mathsf{Sec}}(e)$ at $e$
/* This procedure reads secondary graphs $\overline{\mathsf{Sec}}(f_p)$, each of which
   satisfies the property $(*)$, and constructs an intermediate
   secondary graph $\iota\text{-}\overline{\mathsf{Sec}}(e)$ which satisfies the property $(\dagger)$.
   */
1  Create a working replica $r\text{-}\overline{\mathsf{Sec}}(f_p)$ of each set $\overline{\mathsf{Sec}}(f_p)$;
   /* That is, for each $\overline{g} = (g, \Pi, \pi) \in \overline{\mathsf{Sec}}(f_p)$, the replica $r\text{-}\overline{\mathsf{Sec}}(f_p)$
      contains $r\text{-}\overline{g} = (g, r\text{-}\Pi, \pi)$.  These working copies simply
      prevent the values of state sets $\Pi$ from being modified,
      which might result in violating property $(*)$.                */
2  Initialize the graph $\iota\text{-}\overline{\mathsf{Sec}}(e) = \emptyset$;
3  To each set $r\text{-}\overline{\mathsf{Sec}}(f_p)$, append $r\text{-}\overline{e} = (e, \{(\$, \ldots \$)\}, \Lambda(e))$ as the greatest
   event;
4  Consider $\{e_1, \ldots, e_n\} \coloneqq \{e\} \cup \bigcup_{p \in \mathsf{dom}(e)} \mathsf{Sec}(f_p)$, the topologically ordered
   set of all secondary events, with $e_n = e$;
5  for $i = n$ down to $1$ do
6      Initialize event $\iota\text{-}\overline{e}_i \coloneqq (e_i, \iota\text{-}\Pi_i, \pi_i)$, where $\iota\text{-}\Pi_i = \emptyset$ and $\pi_i = \Lambda(e_i)$;
7      Consider the maximal set $Q_i \subseteq \mathsf{dom}(e)$, such that for each $q \in Q_i$ there
       exists an $e_i$-copy $r\text{-}\overline{e}_i^q = (e_i, r\text{-}\Pi_i^q, \pi_i)$ in $r\text{-}\overline{\mathsf{Sec}}(f_q)$;
8      foreach $q \in Q_i$ do
9          Initialize $\Pi = \emptyset$;
10         Locally in $q$, let $\{r\text{-}\overline{g}_1, \ldots r\text{-}\overline{g}_\ell\} \subseteq r\text{-}\overline{\mathsf{Sec}}(f_q)$ be the set of all events
           $r\text{-}\overline{g}_k \gtrdot r\text{-}\overline{e}_i^q$, with $r\text{-}\overline{g}_k = (g_k, r\text{-}\Pi_k, \pi_k)$;
11         foreach $k, 1 \leq k \leq \ell$ do
12             Update $\Pi \coloneqq \Pi \cup (\Pi \otimes r\text{-}\Pi_k)$, assuming $\emptyset \otimes r\text{-}\Pi_1 = r\text{-}\Pi_1$;
13         end
14         Update $r\text{-}\overline{e}_i^q$ by reassigning $r\text{-}\Pi_i^q \coloneqq r\text{-}\Pi_i^q \cup (r\text{-}\Pi_i^q \lhd \Pi)$;
15         Update $\iota\text{-}\Pi_i \coloneqq \iota\text{-}\Pi_i \cup r\text{-}\Pi_i^q \cup (\iota\text{-}\Pi_i \otimes r\text{-}\Pi_i^q)$, assuming $\emptyset \otimes r\text{-}\Pi_i^q = r\text{-}\Pi_i^q$;
16     end
17     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;
18 end
19 return $\iota\text{-}\overline{\mathsf{Sec}}(e)$;
```

Proposition 3.19 *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 ι-$\overline{\mathsf{Sec}}(e)$ satisfies property $(\dagger)$.*

Proof Since the events ι-$\overline{e}_i$ are added one by one to the set ι-$\overline{\mathsf{Sec}}(e)$, we argue by induction over the set $\{e_1, \ldots, e_n\}$ of events as mentioned in step 4 of secInterim 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, ι-$\overline{e}_n = \iota$-$\overline{e} = (e, \{(\$, \ldots \$)\}, \Lambda(e))$, 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. Definition 3.14).

Now, assuming that all events down to index $\ell+1$ satisfy $(\dagger)$, we consider the ℓ^{th} iteration starting at Step 6. Let π' be a partial state in ι-Π_ℓ. If $\pi' \in \Pi^q_\ell$ already for some $\overline{e}^q_\ell \in \overline{\mathsf{Sec}}(f_q)$, $q \in Q_\ell$ then we have nothing to show since condition $(*)$ already holds, implying that condition $(\dagger)$ holds. Otherwise, the induction hypothesis holds and we know that events ι-$\overline{e}_{\ell+1}$, ι-$\overline{e}_{\ell+2}$, etc. satisfy the claim. Now, since the successor events g_k chosen in Step 10 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 $\rho' := \bigsqcup_{k=1}^m \rho_k \sqsubseteq \rho[e]$. 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, for e_ℓ consider a prefix $\rho' \sqsubseteq \rho[e]$, with $e_\ell \in \rho'$ and a partial frontier H' such that $H' = \top_{\rho'}(e_\ell)$. Consider the smallest set G containing the maximal events $g \in \{e\} \cup \bigcup_{p \in \mathsf{dom}(e)} \mathsf{Sec}(f_p)$ such that $H' = \bigcup_{g \in G} \top_{\rho'}(g)$. Clearly, the events contained in G have been processed before e_ℓ. Now there are two possibilities.

If $G = \{e_\ell\}$ then either $H' = \{e_\ell\}$ in which case, by definition, $\rho'[e_\ell \lhd H'] = (\$, \ldots \$)$ and is already a member of ι-Π_ℓ by the initialization Step 3. Otherwise, it must be the case that $\rho' \sqsubseteq \bigsqcup_{q \in Q_\ell} \rho[f_q]$. And then, H' may be expressed as $H' = \bigcup_{q \in Q_\ell} H_q$ where $H_q = H' \cap \rho[f_q]$ is a partial frontier in $\rho_q = \rho'[f_q]$. And therefore, from the induction hypothesis we have that $\rho'[e_\ell \lhd H'] = \bigotimes_{q \in Q_\ell} \rho_q[e_\ell \lhd H_q]$ is an element of ι-Π_ℓ as a consequence of condition $(*)$ holding over $\overline{\mathsf{Sec}}(f_q)$ and the join operation in Step 15.

If $|G| > 1$, then denote the elements of G as $e_\ell = g_0, g_1, \ldots, g_m$. For $k > 1$, let $H_k = \top_{\rho'}(g_k)$. Since these events g_k are are pairwise concurrent, by Proposition 3.16, the states $\Lambda(H_k) = \Lambda(g_k) \lhd \rho'[g_k \lhd H_k]$ are mutually compatible, and by induction hypotheses $\rho'[g_k \lhd H_k] \in \iota$-$\Pi_k$. From the compatibility of states, it is clear that the order in which events g_k are joined in the local updates on r-$\overline{e}^q_\ell$ in step 10 is immaterial. Finally, if $\bigcup_k H_k = H'$ then nothing more remains to be shown, since the local updates already ensure that $\Lambda(H') \in \iota$-Π_ℓ. Otherwise, since $(*)$ holds on all sets $\overline{\mathsf{Sec}}(f_q)$, using an argument similar to the case of $G = \{e_\ell\}$ we conclude that there must exist states $\pi_q \in \overline{\mathsf{Sec}}(f_q)$ for $q \in Q_\ell$, such that at the end of step 10 for event e_ℓ, we have $\rho'[e_\ell \lhd H'] = \bigotimes_q(\pi_q \lhd \Lambda(H'))$, thereby implying that $(\dagger)$ holds for index ℓ as well. ■

Remark 3.20 *Property (†) and Corollary 3.10 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 $\overline{\mathcal{M}}_e = \{\iota\text{-}\overline{e}_1, \ldots, \iota\text{-}\overline{e}_\ell\} \subseteq \iota\text{-}\overline{\mathsf{Sec}}(e)$ of the minimal iota-events, then processes $p \in \mathsf{dom}(e)$ can compute the yield $Y_e \coloneqq \{\pi \in \bigotimes_{i=1}^{\ell}(\pi_i \lhd \iota\text{-}\Pi_i) \mid \mathsf{dom}(\pi) = \mathsf{ds}(e)\}$ which contains the partial states for all the partial frontiers H as in Lemma 3.7.*

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 Algorithm 3.2.

Algorithm 3.2: SECUPDATE – updates secondary information

```
input  : the latest event $e$ and secondaries $\overline{\mathsf{Sec}}(f_p), f_p \lessdot_p e$
output : secondary information $\overline{\mathsf{Sec}}(e)$ at $e$
/* This procedure reads secondary graphs $\overline{\mathsf{Sec}}(f_p)$, each of which
   satisfies the property $(*)$, and constructs the new secondary
   graph $\overline{\mathsf{Sec}}(e)$ which again satisfies the property $(*)$.      */
1  Initialize $\overline{\mathsf{Sec}}(e) = \emptyset$; append $\overline{e} = (e, \{(\$, \ldots \$)\}, \Lambda(e))$ as its greatest event;
2  Consider $\{e_1, \ldots, e_n\} \coloneqq \{e\} \cup \bigcup_{p \in \mathsf{dom}(e)} \mathsf{Sec}(f_p)$, the topologically ordered
   set of all secondary events, with $e_n = e$;
3  for $i = n$ down to $1$ do
4      Initialize event $\overline{e}_i \coloneqq (e_i, \Pi_i, \pi_i)$, where $\Pi_i = \emptyset$ and $\pi_i = \Lambda(e_i)$;
5      Consider the maximal set $Q_i \subseteq \mathsf{dom}(e)$, such that for each $q \in Q_i$ there
       exists an $e_i$-copy $\overline{e}_i^q = (e_i, \Pi_i^q, \pi_i)$ in $\overline{\mathsf{Sec}}(f_q)$;
6      foreach $q \in Q_i$ do
7          Locally in $\overline{\mathsf{Sec}}(f_q)$, let $\{\overline{g}_1, \overline{g}_2 \ldots\}$ be the set of all events $\overline{g}_k \gtrdot \overline{e}_i^q$ with
           $\overline{g}_k \in \partial_e$, and let $\overline{g}_k = (g_k, \Pi_k, \pi_k)$;
8          Initialize a temporary variable $\Pi = \emptyset$;
9          foreach $k, 1 \leq k \leq \ell$ do
10             Update $\Pi \coloneqq \Pi \cup (\Pi \otimes \Pi_k)$, assuming $\emptyset \otimes \Pi_1 = \Pi_1$;
11         end
12         Update $\overline{e}_i^q$ by assigning $\Pi_i^q \coloneqq \Pi_i^q \cup (\Pi_i^q \lhd \Pi)$;
13         Update $\Pi_i \coloneqq \Pi_i \cup \Pi_i^q \cup (\Pi_i \otimes \Pi_i^q)$, assuming $\emptyset \otimes \Pi_i^q = \Pi_i^q$;
14     end
15     If $e_i \notin \partial_e$ then add $\overline{e}_i \coloneqq (e_i, \Pi_i, \pi_i)$ in $\overline{\mathsf{Sec}}(e)$ as per the partial order;
16 end
17 return $\overline{\mathsf{Sec}}(e)$;
```

Proposition 3.21 *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 Proposition 3.19. The only difference between this procedure and secInterim procedure is that, while in the latter the newly computed augmentations ι-Π_i from step 10 are always projected down to immediate predecessors, in secUpdate procedure the newly computed augmentations Π_i from step 7 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 $\overline{\mathfrak{T}} = \Big(((\overline{X}_p)_{p\in\mathcal{P}}, (\overline{\delta}_a)_{a\in\Sigma}, \overline{\pi}_0), \mathcal{D}\Big)$ where:

- $\overline{X}_p \subseteq \bigcup_{Q\subseteq\mathcal{P}} X_p \times \overline{\mathsf{SEC}}_p \times 2^{X_Q}$, 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 secUpdate(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 secInterim(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 correspond to the partial frontiers which had been missing from the view of one or the other process until they synchronized at e. In this manner, a processes p in SATS $\overline{\mathfrak{T}}$ arrives at the local state $(x, \overline{\mathsf{Sec}}, Y)$ only if process p in ATS $\mathfrak{T}$ arrives at the local state x and the yield is Y at the event in question. The augmented secondary information $\overline{\mathsf{Sec}}$ helps $\overline{\mathfrak{T}}$ in correctly computing yields.

In the next sections, we study automata models defined using synchronization aware transition systems; and we see how SATS constructed in this manner presented here can be exploited to show the characterization results for deterministic Büchi recognizable trace languages and recognizable ω-trace languages.

3.4 Synchronization Aware Büchi Automata

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

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

Clearly, $\lceil \mathsf{Inf}_p(\rho) \rceil \subseteq \mathsf{Inf}_p(\rho)$, and can be understood as a "ceiling" of the local infinity set $\mathsf{Inf}_p(\rho)$ in the sense that it retains only those local states that correspond to the max-degree of p-synchronizations[5] in ρ.

Definition 3.22 *A* deterministic, synchronization aware 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 max-degree of p-synchronizations. A high level intuition behind this – and a reason why acceptance of traces via classical asynchronous Büchi automata[6] is not as straightforward as acceptance of words via Büchi automata – is as follows (also refer to Example 2.15).

Example 3.23 *Consider an asynchronous automaton over finite traces, which has two processes and whose global acceptance state set $\mathcal{F} = \{(x_1, y_1)\}$ 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 Figure 3.10 below.*

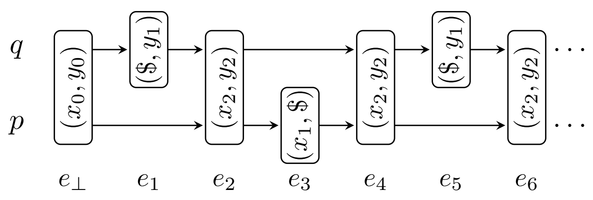


Figure 3.10: The run segment corresponding to $e_1e_2e_3e_4$ repeats ad infinitum. Local states x_1 and y_1 occur infinitely often, but the global state (x_1, y_1) never occurs.

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

[5]Recall the definition from Section 3.1
[6]See definition of DABA in Section 2.3.

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 summarize in hindsight the global states witnessed up to e_i; and only such repeated global summaries should govern whether or not to accept a trace.

The main result of this section is Theorem 3.27, and we proceed piecemeal towards it, via Lemma 3.24, Lemma 3.25, and Proposition 3.26.

Lemma 3.24 *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 $\mathfrak{A} = (\mathfrak{T}, \mathcal{F})$ recognizing a language $L(\mathfrak{A}) = T \subseteq \mathbb{M}(\Sigma, I)$ and first construct an SATS $\overline{\mathfrak{T}}$ from the ATS $\mathfrak{T}$ as demonstrated in the previous section. The definition of a D-SABA $\overline{\mathfrak{A}} = (\overline{\mathfrak{T}}, \mathcal{D}, \overline{\mathcal{F}})$ recognizing $\mathsf{lim}(T)$ now relies on suitably describing the Büchi acceptance condition.

Fix a global accepting state $\pi \in \mathcal{F}$ of $\mathfrak{A}$, a set $Q \subseteq \mathcal{P}$ of processes that make infinitely many transitions, and a partition $\Psi = \{P_1, \ldots P_n\}$ of $\mathcal{P}$ compatible with Q, i.e., $P_k \cap Q \neq \emptyset \Rightarrow P_k \subseteq Q$ and $P_k \cap Q = \emptyset \Rightarrow |P_k| = 1$.

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 $\pi_{|P_k}$, where $P_k \in \Psi$ 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_k witnesses the partial state $\pi_{|P_k}$ at some frontier, then sooner or later every other process in P_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, 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 $x = \pi_{|p}$ and $\mathcal{D}_p(\overline{x}) = \mathsf{dom}(Y) = \mathcal{S}_p$. 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. That is, $\mathcal{P} \setminus Q$ can be non-empty only if Ψ contains singletons. And in particular, if $Q = \mathcal{P}$ then $\mathcal{S}$ becomes redundant.

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

Proof (Lemma 3.24) Let $\mathfrak{A} = (\mathfrak{T}, \mathcal{F})$ 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 $\overline{\mathcal{F}} := \bigcup_\pi \bigcup_Q \bigcup_\Psi \bigcup_\mathcal{S} \{(Q, \overline{F}_{\pi,Q,\Psi,\mathcal{S}})\}$, we claim that $\overline{\mathfrak{A}} := (\overline{\mathfrak{T}}, \mathcal{D}, \overline{\mathcal{F}})$ 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 $(Q, (\overline{F}^p)_{p \in \mathcal{P}}) \in \overline{\mathcal{F}}$ 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 $\Psi = \{P_1, \dots P_n\}$ 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. $P_k \nsubseteq Q$, 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 3.25 *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*[7] $\Theta = \mathsf{lim}_A(T)$.

Once again, before we begin the proof, we first present the necessary constructions. 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

[7]Recall Definition 2.16: $\mathsf{lim}_A(T) := \{\theta \in \mathsf{lim}(T) \mid D(\mathsf{alphinf}(\theta)) = D(A)\}$.

assume that (Q, F) is realizable. That is, for all $p, q \in Q$: $q \in \mathcal{D}_p(F^p) \Leftrightarrow \mathcal{D}_p(F^p) = \mathcal{D}_q(F^q)$. Recall that by the definition of D-SABA, if (Q, F) is realizable then for each p, the component F already determines the max-degree of p-synchronizations of any accepting run of $\mathfrak{A}$. 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 A of letters that can occur infinitely often in any $\theta \in L(\mathfrak{A})$ is precisely the set we fix as parameter, and is given by $A := \mathsf{dom}^{-1}(Q) \setminus \mathsf{dom}^{-1}(\mathcal{P} \setminus Q)$. Note that this does not imply that each letter in A must occur infinitely often in θ, since in general, for $A_1, A_2 \subseteq \Sigma$, if $D(A_1) = D(A_1)$ then $\lim_{A_1}(T) = \lim_{A_2}(T)$. In this sense, A is the largest possible set that is "compatible" with (Q, F).

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 $q \in \mathcal{D}_p(F^p)$, i.e., p, q have identical maximally interacting sets. 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 "cross-over" transitions 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 to the sole Büchi acceptance pair $(Q, F) \in \mathcal{F}$. 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 $\mathfrak{B} = (\overline{\mathfrak{T}}, \overline{\mathcal{F}})$ 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 $\overline{\mathcal{F}}$ as follows:

- if $p \in Q$, then $\overline{X}_p \subseteq X_p \times (2^{X_q})_{q \in \mathcal{P}} \times \{0, 1\}$,
- otherwise, if $p \notin Q$, then $\overline{X}_p \subseteq X_p \times (2^{X_q})_{q \in \mathcal{P}} \times \{0\}$;
- $\overline{\pi}_{0|p} = (x_0, (\emptyset, \ldots \emptyset), 0)$ where $x_0 = \pi_{0|p}$ is the initial p-state of $\mathfrak{T}$;
- for $a \in A$ the transition relation $\Delta_a := \delta_{a,\times} \cup \bigcup_{i \in \{0,1\}} \delta_{a,i}$, 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 $\delta_{a,i}(\overline{\pi})_{|p} := \big(y_p, (Y_{p \to q})_{q \in \mathcal{P}}, i\big)$ 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[8] as follows:
    1. First define $Z_{p\to q}$ as
       a) if $q \in \mathcal{D}_p(F^p) \cap \mathsf{dom}(a)$, then $Z_{p\to q} \coloneqq \{y_q\} \cup \bigcup_{r\in\mathsf{dom}(a)} X_{r\to q}$;
       b) else if $q \in \mathcal{D}_p(F^p) \setminus \mathsf{dom}(a)$, then $Z_{p\to q} \coloneqq \bigcup_{r\in\mathsf{dom}(a)} X_{r\to q}$.
    2. If for each $q \in \mathcal{D}_p(F^p)$: $Z_{p\to q} \cap F^q \neq \emptyset$ then $\forall p \in \mathsf{dom}(a), q \in \mathcal{P}$ define $Y_{p\to q} \coloneqq \emptyset$; else define $Y_{p\to q} \coloneqq Z_{p\to q}$.
- – for $\mathsf{dom}(a)$-states $\overline{\pi} = \big((x_p, (X_{p\to q})_{q\in\mathcal{P}}, 0)\big)_{p\in\mathsf{dom}(a)}$, define the mapping $\delta_{a,\times}(\overline{\pi})_{|p} \coloneqq \big(y_p, (Y_{p\to q})_{q\in\mathcal{P}}, 1\big)$ 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.

- $\overline{\mathcal{F}} \subseteq \overline{X}_{\mathcal{P}}$ is a set containing all global states $\pi = \big((x_p, (X_{p\to q})_{q\in\mathcal{P}}, i_p)\big)_{p\in\mathcal{P}}$ of $\mathfrak{T}$ where $\big[$for each $p \notin Q$, $x_p \in F^p \wedge i_p = 0\big]$ <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 $r \in \mathcal{D}_p(F^p)$ whose local r-state $\pi_{|r} = \big(x_r, (Y_{r\to q})_{q\in\mathcal{P}}, 1\big)$ is such that for each $q \in \mathcal{P}$: $Y_{r\to q} = \emptyset\big]$.

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 3.25) 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 $A = \mathsf{dom}^{-1}(Q) \setminus \mathsf{dom}^{-1}(\mathcal{P} \setminus Q)$ is computed as shown previously, then the NAA $\mathfrak{B} = \big((\overline{X}_p)_{p\in\mathcal{P}}, (\Delta_a)_{a\in\Sigma}, \overline{\pi}_0, \overline{F}\big)$ 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 partition of processes from Q as induced by the run on θ (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 $\mathfrak{B}$.

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

First, we identify a prefix $t_\theta \sqsubset \theta$ such that $\forall q \in \mathcal{P} \setminus Q\colon \max_q(\theta) \in t_\theta$ and $\forall e \in \theta \setminus t_\theta, \exists P \in \Psi\colon \mathsf{ds}(e) \subseteq P$. 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 each part $P \in \Psi$.

Given a prefix $t_i, i > 0$, 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 $(x_p, (\emptyset, \dots \emptyset), 0)$ 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 Δ_a nondeterministically selecting the cross-over transition using the 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 3.4 that there must exist at least one event e_P such that $\theta[e_P] \setminus t_\theta$ contains 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 $(x_p, (\emptyset, \dots \emptyset), 0)$ 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 3.26 *The family of D-SABA-recognizable languages is closed under finite unions.*

Proof 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 Lemma 3.24, Lemma 3.25, and Proposition 3.26, we conclude the following.

Theorem 3.27 *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 [Mus94]. We can now state the same result in terms of the family of D-SABA recognizable languages.

Proposition 3.28 *The family of D-SABA-recognizable languages is closed under finite intersections.*

To conclude the discussion the deterministic trace languages, we define co-Büchi automata that characterize the class of recognizable ω-trace languages that can be described as complements of D-SABA recognizable languages.

Definition 3.29 *A* deterministic, synchronization aware co-Büchi automaton *(a* D-SAcBA*) 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-SAcBA* $\mathfrak{A}$ accepts a trace $\theta \in \mathbb{R}(\Sigma, I)$ *if, for the run* ρ *of* $\mathfrak{A}$ *on* θ*, for any pair* $(Q_i, F_i) \in \mathcal{F}$*, if* $\mathsf{dom}(\mathsf{alphinf}(\theta)) = Q_i$ *then there exists* $p \in \mathcal{P}$ *such that* $p \in \mathcal{P} \colon \lceil \mathsf{Inf}_p(\rho) \rceil \subseteq F_i^p$*.*

The following corollary follows from the definitions and previous results.

Corollary 3.30 *A language* $\Theta \subseteq \mathbb{R}(\Sigma, I)$ *is recognized by a D-SABA* $\mathfrak{A}$ *if and only if the complement language* $\mathbb{R}(\Sigma, I) \setminus \Theta$ *is recognized by a D-SAcBA* $\mathfrak{B}$*.*

3.5 Synchronization Aware Muller Automata

Over synchronization aware 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 3.31 *A* deterministic synchronization aware 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.* $F_i = (F_i^p)_{p\in\mathcal{P}}$ *are tuples of homosynchronous sets* F_i^p*. 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* $p \in \mathcal{P} \colon \lceil \mathsf{Inf}_p(\rho) \rceil = F_i^p$*.*

Theorem 3.32 *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 3.32($\Rightarrow$)) 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 asynchronous transition system 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 Section 3.4, 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 $(i, Q, \Psi, \mathcal{S})$ 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 $\forall p \in \mathcal{P}\colon (p \notin Q \Rightarrow |F_i^p| = 1) \wedge (|F_i^p| > 1 \Rightarrow p \in Q)$;
- partitions Ψ are consistent with the choice of Q, that is $\forall P_k \in \Psi\colon (P_k \cap Q \neq \emptyset \Rightarrow P_k \subseteq Q) \wedge (P_k \cap Q = \emptyset \Rightarrow |P_k| = 1)$;
- 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 $q \in S$. 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, Ψ, and $\mathcal{S}$.

So $\mathbb{I}$ must at least be of the form $(i, Q, \Psi, \mathcal{S}, \ell)$, 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_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 $p \in P_k$ must be confined to a suitable subset of $\bigotimes_{p\in P_k} F_i^p$ which covers each local q-state in $F_i^q, q \in P_{j,k}$. Formally, given i, Q, and Ψ

- we consider a collection $\mathcal{Y} = (\mathcal{Y}_p)_{p\in Q}$ where each set $\mathcal{Y}_p \subseteq 2^{X_{P_k}}$ of partial P_k-states of $\mathfrak{A}$ satisfies the conditions:
  1. For each $Y \in \mathcal{Y}_p$, $Y \subseteq \bigotimes_{q\in P_k} F_i^q$; and
  2. For each $q \in P_k$ 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 $(i, Q, \Psi, \mathcal{S}, \ell, \mathcal{Y})$ 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 $q \in P_k$ visit every state from the local state sets F^q_i.

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, Ψ 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 $x \in F^p_i$ 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 $(i, Q, \Psi, \mathcal{S}, \ell, \mathcal{Y}, m)$, 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 $\mathbb{I} = (i, Q, \Psi, \mathcal{S}, \ell, \mathcal{Y}, m)$ and $\mathbb{I}' = (i, Q', \Psi, \mathcal{S}', \ell', \mathcal{Y}', m')$. 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 either of them; and this creates no problems.

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 3.33 *For any index $\mathbb{I} = (i, Q, \Psi, \mathcal{S}, \ell, \mathcal{Y}, m)$ 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 $F^p_i = \{x\}$.*

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 $i, Q, \Psi, \mathcal{S}$ above, this can happen only if, p belongs to a singleton part in the partition Ψ, which itself is chosen in consistence with Q. And Q in turn could be chosen to exclude p only if $|F^p_i| = 1$. And by construction, $\overline{F}^p_{\mathbb{I}} = \{(x, \overline{\mathsf{Sec}}, Y)\}$ only if $F^p_i = \{x\}$.

Second scenario is the case when $p \in Q$. In this case, the conditions governing the choice of $\overline{F}^p_{\mathbb{I}}$ are described in the discussion of index m. If $\overline{F}^p_{\mathbb{I}} = \{(x, \overline{\mathsf{Sec}}, Y)\}$ then the three conditions together mandate that $\mathcal{Y}_p = \{Y\}$; that P_k-states in Y cover all the q-states occurring in F^q_i, $q \in P_k$; and condition 3 especially mandates that for any $\pi \in Y$ there exists a state $(x, \overline{\mathsf{Sec}}, Y) \in \overline{F}^p_{\mathbb{I}}$ with $\pi_{|p} = x$. And since $\overline{F}^p_{\mathbb{I}}$ 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^q_i, $q \in P_k$. In particular, Y covers F^p_i. Hence $F^p_i = \{x\}$. ■

Proof (Theorem 3.32($\Leftarrow$)) 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 such a component $(\overline{F}^p)_{p\in\mathcal{P}}$ exists

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 Proposition 3.33, 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 3.34 *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 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 $X_i^p := \{x \in X_p \mid \mathcal{D}_p(x) = Q_i^p\}$. Define $\mathcal{F}_\neg := \bigcup_i \Big((\prod_{p \in \mathcal{P}} 2^{X_i^p}) \setminus \{F_j \in \mathcal{F} \mid \forall p \in \mathcal{P} \colon \mathcal{D}_p(F_j^p) = Q_i^p\} \Big)$. 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 $Y_p := \{(x, x') \in X_p \times X'_p \mid \mathcal{D}_p(x) = \mathcal{D}'_p(x')\}$. ■

3.6 Characterization of Deterministic Büchi Recognizability

A prominent result on ω-regular word languages, due to Landweber [PP04], states that a language $L \subseteq \Sigma^\omega$ is deterministically Büchi recognizable iff for some (in fact, for each) deterministic Muller automaton recognizing L the acceptance component – assuming it contains only realizable sets – is closed under supersets (see Theorem 1.10). The stronger (bracketed) version supplies a decision procedure for Büchi recognizability of ω-regular languages. Here we present a weaker existential characterization over infinite traces. We define supersets in a manner that retains the essence of acceptance tables. Consider $F_1 = (F_1^p)_{p \in \mathcal{P}}$ and $F_2 = (F_2^p)_{p \in \mathcal{P}}$ from $\mathcal{F}$ where both F_1 and F_2 are tuples of homosynchronous sets F_1^p and $F_2^p, p \in \mathcal{P}$. We say that F_1 *is a superset of* F_2 denoted $F_1 \supseteq F_2$ if for each $p \in \mathcal{P}$, $F_1^p \supseteq F_2^p$. A table $\mathcal{F}$ is said to be *closed under supersets* if $\big((F \in \mathcal{F}) \wedge (F' \supseteq F)\big) \Rightarrow (F' \in \mathcal{F})$.

While discussing the closure under supersets, we must exempt the acceptance tuples that guarantee the halting of some processes. Let $F \in \mathcal{F}$ be a realizable acceptance tuple with $F^p = \{x\} \subseteq X_p$ for some $p \in \mathcal{P}$. If a run ρ is accepted by referring to F, then it must be the case that process p makes $\underline{\text{finitely}}$ many transitions in ρ only if it is the case that during two successive visits to x, p must visit another state $y \in X_p$ such that $\mathcal{D}_p(y) \subsetneq \mathcal{D}_p(x)$. Then p must halt because otherwise, either $\lceil \mathsf{ds}_p(\rho) \rceil \supsetneq \mathcal{D}_p(x)$ or $\lceil \mathsf{Inf}_p(\rho) \rceil \supsetneq \{x\}$. Such a singleton F^p is referred to as a *finitary acceptance set*.

Definition 3.35 *A Muller acceptance table* $\mathcal{F}$ *is said to be* closed under supersets modulo finitary acceptance sets *if*

(a) whenever $F \in \mathcal{F}$ *does not contain any finitary acceptance sets and* $F' \supseteq F$*, then* $F' \in \mathcal{F}$*; and*

(b) whenever $F \in \mathcal{F}$ *contains finitary acceptance sets* F^p *and* $F' \supseteq F$ *with* $F'^p = F^p$ *for all such* p*, then* $F' \in \mathcal{F}$*.*

Theorem 3.36 *A language Θ is recognized by a D-SABA $\mathfrak{B} = (\mathfrak{T}', \mathcal{D}', \mathcal{F}')$ if and only if Θ is recognized by a D-SAMA $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$ whose acceptance table $\mathcal{F}$ is closed under supersets modulo finitary acceptance sets.*

For the proof of this theorem, we would like to recall the data structure called *latest appearance record*. For a finite set $X = \{x_1, \dots x_N\}$, denote with $X!$ the set of all permutations of X. We define the latest appearance record $\mathsf{LAR} \coloneqq X! \times [1, N]$. Any $M = ((x_{i_1} x_{i_2} \dots x_{i_N}), m) \in \mathsf{LAR}$ is a pair containing a permutation $(x_{i_1} x_{i_2} \dots x_{i_N})$ of X and $1 \leq m \leq N$. The number m is usually called the *hit value*, and the set $\{x_{i_1}, \dots x_{i_m}\}$ of the first m elements in the permutation is referred to as the *hit set* of M. We also refer to an update function $\upsilon\colon \mathsf{LAR} \times X \to \mathsf{LAR}$ is given by $\upsilon\colon ((x_{i_1}, \dots x_{i_N}), m), x \mapsto ((x_{i_\ell}, x_{i_1}, \dots x_{i_{\ell-1}}, x_{i_{\ell+1}}, \dots x_{i_N}), \ell)$ where $x = x_{i_\ell}$.

Now, for the set X_p of local p-states of $\mathfrak{T}$, let $X_{p,Q_1}, X_{p,Q_2}, \dots X_{p,Q_{n_p}}$ be the maximal homosynchronous subsets of X_p. That is, $\forall x \in X_p\colon x \in X_{p,Q_i} \Leftrightarrow \mathcal{D}_p(x) = Q_i$. For each $i, 1 \lneq i \leq n_p$, we now define the latest appearance record $\mathsf{LAR}_{p,Q_i} \coloneqq (X_{p,Q_i})! \times [1, N_i]$ where $N_i \coloneqq |X_{p,Q_i}|$. Note that it may be the case that $n_p \neq n_q$ for $p, q \in \mathcal{P}$.

Proof (Theorem 3.36) For one direction of the theorem, let $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$ be a D-SAMA such that $\mathcal{F}$ is closed under supersets modulo finitary acceptance sets.

Construct a D-SABA $\mathfrak{B} = (\mathfrak{T}', \mathcal{D}', \mathcal{F}')$ where the local p-state sets X'_p of $\mathfrak{T}'$ are given by $X'_p \coloneqq X_p \times \mathsf{LAR}_{p,Q_1} \times \mathsf{LAR}_{p,Q_2} \times \dots \times \mathsf{LAR}_{p,Q_{n_p}}$. The initial local p-state in $\mathfrak{T}'$ is $(\pi_{0|p}, M_{p,1}, \dots, M_{p,n_p})$ where π_0 is the global initial state of $\mathfrak{T}$ and $M_{p,i}$ are arbitrarily chosen initial records. The transition functions are given by $\delta'_a\colon \big((x_p, M_{p,1}, \dots, M_{p,n_p})\big)_{p \in \mathsf{dom}(a)} \mapsto \big((y_p, L_{p,1}, \dots, L_{p,n_p})\big)_{p \in \mathsf{dom}(a)}$ where:

- $(y_p)_{p \in \mathsf{dom}(a)} = \delta_a\big((x_p)_{p \in \mathsf{dom}(a)}\big)$, and
- if $\mathcal{D}_p(y_p) = Q_{i_p}$ then $L_{p,j} \coloneqq \begin{cases} \upsilon(M_{p,j}, y_p) & \text{if } j = i_p \\ M_{p,j} & \text{otherwise.} \end{cases}$

The mapping $\mathcal{D}'_p$ for $\mathfrak{T}'$ is defined as $\mathcal{D}'_p((x_p, \dots, M_{p,i}, \dots)) \coloneqq \mathcal{D}_p(x_p)$.

In order to define the Büchi acceptance table, consider any Muller acceptance tuple $F \in \mathcal{F}$ and a set $R \subsetneq \mathcal{P}$ such that if $r \in R$ then $|F^r| = 1$, and if F^r is a finitary acceptance set then $r \in R$. Intuitively, the processes in R are earmarked as precisely the ones that will halt. Create a Büchi acceptance tuple $(\mathcal{P} \setminus R, F'_R)$ where for each $p \in \mathcal{P}$, assuming $\mathcal{D}_p(F^p) = Q_k$,:

- if $p \notin R$ then $F'^p_R \coloneqq \{(x, M_{p,1}, \dots, M_{p,k}, \dots, M_{p,n_p}) \in X'_p \mid \mathcal{D}_p(x) = Q_k \ \underline{\text{and}}$ the hit set $H_{p,k}$ of $M_{p,k}$ is a superset of $F^p\}$; otherwise
- if $p \in R$ then $F'^p_R \coloneqq \{(x, M_{p,1}, \dots M_{p,n_p}) \in X'_p \mid \mathcal{D}_p(x) = Q_k \ \underline{\text{and}} \ x \in F^p\}$.

Note that for a given Muller acceptance tuple $F \in \mathcal{F}$, we may obtain multiple Büchi acceptance tuples $(\mathcal{P} \setminus R_1, F'_{R_1}), (\mathcal{P} \setminus R_2, F'_{R_2}), \dots$ where each R_i contains (some of) the processes r for which the corresponding Muller set F^r is singleton. R_i may be empty only if there are no finitary acceptance sets in F.

Now, a trace $\theta \in \mathbb{R}(\Sigma, I)$ is accepted by the D-SABA $\mathfrak{B}$:

iff for the run ρ' of $\mathfrak{A}'$ on θ there exists $(Q, F') \in \mathcal{F}'$ such that for each $p \in \mathcal{P}$, $F'^p \cap \lceil \mathsf{Inf}_p(\rho') \rceil \neq \emptyset$; and this holds

iff for each processes $p \in \mathcal{P}$

- if $p \in Q$, then p witnesses a state $(x, X_1^p, \ldots, X_{i_p}^p, \ldots) \in F'^p$ infinitely often, where $\lceil \mathsf{ds}_p(\rho') \rceil = \mathcal{D}'_p((x, X_1^p, \ldots, X_{i_p}^p, \ldots)) = \mathcal{D}_p(X_{i_p}^p \cup \{x\})$; and
- if $p \notin Q$, then p halts at a state $(x, X_1^p, \ldots) \in F'^p$ – and in this case, let $X_{i_p}^p := \{x\}$;

and, by construction, this holds

iff $(X_{i_p}^p)_{p \in \mathcal{P}} \supseteq F$ for some Muller tuple $F \in \mathcal{F}$, satisfying $\forall p \notin Q \colon F^p = X_{i_p}^p$; and this holds

iff $(X_{i_p}^p)_{p \in \mathcal{P}} = G$ for some $G \in \mathcal{F}$ satisfying $\forall p \notin Q \colon G^p = F^p$ (since $\mathcal{F}$ is closed under supersets modulo finitary acceptance sets); and this holds

iff θ induces a run ρ on the D-SAMA $\mathfrak{A}$ such that for each process $p \in \mathcal{P}, G^p = \lceil \mathsf{Inf}_p(\rho) \rceil$; and this holds

iff the D-SAMA $\mathfrak{A}$ accepts θ.

For the other direction of the theorem, let us assume that Θ is recognized by a D-SABA $\mathfrak{B} = (\mathfrak{T}', \mathcal{D}', \mathcal{F}')$. We define a D-SAMA $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$ whose acceptance table $\mathcal{F}$ is closed under supersets modulo finitary acceptance sets as follows:

- for each $p \in \mathcal{P}$, define $X_p := X'_p \times \{0,1\}^{(2^{|\mathcal{P}|}-1)}$;
- for each $(x, \mathbb{B}) \in X_p$, $\mathcal{D}_p((x, \mathbb{B})) := \mathcal{D}'_p(x)$;
- for $a \in \Sigma$ and $\pi \in X'_{\mathsf{dom}(a)}$, if $\delta'_a(\pi) = \pi'$ then define the new mapping $\delta_a\big(((\pi_{|p}, \mathbb{B}_p))_{p \in \mathsf{dom}(a)}\big) := \big((\pi'_{|p}, \mathbb{B}'_p)\big)_{p \in \mathsf{dom}(a)}$, where the new bit-vector is assigned[9] as $\mathbb{B}'_p[Q] := \begin{cases} 1 - \mathbb{B}_p[Q] & \text{if } Q = \mathcal{D}'_p(\pi_{|p}) \\ \mathbb{B}_p[Q] & \text{otherwise} \end{cases}$;
- if π_0 is the initial state of $\mathfrak{B}$, then the initial state of $\mathfrak{A}$ is $((\pi_{0|p}, \mathbb{B}_0))_{p \in \mathcal{P}}$, where $\mathbb{B}_0 = (0, \ldots 0)$.

Every time a process p moves out of a local p-state $(x, \mathbb{B})$ with $\mathcal{D}'_p(x) = Q$ and arrives in a new local state $(y, \mathbb{B}')$, it ensures that $\mathbb{B}[Q] \neq \mathbb{B}'[Q]$. By this construction, during a run ρ of $\mathfrak{A}$, a process p halts after finitely many transitions

[9]Since the number of bits in a bit-vector $\mathbb{B}$ is equal to the number of non-empty subsets of $\mathcal{P}$, we refer to the bit corresponding to a subset Q as $\mathbb{B}[Q]$.

if and only if $\lceil \mathsf{Inf}_p(\rho) \rceil$ is a singleton. In particular, if a processes p loops infinitely often on the same state x in $\mathfrak{B}$, then in $\mathfrak{A}$ it will alternate between $(x, \mathbb{B})$ and $(x, \mathbb{B}')$ ad infinitum where $\mathbb{B}$ and $\mathbb{B}'$ must differ at index $\mathcal{D}'_p(x)$. This immediately provides a mechanism for defining the Muller acceptance table $\mathcal{F}$ of $\mathfrak{A}$.

For each acceptance pair $(Q, F') \in \mathcal{F}'$ of $\mathfrak{B}$, we define a number of Muller acceptance tuples $F \in \mathcal{F}$ for $\mathfrak{A}$ such that

- for each $p \notin Q$, $F^p = \{(x, \mathbb{B})\}$ for some $x \in F'^p$ and $\mathbb{B} \in \{0,1\}^{(2^{|\mathcal{P}|}-1)}$; and
- for each $p \in Q$, $|F^p| \geq 2$ and there exists a state $(x, \mathbb{B}) \in F^p$ such that $x \in F'^p$, and there exists at least one pair of states $(y, \mathbb{B}_1), (z, \mathbb{B}_2) \in F^p$ such that $\mathbb{B}_1$ and $\mathbb{B}_2$ differ (at the least) at index $\mathcal{D}'_p(x)$.

Clearly, the acceptance table $\mathcal{F}$ is closed under supersets modulo singletons, that is, it is closed under supersets modulo finitary acceptance sets.

Now it is trivial to show that $L(\mathfrak{A}) = L(\mathfrak{B})$. ■

As mentioned previously, every ω-regular trace language can be written as a finite Boolean combination of A-infinitary limit languages [DM94]. Our results allow us to state an equivalent claim by referring to classes of automata.

Theorem 3.37 *For any language $\Theta \subseteq \mathbb{R}(\Sigma, I)$ of infinite traces, Θ is D-SAMA recognizable if and only if Θ can be expressed as a finite Boolean combination of D-SABA recognizable languages.*

Proof One direction of this theorem follows trivially from the facts that for each D-SABA there exists a D-SAMA (cf. Theorem 3.36) and that the family of D-SAMAs is closed under finite Boolean operations (cf. Proposition 3.34).

We only need to show how a language recognized by a D-SAMA $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$ can be expressed as a finite Boolean combination of D-SABA recognizable languages. For any $F_i \in \mathcal{F}$, with $F_i = (F_i^p)_{p \in \mathcal{P}}$,

- let $Y_i^p \subseteq X_p$ consist of all the p-states y such that for $x \in F_i^p$, $\mathcal{D}_p(x) = \mathcal{D}_p(y)$;
- let $\Pi_i \subseteq X_{\mathcal{P}}$ be a set of global states satisfying $\forall p \in \mathcal{P}\colon\ \bigcup_{\pi \in \Pi_i} \pi_{|p} = F_i^p$;
- let $Q_i \subseteq \mathcal{P}$ consist of all processes p such that $|F_i^p| = 1$.

For each $\pi \in \Pi_i$ define a D-SABA $\mathfrak{A}_{i,\pi} := (\mathfrak{T}, \mathcal{D}, \mathcal{F}_{i,\pi})$ whose acceptance table $\mathcal{F}_{i,\pi} := \bigcup_{R \subseteq Q_i} \{(\mathcal{P} \setminus R, (\{\pi_{|p}\})_{p \in \mathcal{P}})\}$. If a trace θ is accepted by $\mathfrak{A}_{i,\pi}$ then there exists some $R \subseteq Q_i$ that processes $q \notin R$ visit local q-states $\pi_{|q}$ infinitely often process $q \in R$ eventually stall at states $\pi_{|q}$. It is nowhere required that the global state π ever occurs. We only use π as an easy reference to local states that are drawn from the Muller acceptance tuple F_i.

Referring to languages $L(\mathfrak{A}_{i,\pi})$, we define the language $L_i^+ := \bigcap_{\pi \in \Pi_i} L(\mathfrak{A}_{i,\pi})$ of traces inducing runs that, for some $R \subseteq Q_i$, result in processes $p \in R$ halting in

p-states $x \in F_i^p$, and processes $p \notin R$ visiting all the states (and maybe more) from sets F_i^p.

Next, we define languages that will prevent the processes from visiting any more than their respective acceptance sets F_i^p. For each $p \in \mathcal{P}$, we define a D-SABA $\mathfrak{A}_{i,p} := (\mathfrak{T}, \mathcal{D}, \mathcal{F}_{i,p})$ with $\mathcal{F}_{i,p} := \bigcup_{Q \subseteq \mathcal{P}} \{(Q, (F_{i,p}^q)_{q \in \mathcal{P}})\}$ where the sets $F_{i,p}^q := \begin{cases} Y_i^q \setminus F_i^q & \text{if } q = p \\ Y_i^q & \text{otherwise} \end{cases}$. The language $L(\mathfrak{A}_{i,p})$ consists of traces which, irrespective of the set Q of live processes, ensure that either p halts in a state outside of F_i^p (i.e., if $p \notin Q$) or p infinitely often visits states outside of F_i^p (i.e., if $p \in Q$). In this sense, $L(\mathfrak{A}_{i,p})$ consists of all traces on whose runs at least process p "misbehaves". Thus, the language $L_i^- := \bigcup_{p \in \mathcal{P}} L(\mathfrak{A}_{i,p})$, comprises of traces on whose runs at least one process misbehaves.

Finally, we can express $L(\mathfrak{A}) = \bigcup_{F_i \in \mathcal{F}} \left(L_i^+ \cap \overline{L_i^-} \right)$. ■

3.7 Discussion

We introduced synchronization aware 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. In this sense, we provide an automata-theoretic justification to the term *deterministic trace languages* which was introduced in [Mus94]. Henceforth, these may equivalently 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.

At the same time, if the independence relation is empty then every transition system over words is a special case of a synchronization aware transition system over traces. This is because the transition systems over words consist of only one process, which belongs to the domain of every letter. Therefore we can associate a trivial mapping $\mathcal{D}$ with such a transition system which maps every local state to this unique process.

The procedure that we present in Section 3.3 to construct an SATS $\overline{\mathfrak{T}}$ 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 to work in combination with Zielonka's original construction, and apply it to "I-diamond" finite state automata to directly obtain an SATS without an intermediate ATS.

We also consider it important to explore a definition of synchronization aware cellular automata, where, although a set of processes synchronize on a letter, exactly one of them changes its state. With the hitherto existing definitions, it is known that asynchronous cellular automata are equivalent to asynchronous automata for the case of languages of finite traces. But it is not clear whether a family of suitably defined “synchronization aware cellular transition systems” can also be shown to be equivalent to the family of SATS.

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 synchronization aware weak automata; but it remains to be seen if they characterize the family of finite Boolean combinations of reachability and safety languages. Although the formal definitions of reachability and safety languages are straightforward, we conjecture that it is not possible to characterize these languages in terms of any family of asynchronous automata. One reason lies in the distributed nature of acceptance conditions of asynchronous automata. Any definition of safety, reachability, or weak asynchronous automata must refer to the states that are locally acquired by the processes and infer the global states acquired by the automaton.

This is quite different from the results shown in this chapter (see Section 3.4), where we infer the global states acquired infinitely often by an ATS by referring to the maximal yields witnessed infinitely often by individual processes. It is possible to answer whether or not a global state occurs infinitely often even after ignoring arbitrarily long run prefixes. On the other hand, it is much more difficult to answer whether or not a global state occurs at least once.

However, we would still like to settle the open question whether finite Boolean combinations of reachability and safety trace languages are exactly those that lie in the intersection of D-SABA and D-SAcBA recognizable trace languages. This is quite different from answering whether or not there exists a single automaton model characterizing these languages.

Chapter 4

Trace-closed Languages and I-diamond Automata

In this chapter, we look once again at the problem of obtaining a Borel-like classification of recognizable ω-trace languages. Here, we provide a solution in the domain of trace-closed languages of infinite words, which is invoked via the morphism Γ^{-1} (see definition in Section 2.1) yielding sets of linearizations of infinitary trace languages.

Recall from Chapter 1 that recognizable ω-languages can be obtained by various operations from recognizable languages of finite words. In general, any recognizable ω-language L can be represented as a finite union $\bigcup_i K_i \cdot K_i'^{\omega}$, with K_i, K_i' recognizable. There are also notions of subclasses of recognizable ω-languages that are obtained from given recognizable languages K in the following ways:

- $\mathsf{ext}(K) = \{\alpha \in \Sigma^\omega \mid \alpha \text{ has a prefix in } K\}$
- $\mathsf{lim}(K) = \{\alpha \in \Sigma^\omega \mid \alpha \text{ has infinitely many prefixes in } K\}$

For recognizable languages K, languages $\mathsf{lim}(K)$ can be characterized in a straightforward manner as languages recognized by deterministic Büchi automata, and a result due to Landweber states that it is decidable whether a given recognizable ω-language is deterministically Büchi recognizable [PP04, Chapter 1]. The same is true for languages $\mathsf{ext}(K)$, which are recognized by E-automata[1] (reachability automata). Finite Boolean combinations of languages $\mathsf{ext}(K)$ yield the family of *weakly recognizable* languages, and this class can be characterized in terms of deterministic weak automata (DWAs) [Sta83]. Finite Boolean combinations of languages $\mathsf{lim}(K)$ result in all recognizable ω-languages [PP04].

However, analogous results for deterministic Büchi recognizable ω-trace languages have only recently been established in terms of "synchronization-aware" automata in the previous chapter. Automata theoretic exploration of languages $\mathsf{ext}(T)$, for recognizable trace languages T, is still open. And while asynchronous automata are useful in implementing distributed monitors and distributed controllers, their constructions are prohibitively expensive even by automata-theoretic standards. On the other hand, for applications like model-checking and formal

[1]An E-automaton $\mathfrak{A} = (\mathfrak{T}, F)$ accepts a word $\alpha \in \Sigma^\omega$ if for the run ρ of $\mathfrak{A}$ on α: $\mathsf{Occ}(\rho) \cap F \neq \emptyset$.

verification, word automata recognizing trace-closed languages would already allow for analysis of most of the interesting properties pertaining to distributed computations.

Therefore, in this chapter, we build upon the theory of trace-closed languages of words (see Chapter 2 for quick recall) and study classes of recognizable ω-languages of words which allow us to "transfer" interesting results to the corresponding classes of recognizable ω-trace languages. In particular, motivated by the Borel hierarchy for regular languages of infinite words, our main contribution is a new setup for a classification theory for recognizable ω-trace languages in terms of trace-closed, recognizable ω-languages of words.

As illustrated in Figure 4.1a, for a trace-closed word language $K = \Gamma^{-1}(T)$, we firstly show that K can be modified to K_I such that $\mathsf{ext}(K_I)$ is also trace-closed and corresponds to the linearization of $\mathsf{ext}(T)$ (here I denotes the independence relation over the alphabet Σ). Building on this, we are able to characterize the class of Boolean combinations of languages $\mathsf{ext}(T)$ as precisely those whose linearizations are recognized by the class of "I-diamond" deterministic weak automata (DWAs).

Next, we consider infinitary limits. Here, the situation is different, in that there exist regular trace languages T such that although the trace-closed word language L corresponding to $\mathsf{lim}(T)$ is recognizable, it is not recognized by any I-diamond deterministic Büchi automaton (DBA). We therefore introduce the class of *limit-stable* word languages K – and by extension limit-stable trace languages T – such that the correspondence of Figure 4.1b holds, and $\mathsf{lim}(K)$ can be characterized in terms of I-diamond DBA.

$$\begin{tikzcd}
T \arrow[rr] \arrow[d, leftrightarrow] & & \mathsf{ext}(T) \arrow[d, leftrightarrow] \\
K \arrow[r] & K_I \arrow[r] & \mathsf{ext}(K)
\end{tikzcd}$$

(a) The correspondence of infinitary extensions for all rec. trace languages T.

$$\begin{tikzcd}
T \arrow[r] \arrow[d, leftrightarrow] & \mathsf{lim}(T) \arrow[d, leftrightarrow] \\
K \arrow[r] & \mathsf{lim}(K)
\end{tikzcd}$$

(b) The correspondence of infinitary limits for all *limit-stable* trace languages T.

Figure 4.1: From rec. trace-closed languages to rec., trace-closed ω-languages.

Departing from our previous discussions using only asynchronous transition automata, we analyze languages in terms of asynchronous cellular transition automata. Contributing to previously known algebraic results for recognizable ω-trace languages [GP92a, DM94, Mus94, DR95], we also make a short digression into algebra for results pertaining to the limits of limit-stable languages. We present our results after briefly revisiting these definitions in the next section.

4.1 Cellular Automata and Elementary Algebra

We denote the classes of recognizable languages of finite and infinite traces with $\mathsf{Rec}(\mathbb{M}(\Sigma, I))$ and $\mathsf{Rec}(\mathbb{R}(\Sigma, I))$ respectively. And for classes $\mathcal{K}$ of recognizable languages of finite words, we refer naturally to classes $\mathsf{ext}(\mathcal{K})$ and $\lim(\mathcal{K})$ of infinitary extensions and infinitary limits of all languages $K \in \mathcal{K}$. Once again highlighting the close relationship between trace languages and trace-closed languages, we present an alternative definition of recognizable trace languages.

Definition 4.1 *A trace language $T \subseteq \mathbb{M}(\Sigma, I)$ (respectively $\Theta \subseteq \mathbb{R}(\Sigma, I)$) is called a* recognizable trace language *iff $\Gamma^{-1}(T)$ (respectively $\Gamma^{-1}(\Theta)$) is a recognizable language of words.*

Asynchronous cellular automata have been introduced [DM94, GP92a] as acceptors of recognizable ω-trace languages. However, a global view of their (local) transition relations yields a notion of automata that recognize trace-closed word languages. Throughout this section, we take this global view of asynchronous automata. Formally, a *deterministic asynchronous cellular automaton (DACA)* over (Σ, I) is a 4-tuple $\mathfrak{A} = (\prod_{a\in\Sigma} Q_a, (\delta_a)_{a\in\Sigma}, q_0, F)$, consisting of sets Q_a of *local states* for each letter $a \in \Sigma$, and where $q_0 \in \prod_{a\in\Sigma} Q_a$, $\delta_a\colon \prod_{b\in D_a} Q_b \to Q_a$ and $F \subseteq \prod_{a\in\Sigma} Q_a$. Given a state $q \in \prod_{a\in\Sigma} Q_a$ and a letter $b \in \Sigma$, the unique b-successor $\delta(q, b) = q' = (q'_a)_{a\in\Sigma} \in \prod_{a\in\Sigma} Q_a$ is given by $q'_b = \delta_b((q_a)_{a\in D_b})$ and $q'_a = q_a$ for all $a \neq b$. That is, the only component that changes its state is the component corresponding to b. Given a word $u \in \Sigma^*$ the *run* ρ_u of $\mathfrak{A}$ on u is a sequence of states given as usual by $\rho_u(0) = q_0$ and $\rho_u(i+1) = \delta(\rho_u(i), u[i])$. This definition extends naturally to infinite runs ρ_α on infinite $\alpha \in \Sigma^\omega$. Define $\mathsf{Occ}_a(\rho)$ of (a finite or an infinite) run ρ to be the set $\{\rho(0)_a, \rho(1)_a, \ldots\} \subseteq Q_a$. Likewise, $\mathsf{Inf}_a(\rho) = \{q \in Q_a \mid \exists^\infty n\colon \rho(n)_a = q\}$.

A *deterministic asynchronous cellular Muller automaton* [DM94] (a *DACMA*) is an asynchronous automaton $\mathfrak{A} = (\prod_{a\in\Sigma} Q_a, (\delta_a)_{a\in\Sigma}, q_0, \mathcal{F})$ with $\mathcal{F} \subseteq \prod_{a\in\Sigma} 2^{Q_a}$. A DACMA *accepts* $\alpha \in \Sigma^\omega$ if for some $F = (F_a)_{a\in\Sigma} \in \mathcal{F}$ and all $a \in \Sigma$ we have $\mathsf{Inf}_a(\rho_\alpha) = F_a$. A *deterministic asynchronous cellular Büchi automaton* (a *DACBA*) is a tuple $\mathfrak{A} = (\prod_{a\in\Sigma} Q_a, (\delta_a)_{a\in\Sigma}, q_0, \mathcal{F})$, $F \subseteq \prod_{a\in\Sigma} 2^{Q_a}$. A DACBA *accepts* $\alpha \in \Sigma^\omega$ if for some $F = (F_a)_{a\in\Sigma} \in \mathcal{F}$ and all $a \in \Sigma$ we have $F_a \subseteq \inf_a(\rho_\alpha)$.

As was the case with the class of DAMA and DABA, while it is known that the class of DACMAs characterize precisely the class of recognizable ω-trace languages [DM94], no such correspondence is known for the class of languages recognized by DACBAs [Mus94].

Every $T \in \mathsf{Rec}(\mathbb{M}(\Sigma, I))$ (respectively $\Theta \in \mathsf{Rec}(\mathbb{R}(\Sigma, I))$) is recognized by a DACA [DR95] (respectively a DACMA). Via their global behaviors, asynchronous automata accept the corresponding trace-closed languages, and in particular, every recognizable trace-closed language (resp. trace-closed recognizable ω-language) is recognized by an I-diamond DFA (resp. I-diamond Muller automaton). In fact for every trace-closed $K \in \mathsf{REG}$, the minimal DFA $\mathfrak{A}_K$ accepting K is I-diamond. Recall that a word automaton $\mathfrak{A} = (Q, \Sigma, q_0, \delta)$ is called *I-diamond* if for every $(a, b) \in I$ and every state $q \in Q$, $\delta(q, ab) = \delta(q, ba)$.

Finally, we present some basic algebraic definitions (cf. [CPP08] for a detailed exposition). Given a language T of finite traces, a semigroup S, and a morphism

$\varphi\colon \mathbb{M}(\Sigma, I) \to S$, φ is said to *recognize* T if there exists $P \subseteq S$ with $T = \varphi^{-1}(P)$. By extension, S is said to recognize T if such a morphism exists. A *linked pair* of a semigroup is a tuple $(s, e) \in S^2$ with $s \cdot e = s$ and $e \cdot e = e$. We state a well known consequence of Ramsey's theorem: Let A be a (possibly infinite) alphabet, S be any finite semigroup and $f\colon A^+ \to S$ any mapping. Given an infinite sequence $\alpha \in A^\omega$ and an arbitrary factorization $\alpha = (u_i)_i$ of α into words $u_i \in A^+$, there exists a linked pair (s, e) and a strictly monotone sequence $(n_i)_i$ of natural numbers with the property that $f(u_0 \cdots u_{n_0}) = s$ and $f(u_{n_i} \cdots u_{n_{i+1}-1}) = e$ for all $i \in \mathbb{N}$. Let $(u'_i)_i$ be given by $u'_0 = u_0 \cdots u_{n_0}$ and $u'_i = u_{n_i} \cdots u_{n_{i+1}-1}$ for $i \geq 1$. We say this *superfactorization* is *associated* with (s, e). We will often use Ramsey's theorem implicitly. Given a semigroup S, a morphism $\varphi\colon \mathbb{M}(\Sigma, I) \to S$ is said to *saturate* $\Theta \subseteq \mathbb{R}(\Sigma, I)$ if for every linked pair (s, e) of S we have either $\varphi^{-1}(s) \odot (\varphi^{-1}(e))^\omega \cap \Theta = \emptyset$ or $\varphi^{-1}(s) \odot (\varphi^{-1}(e))^\omega \subseteq \Theta$. Let Θ be a language of infinite traces, S be a finite semigroup, and $\varphi\colon \mathbb{M}(\Sigma, I) \to S$ a saturating morphism. Then, φ *recognizes* Θ if for some set P of linked pairs of S we have $\Theta = \bigcup_{(s,e) \in P} \varphi^{-1}(s) \odot (\varphi^{-1}(e))^\omega$. Again, we say S recognizes Θ if such a morphism exists. These notions of recognizability coincide with the corresponding notions from Definition 4.1.

4.2 Infinitary Extensions of Trace-Closed Languages

We wish to extend the well-studied relations between recognizable and recognizable ω-languages to trace languages. We first look at reachability languages and their Boolean combinations, i.e. the weakly recognizable languages, and study how they can be obtained as a result of infinitary operations on recognizable trace languages. In the classification hierarchy of recognizable ω-languages, reachability and safety languages occupy the lowest levels. For trace languages, recalling Definition 2.13, we have that the infinitary extension of $T \subseteq \mathbb{M}(\Sigma, I)$ is the ω-trace language given by $\mathsf{ext}(T) \coloneqq T \odot \mathbb{R}(\Sigma, I)$.

Extrapolating the definition of E-automata for word languages, we define E-automata for trace languages where a run is accepting if for each $a \in \Sigma$ some predefined local states from Q_a are reached. Formally, a *deterministic asynchronous E-automaton* (a *DAEA*) is a tuple $\mathfrak{A} = (\prod_{a \in \Sigma} Q_a, (\delta_a)_{a \in \Sigma}, q_0, \mathcal{F})$ with $\mathcal{F} \subseteq \prod_{a \in \Sigma} 2^{Q_a}$. The DAEA $\mathfrak{A}$ accepts $\alpha \in \Sigma^\omega$ if for some $F = (F_a)_{a \in \Sigma} \in \mathcal{F}$ we have that $\mathsf{Occ}_a(\rho_\alpha) \cap F_a \neq \emptyset$. Note that given a DACA $\mathfrak{B}$ with $L(\mathfrak{B}) = T$, in order to accept $\mathsf{ext}(T)$, any DAEA $\mathfrak{A}$ must infer the "global-state reachability" of $\mathfrak{B}$ by referring only to "local-state reachability" in $\mathfrak{A}$. A simple counterexample suffices to show that this is a difficult task.

Proposition 4.2 *There exist languages* $T \subseteq \mathsf{Rec}(\mathbb{M}(\Sigma, I))$ *such that* $\mathsf{ext}(T)$ *is not recognized by any DAEA.*

Proof Let (Σ, I) be a independence alphabet with $\Sigma = \{a, b, c\}$ and bIc. Consider the trace language $T \coloneqq \{t \in \mathbb{M}(\Sigma, I) \mid$ there exist at least two concurrent events $e_1, e_2 \in t\colon \lambda(e_1) = b \wedge \lambda(e_2) = c\}$, containing all finite traces where b and c appear concurrently. Let us assume that there exists a DAEA $\mathfrak{A} = (\prod_{a\in\Sigma} Q_a, (\delta_a)_{a\in\Sigma}, q_0, \mathcal{F})$ recognizing $\mathsf{ext}(T)$.

Clearly, $\Gamma(ab^\omega c^\omega) \in \mathsf{ext}(T)$ must be accepted by $\mathfrak{A}$, and let us say by referring to $F^1 = (F_a^1)_{a\in\Sigma}$. Then there must exist a finite prefix $u_1 = ab^{n_1} \sqsubset ab^\omega c^\omega$ such that the run ρ_1 of $\mathfrak{A}$ over $\Gamma(u_1)$ ends in local states $q_a^1 \in F_a^1$ and $q_b^1 \in F_b^1$. Then again $\Gamma(u_1 \cdot ab^\omega c^\omega) \in \mathsf{ext}(T)$ must be accepted by $\mathfrak{A}$ be referring to some tuple $F^2 = (F_a^2)_{a\in\Sigma}$. Now there must exist a prefix $u_2 = u_1 \cdot ab^{n_2}$ such that the run ρ_2 of $\mathfrak{A}$ over $\Gamma(u_2)$ ends in some local states $q_b^2 \in F_b^2$ and $q_b^2 \in F_a^2$. In this manner we can construct a number of prefixes u_i, such that $u_i = u_{i-1} \cdot ab^{n_i}$ forces $\mathfrak{A}$ to arrive at local states $q_a^i \in F_a^i$ and $q_b^i \in F_b^i$, and at the same time $\Gamma(u_{i-1} \cdot ab^\omega c^\omega)$ is accepted by $\mathfrak{A}$ by referring to F_i.

Since $\mathcal{F}$ is finite, there must exist a smallest $j \in \mathbb{N}$ such that the trace $\Gamma(u_{j-1} \cdot ab^\omega c^\omega)$ is accepted by referring to F^j and for some $j' \lneq j, F^j = F^{j'}$. In particular, the run on $\Gamma(u_{j-1})$ has already witnessed local states $q_a^j \in F_a^j$ and $q_b^j \in F_b^j$. By independence of letters b and c, it follows that over the trace $\Gamma(u_{j-1} \cdot ac^\omega)$, $\mathfrak{A}$ will visit the local c-state $q_c^j \in F_c^j$ as well. Therefore, $\Gamma(u_{j-1} \cdot ac^\omega) \notin \mathsf{ext}(T)$ is accepted by $\mathfrak{A}$, which contradicts our assumption that $\mathfrak{A}$ recognizes $\mathsf{ext}(T)$. ■

A similar argument can be drawn against a possible definition of deterministic asynchronous weak cellular automata, defined in terms of SCCs that occur locally within Q_a for each $a \in \Sigma$. This means that the class of reachability languages resists characterization in terms of deterministic asynchronous automata. We therefore concentrate on the classes of I-diamond automata and trace-closed reachability languages in the hope of finding reasonable characterizations.

First, we note that the definition of infinitary extensions of a trace-closed languages is not sound with respect to trace equivalence of ω-words; that is, if $T \in \mathsf{Rec}(\mathbb{M}(\Sigma, I))$ and $K = \Gamma^{-1}(T)$, then, in general, $\mathsf{ext}(K) \neq \Gamma^{-1}(\mathsf{ext}(T))$.

Example 4.3 *Let $\Sigma = \{a, b, c\}$, and bIc. Define $K := [ab]_{\sim_I}$. Clearly K is trace-closed and, moreover, $acb \notin K$. Let $T = \Gamma(K)$. Clearly $abc^\omega, acbc^\omega, accbc^\omega, \ldots$ are equivalent words since they induce the same infinite trace which belongs to* $\mathsf{ext}(T)$*. However, while $abc^\omega \in \mathsf{ext}(K)$, $ac^+bc^\omega \nsubseteq \mathsf{ext}(K)$.*

Definition 4.4 *Let $K \subseteq \Sigma^*$ be trace-closed. Define the I-*suffix extended *trace-closed language (or I-*suffix extension*) of K as $K_I := K \cup \bigcup_{a\in\Sigma} [Ka^{-1}aI_a^*]_{\sim_I}$.*

Due to the closure of $\mathsf{Rec}(\mathbb{M}(\Sigma, I))$ under concatenation and finite union [DR95], we know that K_I is recognizable whenever K is recognizable.

Proposition 4.5 *For a language $T \in \mathsf{Rec}(\mathbb{M}(\Sigma, I))$, let $K = \Gamma^{-1}(T)$, and let K_I be the I-suffix extension of K. Then it holds that $\Gamma^{-1}(\mathsf{ext}(T)) = \mathsf{ext}(K_I)$.*

Proof From the definitions of K_I and T, we trivially observe that for every $\alpha \in \mathsf{ext}(K_I)$ it holds that $\Gamma(\alpha) \in \mathsf{ext}(T)$. Therefore, $\mathsf{ext}(K_I) \subseteq \Gamma^{-1}(\mathsf{ext}(T))$.

To show $\Gamma^{-1}(\mathsf{ext}(T)) \subseteq \mathsf{ext}(K_I)$, it is sufficient to show that: (1) for every infinite trace in $\theta \in \mathsf{ext}(T)$, there exists a linearization in $\alpha \in \mathsf{ext}(K_I)$ such that $\theta = \Gamma(\alpha)$; (2) the language $\mathsf{ext}(K_I)$ is trace-closed.
(1) Consider $\theta \in \mathsf{ext}(T)$. Hence, there exist $t \in T$ and $\theta' \in \mathbb{R}(\Sigma, I)$ such that $\theta = t \odot \theta'$. From the definitions, it follows that for any $w \in \Gamma^{-1}(t)$ and $\beta \in \Gamma^{-1}(\theta'), w \cdot \beta \in \mathsf{ext}(K)$ and therefore in $\mathsf{ext}(K_I)$.
(2) Let $\alpha \in \mathsf{ext}(K_I)$, and $t \in T$ be a trace such that $t \sqsubset \Gamma(\alpha)$. Consider any $\beta \in \Sigma^\omega$ such that $\beta \sim_I \alpha$. Trace equivalence implies that $t \sqsubset \Gamma(\beta)$. Moreover, there exists a minimal natural number $i \in \mathbb{N}, t \sqsubseteq \Gamma(\beta[1,i])$. Observe that $\beta[i]$ is a maximal symbol appearing in t because otherwise we can contradict the minimality of i and find $i' < i$ such that $t \sqsubseteq \Gamma(\beta[1,i'])$.

Now, let $s \in \mathbb{M}(\Sigma, I)$ be the finite trace such that $t \odot s = \Gamma(\beta[1,i])$. It must hold that either s is the empty trace or $\beta[i] \times \mathsf{alph}(s) \subseteq I$, because otherwise $t \odot s \neq \Gamma(\beta[1,i])$. This implies $\beta[1,i] \in K_I$, and hence $\beta \in \mathsf{ext}(K_I)$. ∎

Remark 4.6 *In general $K_I \neq (K_I)_I$. However, iterated I-suffix extensions preserve the infinitary extension languages, i.e.* $\mathsf{ext}(K) \subseteq \mathsf{ext}(K_I) = \mathsf{ext}((K_I)_I) = \mathsf{ext}(((K_I)_I)_I)\ldots$ *and so on.*

Proposition 4.5 provides us the basis for generating the class $\mathrm{BC}(\mathsf{ext}(\mathbb{M}(\Sigma, I)))$ of weakly recognizable trace-closed languages. Henceforth, whenever we speak of the language $\Gamma^{-1}(\mathsf{ext}(T))$ we refer to $\mathsf{ext}(\Gamma^{-1}(T)_I)$. Similarly, for a trace-closed language K we always mean $\mathsf{ext}(K_I)$ whenever we say $\mathsf{ext}(K)$.

Theorem 4.7 *A trace-closed language $L \subseteq \Sigma^\omega$ is recognized by an I-diamond DWA iff $L \in \mathrm{BC}(\mathsf{ext}(\mathcal{K}))$ for a set $\mathcal{K} \subseteq 2^{\Sigma^*}$ of trace-closed recognizable languages.*

Proof Given trace-closed recognizable languages $K \in \mathcal{K}$, we construct I-diamond DWA $\mathfrak{A}_K$ accepting $\mathsf{ext}(K)$ as mentioned previously. Let $L := \bigcup_i(\bigcap_j L_{i,j})$ be the language expressed in disjunctive normal form over $\mathsf{ext}(\mathcal{K})$ (for each i, j, $L_{i,j}$ is either of the form $\mathsf{ext}(K)$ or $\overline{\mathsf{ext}(K)}$). We define the product DWA $\mathfrak{A} := (\prod_{K\in\mathcal{K}} Q_K, \Sigma, (q_0^K)_{K\in\mathcal{K}}, \delta, F)$ where:

- $\delta((p^K)_{K\in\mathcal{K}}, a) = (q^K)_{K\in\mathcal{K}}$ if and only if $\delta_K(p^K, a) = q^K$ for all $K \in \mathcal{K}$.
- The tuple $(q^K)_{K\in\mathcal{K}} \in F$ if and only if it satisfies some conjunct. That is, for some i it holds that whenever $L_{i,j} = \mathsf{ext}(K)$ then $q^K = \bot_K$, and whenever $L_{i,j} = \overline{\mathsf{ext}(K)}$ then $q^K \neq \bot_K$ for all $K \in \mathcal{K}$.

The above product retains the structural I-diamond property; and since the family of DWAs is closed under finite Boolean combinations, it is easily verified that $\mathfrak{A}$ is an I-diamond DWA accepting L.

For the other direction, consider the minimal DWA $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$ that accepts L. Since trace equivalence $\sim_I$ over finite words is a finer congruence than the language congruence $\sim_L$ (i.e. $u \sim_I v \Rightarrow u \sim_L v$ for all $u, v \in \Sigma^*$), it follows that for any pair of finite trace equivalent words $u, v \in \Sigma^*, \delta(q_0, u) = \delta(q_0, v)$. Thus, $\mathfrak{A}$ is I-diamond.

For each SCC $S \subseteq Q$ of $\mathfrak{A}$, let $K_S \in \mathit{REG}$ be the trace-closed language accepted by the I-diamond DFA $\mathfrak{A}_S := (Q, \Sigma, q_0, \delta, S)$. Recall that each SCC of a DWA contains either only accepting states or rejecting states. Then, the language L accepted by $\mathfrak{A}$ is given by the following $\overline{\text{disjunction}}$ over all accepting SCC's $L := \bigcup_S L_S$, where $L_S := \mathsf{ext}(K_S) \cap \bigcap_{S' \neq S} \overline{\mathsf{ext}(K_{S'})}$. ■

4.3 Infinitary Limits of Trace-closed Languages

We now consider the *infinitary limit* operator. In the case of word languages, this operator extends recognizable languages to the family of recognizable ω-languages that are DBA recognizable. This is not straight forward for traces, and here we seek an effective characterization of languages $T \in \mathsf{Rec}(\mathbb{M}(\Sigma, I))$, such that $\Gamma^{-1}(\mathsf{lim}(T))$ is recognized by an I-diamond DBA. Recall Definition 2.14 describing infinitary limits $\mathsf{lim}(T), T \subseteq \mathbb{M}(\Sigma, I)$.

It is open whether there exists any characterization for the class of languages recognized by the family of DACBAs, however there do exist recognizable languages $T \subseteq \mathbb{M}(\Sigma, I)$ such that $\mathsf{lim}(T)$ is not recognized by any DACBA [Mus94]. In fact, even when relying on trace-closed word languages and I-diamond automata, we cannot hope to characterize these languages in the manner of infinitary extensions as demonstrated previously in Section 4.2.

Example 4.8 *Let $\Sigma = \{a, b\}$, and aIb. Define $K := [(aa)^+(bb)^+]_{\sim_I}$ as the trace-closed language with even number of occurrences of a's and b's. The minimal DFA accepting this language is shown in Figure 4.2. If $T = \Gamma(K)$, then*

$$\mathsf{lim}(T) = \Theta := \left\{ \theta \in \mathbb{R}(\Sigma, I) \;\middle|\; \begin{array}{l} |\theta|_a \text{ even}, |\theta|_b = \infty, \text{ or} \\ |\theta|_a = \infty, |\theta|_b \text{ even}, \text{ or} \\ |\theta|_a = |\theta|_b = \infty \end{array} \right\}$$

The trace-closed language $L = \Gamma^{-1}(\Theta)$ consists of all infinite words $\alpha \in \Sigma^\omega$ that satisfy the same conditions as $\theta \in \Theta$ above.

It is easy to verify that the DFA of Figure 4.2 does not accept L when equipped with a Büchi acceptance condition. For instance, over the word $\alpha = ab(aabb)^\omega$, the automaton can loop forever in states 4, 6, and 7, thereby witnessing infinitely many a's and b's, without ever visiting state 8.

Proposition 4.9 *There exists $\underline{\text{no}}$ I-diamond deterministic parity automaton, and therefore $\underline{\text{no}}$ I-diamond deterministic Büchi automaton, recognizing the trace-closed language $L \subseteq \Sigma^\omega$ as described in Example 4.8.*

Proof Firstly, verify that L is an trace-closed recognizable ω-language. The transition graph of Figure 4.2 can be equipped with Muller accepting conditions

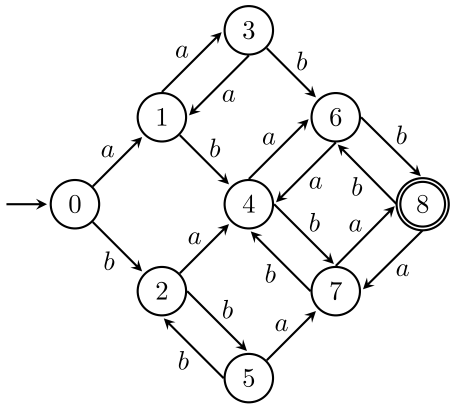


Figure 4.2: The minimal DFA recognizing language K of Example 4.8.

to recognize L, namely $\mathcal{F} := \{\{6,8\}, \{7,8\}, \{4,6,7\}, \{4,6,8\}, \{4,7,8\}, \{6,7,8\}, \{4,6,7,8\}\}$. Also note that since the Muller sets are closed under supersets, L is in fact recognized by some DBA.

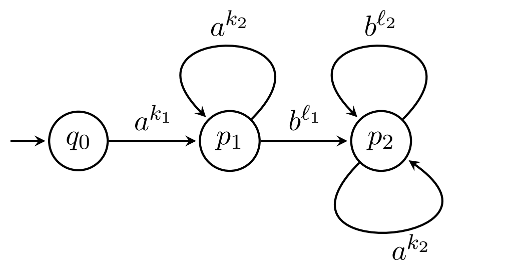
We present the proof for the claim with respect to DBAs. The proof with respect to deterministic parity automata can be obtained by an identical argument, by simply replacing the terms "Büchi accepting state" with terms "states with high even color" (assuming parity acceptance condition refers to highest even parity).

Now, let us assume that L is also recognized by some I-diamond DBA $\mathfrak{A}_L$. Let q_0 be the initial state and δ be the transition function of this automaton. Let $k_1 > 0, k_2 > 0$ be the smallest integers such that for any $n \in \mathbb{N}, \delta(q_0, a^{k_1}) = \delta(q_0, a^{k_1+n\cdot k_2}) = q_1$. Also, let $\ell_1 > 0, \ell_2 > 0$ be smallest integers such that for any $n \in \mathbb{N}, \delta(q_1, b^{\ell_1}) = \delta(q_1, b^{\ell_1+n\cdot\ell_2}) = q_2$. Owing to the I-diamond property of $\mathfrak{A}_L$, as shown in Figure 4.3a, $\delta(q_0, a^{k_1+k_2}b^{\ell_1}) = \delta(q_0, a^{k_1}b^{\ell_1}a^{k_2}) = p_2$.

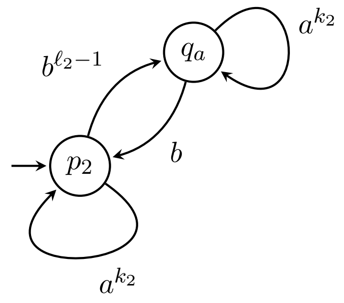
We claim that $k_1 + k_2$ must be even. On the contrary, let us assume that $k_1 + k_2$ is odd. Then there are two possibilities. If k_1 is even, then the word $a^{k_1}b^{\ell_1}b^{\omega} \in L$. Therefore, the ℓ_2-loop at state p_2 must contain at least one Büchi accepting state. But then, the word $a^{k_1+k_2}b^{\ell_1}b^{\omega}$ is also accepted, which is a contradiction.

On the other hand, if k_1 is odd, then since $k_1 + k_2$ is odd it follows that k_2 must be a non-zero even. Moreover, the ℓ_2-loop at state p_2 cannot contain any Büchi accepting states, because otherwise $a^{k_1}b^{\omega}$ will be accepted by the automaton. Now, ℓ_1 can be either odd or even. In the former case (ℓ_1 odd), it must hold that the k_2-loop at state p_2 must also not contain any Büchi accepting state since $a^{k_1}b^{\ell_1}a^{\omega} \notin L$ for ℓ_1 odd. But then $a^{k_1}b^{\ell_1}(a^{k_1}b^{\ell_1})^{\omega}$ is also rejected by the automaton, which is a contradiction. In the latter case (ℓ_1 even), the k_2-loop at p_2 must have a Büchi accepting state since $a^{k_1}b^{\ell_1}a^{\omega} \in L$. But then, ℓ_2 must be even as well because $a^{k_1}b^{\ell_1+\ell_2}a^{\omega}$ is accepted by the automaton (and ℓ_2 odd makes $\ell_1 + \ell_2$ odd as well). So $\ell_2 > 0$ even implies that $\ell_2 - 1 > 0$ odd.

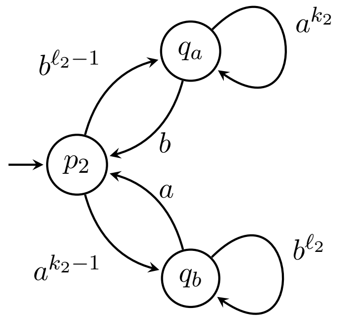
Now we look closely at the ℓ_2-loop at p_2, and refer to the state $q_a = \delta(p_2, b^{\ell_2-1})$ as shown in Figure 4.3b. Note that $\delta(p_2, a^{k_2}b^{\ell_2-1}) = q_a$, so by the I-diamond



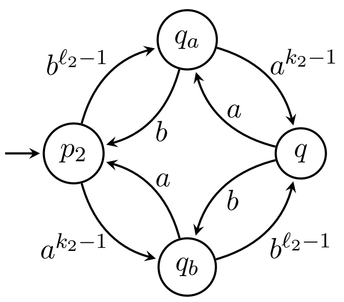
(a) Assumption: $k_1 > 0, k_2 > 0, \ell_1 > 0, \ell_2 > 0$



(b) Unfolding b^{ℓ_2}-loop at state p_2



(c) Unfolding both loops at p_2



(d) The loops must intersect

Figure 4.3: Behavior of any I-diamond DBA $\mathfrak{A}_L$.

property, q_a must have a loop on a^{k_2}. Also, the k_2-loop on q_a cannot have any Büchi accepting states because $a^{k_1}b^{\ell_1+\ell_2-1}a^\omega \notin L$. But then, $a^{k_1}b^{\ell_1}(b^{\ell_2-1}a^{k_2}b)^\omega$ is rejected by the automaton as well. This final contradiction establishes our claim that $k_1 + k_2$ must be even.

By symmetry of letters a and b, we also obtain that $\ell_1 + \ell_2$ must be even. Finally, starting from these necessities, we derive a contradiction toward proving that our original assumption of existence of $\mathfrak{A}_L$ is false.

Recalling the earlier arguments of this proof, we can immediately establish that both k_1 and ℓ_1 must be even. This implies that k_2 and ℓ_2 must also be even. Now, just as above, we also expand the k_2-loop at state p_2, and observe that there must exist an ℓ_2-loop at such a state q_b as shown in Figure 4.3c. And finally, we add one more level of detail, and observe that owing to the I-diamond structure $\delta(p_2, a^{k_2-1}b^{\ell_2-1}) = \delta(p_2, b^{\ell_2-1}a^{k_2-1}) = q$. Therefore, the k_2- and ℓ_2-loops at p_2 necessarily have a state q in common as shown in Figure 4.3d.

Since k_2 and ℓ_2 are even, $k_2 - 1$ and $\ell_2 - 1$ are odd. Hence, the loop $q_b \xrightarrow{b^{\ell_2-1}}$

$q \xrightarrow{b} q_b$ cannot contain any Büchi accepting state, because otherwise $a^{k_1}b^{\ell_1}a^{k_2-1}b^{\omega}$ will be accepted, but $k_1 + k_2 - 1$ is odd. By a symmetric argument, the loop $q_a \xrightarrow{a^{k_2-1}} q \xrightarrow{a} q_a$ cannot contain any Büchi accepting state. But then, if we restrict an infinite run to the loop $q_b \to q \to q_a \to q \to q_b \cdots$ then it contains infinitely many a's and infinitely many b's and is still rejected by the automaton. ■

The above result can be easily generalized as follows.

Corollary 4.10 *There exists a family $\mathcal{K}$ of recognizable trace-closed languages, namely $\mathcal{K} := \{[(a^m)^+(b^n)^+]_{\sim_I} \mid m, n \geq 2\}$ over $\Sigma = \{a, b\}$, such that given $T = \Gamma(K)$ for any $K \in \mathcal{K}$, there exists no I-diamond DBA recognizing $\Gamma^{-1}(\mathsf{lim}(T))$.*

We conclude that, in general, there is no hope of characterizing the class $\Gamma^{-1}(\mathsf{lim}(\mathsf{Rec}(\mathbb{M}(\Sigma, I))))$ trace-closed recognizable ω-languages in terms of deterministic I-diamond Büchi automata. Therefore, we explore subclasses of trace-closed recognizable ω-languages in our attempt to obtain a reasonable characterization.

Definition 4.11 *A trace-closed language $K \subseteq \Sigma^*$ is I-*limit-stable *(or simply* limit-stable*) if $\mathsf{lim}(K)$ is also trace-closed. By extension, $T \subseteq \mathbb{M}(\Sigma, I)$ is* limit-stable *if $\Gamma^{-1}(T)$ is.*

Toward characterizing limit-stable languages, we introduce some new definitions. Let $T \subseteq \mathbb{M}(\Sigma, I)$ be a language of traces and let $t \sqsubset t'$ be two traces. The *prefix graph* of the pair (t, t') is the directed, acyclic graph $G_{t,t'} = (V, E)$ with $V = \{x \in \mathbb{M}(\Sigma, I) \mid t \sqsubseteq x \sqsubseteq t'\}$ and $(x, y) \in E$ if $y = x \odot a$ for some $a \in \Sigma$. A *cut* of $G_{t,t'}$ is a set $C \subseteq V \setminus \{t, t'\}$ such that each path from t to t' in $G_{t,t'}$ visits at least one vertex from C. Note that if $t' = t \odot a$ for some $a \in \Sigma$, then $G_{t,t'}$ does not admit a cut. A pair (t, t') is *T-separable* if $G_{t,t'}$ admits a cut $C \subseteq T$.

Let $\theta \in \mathsf{lim}(T)$. Define an infinite transition-graph $G = G_\theta = (V, \Delta)$ with $V = \{t \in \mathbb{M}(\Sigma, I) \mid t \sqsubseteq \theta\}$ and $(t, a, t') \in \Delta$ if $t' = t \odot a$ for some $a \in \Sigma$. Then there is a one to one correspondence between the paths starting from ε through G and the linearizations of θ. More precisely, for any finite word $u \in \Sigma^*$, there exists a path ρ_u from ε on u in G_θ iff u is the linearization of some prefix t of θ. An infinite word α is a linearization of θ iff $\alpha[1, n]$ is a linearization of some prefix t_n of θ for all $n \in \mathbb{N}$. Hence, an ω-word α is a linearization of θ iff it induces a run ρ_α in G_θ.

Let S be a finite semigroup, let $P \subseteq S$, and let (s, e) be a linked pair of S. Let φ be a morphism from $\mathbb{M}(\Sigma, I)$ onto S. The pair (s, e) has the *P-cut property* if

- either for every factorization $\varphi(a_1) \cdots \varphi(a_k) = e$ with $a_i \in \Sigma$, we have $e\varphi(a_1 \odot \cdots \odot a_j) \in s^{-1}P$ for some $j \in [1, k]$;
- or for every factorization $\varphi(a_1) \cdots \varphi(a_k) = e$ with $a_i \in \Sigma$, we have $e\varphi(a_1 \odot \cdots \odot a_j) \notin s^{-1}P$ for all $j \in [1, k]$.

Lemma 4.12 *Let $T \in \mathsf{Rec}(\mathbb{M}(\Sigma, I))$. Then there exists a finite semigroup S and a saturating morphism $\alpha \colon \mathbb{M}(\Sigma, I) \to S$ which recognizes both $\mathsf{lim}(T)$ and T.*

Proof There exists a finite semigroup S' and a morphism φ' which recognizes $\lim(T)$. Furthermore, there exists a finite semigroup S'' and a morphism φ'' which recognizes T, say $T = \varphi''^{-1}(P'')$. Let $S := S' \times S''$ and $\varphi := \varphi' \times \varphi''$. Clearly, α recognizes T, taking $P \subseteq S$ as $S' \times P''$. Moreover, it saturates $\lim(T)$.

To see this, observe that every linked pair $((s', e'), (s'', e''))$ of S induces a linked pair (s', e') of S'. Now $\varphi^{-1}((x, y)) = \varphi'^{-1}(x) \cap \varphi''^{-1}(y) \subseteq \varphi'^{-1}(x)$. Letting $(s, e) = ((s', s''), (e', e'')) \in S$, this shows that $\varphi^{-1}(s) \odot \varphi^{-1}(e)^\omega \subseteq \Theta$ or $\varphi^{-1}(s) \odot \varphi^{-1}(e)^\omega \cap \Theta = \emptyset$.

It remains to show that there still exists a set of linked pairs (s, e) of S recognizing $\lim(T)$. To see this, pick any linked pair of S', say (s', e'). Then any trace θ associated with this pair admits a factorization $xy_1y_2\cdots$ with $\varphi(x) = s$ and $\varphi(y_i) = e$.

Now this factorization admits a superfactorization which is associated with a linked pair of S''. The claim now follows. ■

Such a morphism is said to *simultaneously recognize* T and $\mathsf{lim}(T)$. Given an automaton, we write $p \overset{u}{\to} q$ if some $u \in \Sigma^*$ leads from p to q, and $p \overset{u}{\Rightarrow} q$ if a final state is also visited.

Definition 4.13 *Given (Σ, I), let $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$ be an I-diamond automaton. $\mathfrak{A}$ is* F, I-cycle closed, *if for all $u \sim_I v$ and all q we have $q \overset{u}{\Rightarrow} q$ iff $q \overset{v}{\Rightarrow} q$.*

We can now give an effective characterization of limit-stable languages. Note that Lemma 4.12 ensures that the property (e) below is not trivially satisfied.

Theorem 4.14 *For any $T \in \mathsf{Rec}(\mathbb{M}(\Sigma, I))$ and $K = \Gamma^{-1}(T)$, the following are equivalent:*

(a) K, and therefore T, is limit-stable.

(b) For all sequences $(t_i)_{i\in\mathbb{N}} = t_0 \sqsubset t_1 \sqsubset t_2 \cdots \subseteq T$ and all sequences $(u_i)_{i\in\mathbb{N}}$ with $u_i \in \Gamma^{-1}(t_i)$, there exists a subsequence $(u_{j_i})_i$ and a sequence $(v_{j_i})_i$ of proper prefixes $v_{j_i} \sqsubset u_{j_i}$ with $|v_{j_i}| < |v_{j_{i+1}}|$ and $v_{j_i} \in K$ for all $i \in \mathbb{N}$.

(c) For any $\theta \in \mathsf{lim}(T)$ there exists a strictly monotone sequence $(n_i)_i$ such that any infinite path ρ in G_θ visits T in each segment $\rho(n_i, n_{i+1} - 1)$.

(d) Let $(t_i)_i$ be a sequence of traces in T. Then there exists a subsequence $(t_{m_i})_i$, such that $(t_{m_i}, t_{m_{i+1}})$ is T-separable for all i.

(e) If the languages T and $\mathsf{lim}(T)$ are simultaneously recognized by a morphism $\varphi\colon \mathbb{M}(\Sigma, I) \to S$ for some finite semigroup S, then every linked pair (s, e) has the $\varphi(T)$-cut property.

(f) Any DFA $\mathfrak{A}$ recognizing K is F, I-cycle closed.

Proof (a) $\Longrightarrow$ (b): If (b) is false, then we may choose a sequence $(t_i)_i$ of traces in T with the property that for some sequence $(u_i)_i$ of linearizations of $(t_i)_i$, every subsequence $(u_{n_i})_i$, and every sequence $(v_{n_i})_i$ of proper prefixes $v_{n_i} \sqsubset u_{n_i}$, $v_{n_i} \in K$, we have $\sup_i |v_{n_i}| < \infty$. Since $|\Sigma| < \infty$, we have that Σ^∞ is a compact space. Hence $(u_i)_i$ has a converging subsequence $(u_{m_i})_i$. Because every subsequence of $(u_i)_i$ has the properties given in the previous sentence, so does $(u_{m_i})_i$. Let $\alpha = \lim_{i\to\infty} u_{m_i}$. Then $\alpha \sim_I \beta$ for some $\beta = x \cdot y_1 \cdot y_2 \cdots$ with $x \cdot y_1 \cdots y_i \in \Gamma^{-1}(t_{m_i})$. Hence, $\beta \in \mathsf{lim}(L)$. But, by construction, $\alpha \notin \mathsf{lim}(K)$ because for some $n \in \mathbb{N}$ no prefix of length $> n$ of $alpha$ is in K.

(b) $\Longrightarrow$ (a): Let $\theta = \bigsqcup_i t_i$ for traces $t_i \in T$. We may assume that $t_i \sqsubset t \sqsubset t_{i+1}$ implies $t \notin T$. Let $\alpha \in \Gamma^{-1}(\theta)$. Then we pick prefixes $(w_i)_i$ of α, such that w_i is of minimal length with $t_i \sqsubseteq \Gamma(w_i)$. Consider the subsequence $(t_{2i})_i$ of $(t_i)_i$. Each w_{2i+1} is a prefix of some linearization of $t_{2(i+1)}$, say $u_{2(i+1)}$. We apply (b) to the sequence $(t_{2i})_i$ and get a sequence $(v_{2i})_i$ of proper prefixes of the u_{2i}, such that $\sup_i |v_{2i}| = \infty$ and $v_{2i} \in K$. We now have to show that v_{2i} is already a prefix of w_{2i-1}. Suppose not, i.e. $w_{2i-1} \sqsubset v_{2i} \sqsubset u_{2i}$. Then this would give a trace $t \in T$ with $t_{2i-1} \sqsubset t \sqsubset t_{2i}$.

(a) $\Longrightarrow$ (f): Suppose $\mathfrak{A}$ is not I-cycle closed. Then there exists $q \in Q$ and $u \sim_I v$ with $q \overset{u}{\Rrightarrow} q$ but not $q \overset{v}{\Rrightarrow} q$. Since $\mathfrak{A}$ is I-diamond, this means that the run $q \xrightarrow{v} q$ exists, but does not visit a final state. Now pick $x \in \Sigma^*$ with $q_0 \xrightarrow{x} q$. Then $\alpha = x \cdot u^\omega \in \mathsf{lim}(K)$ and $\beta = x \cdot v^\omega \notin \mathsf{lim}(L)$. But clearly $\alpha \sim_I \beta$ implies that $\mathsf{lim}(K)$ is not trace-closed.

(f) $\Longrightarrow$ (a): Let $\alpha \sim_I \beta$ and let $\alpha \in \mathsf{lim}(K)$. Take $\mathfrak{A} = \mathfrak{A}_K$ and consider extended transition profiles $\tau_w \subseteq Q \times \{0,1\} \times Q$ for $w \in \Sigma^*$ defined by $(p,1,q) \in \tau_w$ iff $p \overset{w}{\Rrightarrow} q$ and $(p,0,q) \in \tau_w$ iff $p \xrightarrow{w} q$ but not $p \overset{w}{\Rrightarrow} q$. Then we can factorize $\alpha = uv_0v_1v_2\cdots$ for finite words $u, v_0, v_1, \ldots$ with $\tau_u \cdot \tau_{v_i} = \tau_u$ and $\tau_{v_i} \cdot \tau_{v_i} = \tau_{v_i}$. Likewise, we can factorize $\beta = u'v_0'v_1'\cdots$.

Next, find $r \in \mathbb{N}$ with $\Gamma(u'v_0') \sqsubseteq \Gamma(uv_0\cdots v_r)$. This gives $x \in \Sigma^*$ with $u'v_0' \cdot x \sim_I uv_0\cdots v_r$. Conversely, there exists $m \in \mathbb{N}$ with $\Gamma(uv_0\cdots v_{r+1}) \sqsubseteq \Gamma(u'v_0'\cdots v_m')$ and therefore there exists $y \in \Sigma^*$ with $u'v_0'\cdots v_m' \sim_I uv_0\cdots v_r v_{r+1} y \sim_I u'v_0' x v_{r+1} y$, which implies $x v_{r+1} y \sim_I v_1' \cdots v_m'$.

Notice that if $q_0 \xrightarrow{u} q$ and $q_0 \xrightarrow{u'} q'$, then (by trace equivalence and the fact that $\mathfrak{A}$ is I-diamond) we have $q' \xrightarrow{x} q$. Likewise, we have $q \xrightarrow{y} q'$ and $q' \xrightarrow{xv_{r+1}y} q'$. Now we can apply (f) to see that $q' \xRightarrow{xv_{r+1}y} q'$ iff $q' \xRightarrow{v_1'\cdots v_m'} q'$. However, since $\alpha \in \mathsf{lim}(K)$, since $\tau_{v_{r+1}} = \tau_{v_i}$ for all i, and since $q \xRightarrow{v_{r+1}} q$, we have $q' \xRightarrow{xv_{r+1}y} q'$. Hence, $q' \xRightarrow{v_1'\cdots v_m'} q'$. Since furthermore $\tau_{v_1'\cdots v_m'} = \tau_{v_i'}$, we have for all $i, q' \underset{F}{\overset{v_i'}{\Rightarrow}} q'$ whence $\beta \in \mathsf{lim}(K)$.

(a) $\Longrightarrow$ (c): Let $\theta \in \lim(T)$. If for every $n \in \mathbb{N}$ there exists a run ρ_n through G_θ that visits a trace $t \in T$ only after n positions, then there exists a run through G_θ which never visits a trace in T. This is because $(\rho_n)_n$ admits a converging subsequence (the space is compact) and because the set $[G_\theta]$ of all paths is closed and so this limit must itself be a path through G_θ. This contradicts (a). Hence there exists n_0, such that every path through G_θ visits T after at most n_0 steps. We now consider all finite segments of length n_0 and extend them. Let U be the set of all those segments. Let $u \in U$. By a similar argument as before, there exists a number n_u, such that every extension $v = ux$ of length n_1 has visited T at least once after u. Since there are finitely many segments in U, we can take the maximum $n_1 = \max_{u \in U} n_u$. In this way, we construct $(n_i)_i$.

(c) $\implies$ (d): Given $(t_i)_i \subseteq T$ we let $\theta = \bigsqcup_i t_i$ and pick $(n_i)_i$ as in (c). Now pick m_0 arbitrarily; and, given m_i, pick m_{i+1}, such that $|t_{m_{i+1}}| > \min\{n_{j+1} \mid |t_{m_i}| < n_j\}$. Now consider $(t_{m_i}, t_{m_{i+1}})$. Because there exists n_j with $|t_{m_i}| < n_j < n_{j+1} < |t_{m_{i+1}}|$, we have that every path from t_{m_i} to $t_{m_{i+1}}$ visits T at least once. Hence, $(t_{m_i}, t_{m_{i+1}})$ is T-separable.

(d) $\implies$ (e): Let φ and S be as in the statement. Let (s, e) be a linked pair. If $\lim(T) \cap \varphi^{-1}(s) \odot (\varphi^{-1}(e))^\omega = \emptyset$, then for every factorization $\varphi(a_1 \odot \cdots \odot a_k) = e$ with $a_i \in \Sigma$ and every i we have $e\varphi(a_1 \odot \cdots \odot a_i) \notin s^{-1}P$. Indeed, if for some $\varphi(a_1 \odot \cdots \odot a_k) = e$ we have $e\varphi(a_1 \odot \cdots \odot a_i) \in s^{-1}P$, then the trace $x(a_1 \odot \cdots \odot a_k)^n(a_1 \odot \cdots \odot a_i) \in T$ for every $x \in \varphi^{-1}(s)$ and $n \in \mathbb{N}$. This contradicts the premise.

Now if $\lim(T) \supseteq \varphi^{-1}(s) \odot (\varphi^{-1}(e))^\omega$ we pick an arbitrary factorization $\varphi(a_1 \odot \cdots \odot a_k) = e$ and consider the sequence $(t_i)_i$ of traces given by $t_0 = x \in \varphi^{-1}(s)$ and $t_{i+1} = t_i \odot a_1 \odot \cdots \odot a_k$. Then by (d), there exists a subsequence $(t_{a_i})_i$, such that the pair $(t_{a_i}, t_{a_{i+1}})$ is stable. Since $\varphi(t_{a_i}) = s = se$ for all i, this implies that $x(a_1 \odot \cdots \odot a_k)(a_1 \odot \cdots \odot a_k)^r \odot a_1 \odot \cdots \odot a_j \in T$ for some $1 \leq j \leq k$ and $0 \leq r < a_{i+1} - a_i$. Hence, $\varphi((a_1 \odot \cdots \odot a_k)^{r+1} \odot a_1 \odot \cdots \odot a_j) = e\varphi(a_1 \odot \cdots \odot a_j) \in s^{-1}P$.

(e) $\implies$ (a): By lemma 4.12, we may pick a finite semigroup S, a subset P of S and a morphism φ from $\mathbb{M}(\Sigma, I)$ onto S which recognizes T and saturates $\lim(T)$. By (e), every linked pair has the P-cut property. Let $\alpha \in \Gamma^{-1}(\theta)$ for some $\theta \in \lim(T)$. We may factorize $\theta = \alpha(0) \odot \alpha(1) \odot \cdots$. Let (s, e) be a linked pair associated with a superfactorzation of this factorization and denote the corresponding factorization of α by $\alpha = uv_0v_1v_2\cdots$. Let $v_i = v_{i1} \cdots v_{ik_i}$ with $v_{ij} \in \Sigma$. Then, because $\varphi(\Gamma(v_i)) = e$ and because (s, e) has the P-cut property, the factorization $e = \varphi(v_{i1} \odot \cdots \odot v_{ik_i})$ satisfies $e\varphi((v_{i1} \odot \cdots \odot v_{ij}) \in s^{-1}P$ for some j. Hence $\varphi(Gamma(u) \odot \Gamma(v_0) \odot \cdots \odot \Gamma(v_{r-1}) \odot v_{r1} \odot \cdots \odot v_{rj}) = s \cdot e \cdot \varphi(v_{r1} \odot \cdots \odot v_{rj}) \in P$. Hence α has infinitely many prefixes in T, so $\alpha \in \lim(L)$. ■

Corollary 4.15 *Let $K = \Gamma^{-1}(T)$ for some $T \in \mathsf{Rec}(\mathbb{M}(\Sigma, I))$. Given the automaton $\mathfrak{A}_K$, it is decidable in time $\mathcal{O}(|Q|^2 \cdot |\Sigma|(|\Sigma| + \log|Q|))$ whether or not K is limit-stable.*

Proof Let $\mathfrak{A}_K = (Q, \Sigma, q_0, \delta, F)$ be any DFA recognizing a language K. Write $\mathfrak{A}_{q,q'} = (Q, \Sigma, q, \delta, \{q'\})$ and $\mathfrak{A}_q = (Q, \Sigma, q, \delta, F)$. Denote by $K \cdot \Sigma^*$ the language containing a finite prefix from K. Note that F, I-cycle closure is equivalent to the following property: For every state $q \in Q$ the language $K_q = L(\mathfrak{A}_{q,q}) \cap L(\mathfrak{A}_q) \cdot \Sigma^*$ is trace-closed.

Since K_q is recognizable and a DFA for K_q can be constructed from $\mathfrak{A}_K$ in $\mathcal{O}(|Q| \cdot |\Sigma|)$ (take $Q \times \{0, 1\}$ as states and memorize reaching F in the second component), we can obtain the minimal DFA for K_q from $\mathfrak{A}_K$ in time $\mathcal{O}(|Q| \cdot |\Sigma| + |Q| \cdot |\Sigma| \cdot \log |Q|) = \mathcal{O}(|Q| \cdot |\Sigma| \cdot \log |Q|)$ using Hopcroft's algorithm. We then have to check if this automaton is I-diamond. This requires time $\mathcal{O}(|Q| \cdot |\Sigma|^2)$. So we have time $\mathcal{O}(|Q| \cdot |\Sigma|(|\Sigma| + \log |Q|))$ for every $q \in Q$. ■

Any I-diamond DWA recognizing a trace-closed language is trivially F, I-cycle closed since for any word $u \in \Sigma^*$ and any $q \in Q$, it holds that $q \overset{u}{\Rightarrow} q$ if and if all states in the path taken by u are accepting. This is because a path from q back to itself also implies an SCC, and therefore any $v \sim_I u$ will also remain in the same SCC which comprises solely of accepting states.

It is also straightforward that for a limit-stable language K, the complement language $\overline{\mathsf{lim}(K)}$ of K's infinitary extension is also recognized by an F, I-cycle closed deterministic co-Büchi automaton (DcBA). The following result is then a consequence of Theorem 4.7, Theorem 4.14, and the definitions.

Theorem 4.16 *A trace-closed language $L \subseteq \Sigma^\omega$ is recognized by an I-diamond DWA if and only if it is recognized by both an F, I-cycle closed DBA and an F, I-cycle closed DcBA.*

This result is in nice correspondence with the classical Borel level where weakly recognizable languages are precisely those that lie in the intersection of deterministic Büchi and deterministic co-Büchi recognizable languages. Finally, we now demonstrate that the class of limit-stable languages is expressive enough to generate all trace-closed recognizable ω-language.

4.4 Recognizable Trace-Closed Languages

In [DM94], it was shown that for every recognizable ω-trace language $\Theta \subseteq \mathbb{R}(\Sigma, I)$, the corresponding trace-closed recognizable ω-language $L = \Gamma^{-1}(\Theta)$ is recognized by an I-diamond deterministic Muller automaton (DMA). On the other hand, and quite similar to the problem we saw in the case of the general class of I-diamond DBA, it is not true that every I-diamond DMA recognizes a trace-closed language. Similar to the property of F, I-cycle closure for DBAs, we present a condition over the acceptance component $\mathcal{F}$ of I-diamond DMAs to enable a characterization.

Given an automaton, two of its states p, q, and a word $u \in \Sigma^*$, we denote with $\mathsf{Occ}(p \xrightarrow{u} q)$ the set of states occurring in the run from p to q over u.

Definition 4.17 *Given (Σ, I), an I-diamond DMA $\mathfrak{A} = (Q, \Sigma, q_0, \delta, \mathcal{F})$ is said to be* $\mathcal{F}, I$-cycle closed *if for all $u, v \in \Sigma^*$ such that $u \sim_I v$, and all $q \in Q$, we have $\mathsf{Occ}(q \xrightarrow{u} q) \in \mathcal{F}$ iff $\mathsf{Occ}(q \xrightarrow{v} q) \in \mathcal{F}$.*

$\mathcal{F}, I$-cycle closure was mentioned in [Mus94, Chapter 7] under the simple term *closure* (see Definition 2.7). We obtain an independent proof of the following result by using an approach very similar to that we used to show the equivalence Theorem 4.14:((a)) $\Leftrightarrow$ ((f)).

Theorem 4.18 (also cf. Theorem 2.8) *For any language $\Theta \subseteq \mathbb{R}(\Sigma, I)$ of infinite traces, Θ is recognized by a DACMA if and only if the trace-closed language $L = \Gamma^{-1}(\Theta)$ is recognized by an $\mathcal{F}, I$-cycle closed DMA.*

The $\mathcal{F}, I$-cycle closure property of I-diamond DMAs is easily understood as an extension of the F, I-closure property of I-diamond DBAs. As a final justification of considering the class of infinitary limits of limit-stable languages, we see that this class is indeed suitable for describing all trace-closed, recognizable ω-languages.

Theorem 4.19 *Let L be a trace-closed ω-language. L is a recognizable ω-language iff L is a finite Boolean combination of infinitary limits of limit-stable languages, i.e., a finite Boolean combination of F, I-cycle closed DBA recognizable languages.*

Proof Let $L \subseteq \Sigma^\omega$ be recognizable, trace-closed. Pick a DACMA (c.f. Sec. 4.1) $\mathfrak{M}$ recognizing L. Recall that the global transition behavior of $\mathfrak{M}$ gives an I-diamond DFA, and we denote this DFA by $\mathfrak{A} = (\prod_{a\in\Sigma} Q_a, \Sigma, q_0, \delta)$. Given $q \in Q_a$, we define the DBA $\mathfrak{A}_q = (\prod_{a\in\Sigma} Q_a, \Sigma, q_0, \delta, F_q)$, where $F_q = \{q\} \times \prod_{b\neq a} Q_b$. Note that $\mathfrak{A}_q$ is F_q, I-cycle closed, because for any $q' \in \prod_{a\in\Sigma} Q_a$ and all $u \sim_I v$ with $q' \xrightarrow{u} q'$ and $q' \xrightarrow{v} q'$ we have[2] $\mathsf{Occ}_a(q' \xrightarrow{u} q') = \mathsf{Occ}_a(q' \xrightarrow{v} q')$. Now:

$$L = \bigcup_{(F_a)_{a\in\Sigma}\in\mathcal{F}} \bigcap_{a\in\Sigma} \bigcap_{q\in F_a} L(\mathfrak{A}_q) \cap \bigcap_{q\notin F_a} \overline{L(\mathfrak{A}_q)} \qquad \blacksquare$$

We therefore obtain a Borel-like classification for recognizable ω-trace languages where the lowest level is occupied by reachability and safety languages. Finite Boolean combinations of these languages are characterized by I-diamond DWA. At the next level, we have infinitary limits of limit-stable languages and their complements. And the Boolean combinations of these languages generate the class of all recognizable ω-trace languages.

4.5 Discussion

Motivated by the Borel hierarchy for recognizable ω-languages, the main contribution of this chapter is a new setup for a classification theory of languages of infinite traces in terms of trace-closed word languages. The recourse to word languages is necessitated by a basic observation that there does not exist any equivalent definition of deterministic asynchronous automata recognizing reachability (trace) languages and their complements, the safety languages. Although deterministic trace languages may be characterized in terms of D-SABA, the complexity of the current algorithm to obtain these automata is quite high.

Once in the realm of words, for any trace language $T \in \mathsf{Rec}(\mathbb{M}(\Sigma, I))$, we investigated the relationship between its infinitary extension $\mathsf{ext}(T)$ and the infinitary

[2]This can be proven by an induction on the number of swapping operations needed to obtain v from u.

extension $\mathsf{ext}(K)$, where $K = \Gamma^{-1}(T)$. We showed that any such K can be modified to K_I such that $\mathsf{ext}(K_I)$ is also trace-closed and thus corresponds to the linearizations of $\mathsf{ext}(T)$. Building on this correspondence, we characterized the class of I-diamond DWA recognizable trace-closed languages in terms of Boolean combinations of trace-closed extensions of languages from REG. In a similar vein, we characterized the class of languages $T \in \mathsf{Rec}(\mathbb{M}(\Sigma, I))$ for which the linearization language of $\mathsf{lim}(T)$ is recognizable by an I-diamond DBA obtained from the minimal DFA for $\Gamma^{-1}(T)$, called limit-stable languages. Moreover, we showed that this class of languages is a decidable, proper subclass of finite recognizable trace languages.

These characterizations are further justified because they lead to two interesting results. First, that every recognizable language of infinite traces is a Boolean combination of languages $\mathsf{lim}(T)$ for limit-stable languages T. Second, that I-diamond DWA recognizable languages are precisely those that are both I-diamond deterministic Büchi and I-diamond deterministic co-Büchi recognizable.

Looking back at the language classes studied in Chapter 3, we see that the class of deterministic trace languages strictly subsumes the class of limits of limit-stable languages. However, the class of recognizable ω-trace languages may be described in terms of Boolean combinations of languages from either of these two classes.

Therefore, an important open question remains whether these classes of languages can also be characterized in terms of logic. Such a characterization will allow for a direct comparison with Borel levels.

Chapter 5

On the Distributed Controller Synthesis Problem

An important application of the theory of recognizable ω-languages is in the area of "controller synthesis". In one setting, the controller synthesis problem is viewed as a turn-based game between an adversarial *environment*, called *Player* 0, and a *system*, called *Player* 1. Disjoint alphabets Σ_0 and Σ_1 describe the *moves* that are available to Player 0 and Player 1 respectively. A *play* is an infinite word $\alpha = (a_1, x_1)(a_2, x_2)\ldots \in \Sigma^\omega$ where $\Sigma^\omega = (\Sigma_0 \times \Sigma_1)$. The play α can be understood as comprising of infinite number of *rounds*, where each round i consists of a *Player* 0 *move* $a_i \in \Sigma_0$ followed by a *Player* 1 *move* $x_i \in \Sigma_1$. In general, if after a play prefix $w \in \Sigma^*$ the players play a round with respective moves a and x, then the prefix w is *extended* into a new prefix $v = w \cdot (a, x)$.

Assigning $\Sigma \coloneqq \Sigma_0 \times \Sigma_1$, the *winner* of this game is decided by referring to a *winning condition* $L \subseteq \Sigma^\omega$. Player 0 wins the play α if $\alpha \notin L$, and Player 1 wins if $\alpha \in L$. In order to make their moves in each round, players may access *strategies* that guide their next moves. A *Player* 0 *strategy* $\tau_0 \colon \Sigma^* \to \Sigma_0$ is a mapping that indicates what should be the move of Player 0 given the finite play prefix so far. A *Player* 1 *strategy* $\tau_1 \colon \Sigma^* \to \Sigma_1^{\Sigma_0}$ is mapping that advices how, given the finite play prefix, Player 1 can respond to any move of Player 0. That is, for a play prefix $w \in \Sigma^*$, the Player 1 strategy yields a mapping $\tau_1(w) = \Omega \colon \Sigma_0 \to \Sigma_1$, which prepares Player 1 for any Player 0 move. We interchangeably use the shorthands $[\tau_1(w)](a) = x$ and $\tau_1(w) \colon a \mapsto x$ to denote $\tau_1(w) = \Omega$ and $\Omega(a) = x$.

A play $\alpha = (a_i, x_i)_{i \in \mathbb{N}} \in \Sigma^\omega$ is said to be *consistent with Player* 1 *strategy* if for each $i \in \mathbb{N}, \tau_1(\alpha[1, i-1]) \colon a_i \mapsto x_i$, assuming $\alpha[1, 0] \coloneqq \varepsilon$. A strategy τ_1 is called a *winning strategy* for Player 1 if for each play $\alpha \in \Sigma^\omega$ that is consistent with $\tau_1, \alpha \in L$. A winning strategy thus ensures that Player 1 always moves in such a manner that, no matter how Player 0 moves in its turns, the play always satisfies the winning condition.

Solving a game L is equivalent to answering the question, "Given a winning condition $L \subseteq (\Sigma_0 \times \Sigma_1)^\omega$, does there exist a winning strategy for Player 1?"

In the context of controller synthesis problem, the winning condition is called a *specification*, and a winning strategy for Player 1 is called a *controller* for the system. The controller thus ensures that the system always satisfies the specifications, irrespective of environment's actions. The controller synthesis problem now asks, "Given a specification $L \subseteq (\Sigma_0 \times \Sigma_1)^\omega$, does there exist a controller for the system?" Stated in this manner, this problem is also known as the Church's synthesis problem (see [Tho08] for a survey).

Theorem 5.1 (Büchi & Landweber [BL69]) *For any recognizable game $L \subseteq (\Sigma_0 \times \Sigma_1)^\omega$, it is decidable whether or not Player 1 has a winning strategy; and if yes, then such a winning strategy can be represented using finite memory.*

We focus only on the first part of the theorem, namely the decidability question.

5.1 Distributed Church's Synthesis Problem

We now generalize the classical Church's synthesis problem to the case of distributed synthesis. The task at hand is to device a winning strategy or a controller for a distributed system, which consists of a set $\mathcal{P}$ of processes. The game in this setting is also turn based, where each turn comprises of a move from the environment and a move from the system. In a single move, the environment can play multiple, mutually independent letters $a_1, \ldots a_n \in \Sigma_0, n \leq |\Sigma_0|$. Each letter $a_i \in \Sigma_0$ then requires the processes $\mathsf{dom}(a_i) \subseteq \mathcal{P}$ to play a collective response $x_i \in \Sigma_1$. The system's distributed move is complete only when all the letters a_i of the environment's move have been responded to.

In this manner, the environment is "omniscient" since it can witness the state of the entire system, whereas the processes of the system can only synchronize if the environment plays a letter allowing/inviting them to do so. In order to keep the setting simple, we assume that the system (i.e. the set of synchronizing processes) always responds deterministically. Formally, this setting is described as follows.

Given an alphabet $\Sigma \subseteq \Sigma_0 \times \Sigma_1$, we allow the independence relation $I \subseteq \Sigma \times \Sigma$ to be controlled by the environment. That is, we only refer to an independence relation $I_0 \subseteq \Sigma_0 \times \Sigma_0$, and define $(a, x)\ I\ (b, y) :\Leftrightarrow aI_0b$. In this sense, we say that I is *induced* by I_0. The independence alphabet (Σ, I) governs the possible architectures of the system. A language $\Theta \subseteq \mathbb{R}(\Sigma, I)$ defines a *distributed game*, which we simply refer to as "game Θ".

An environment move is a tuple $\langle a_1, \ldots a_n \rangle, n \leq |\Sigma_0|$, such that for all $i, j \leq n$, if $i \neq j$ then $a_i I a_j$. Since environment moves consist only of mutually independent letters, they are equivalent up to permutations. For example, if $a, b \in \Sigma_0$ are independent letters, then the environment moves $\langle a, b \rangle$ and $\langle b, a \rangle$ are semantically equivalent. For convenience, we refer to a letter from a move as $a \in \langle a, b \rangle$. Define the set $\mathcal{C}_I$ comprising of different ways the environment can "challenge" the controller in a single move as $\mathcal{C}_I := \{\langle a_1, \ldots a_n \rangle \in \Sigma_0^+ \mid a_i I a_j, i \neq j\}$. Now, let $\mathcal{V}$ denote the set comprising of all tuples $V = \langle x_1, \ldots x_m \rangle$ of elements of Σ_1, where for any tuple $m \leq |\Sigma_0|$.

A *play* $\theta \in \mathbb{R}(\Sigma, I)$ of a game consists of an infinite sequence of *rounds*. In each round, first Player 0 chooses to play a move $C = \langle a_1, a_2, \ldots, a_n \rangle \in \mathcal{C}_I$, and then

Player 1 responds with its choice of a tuple $V = \langle x_1, x_2, \ldots, x_n\rangle \in \mathcal{V}$ such that for each $i \in [1, n], (a_i, x_i) \in \Sigma$. The play θ is *won* by Player 1 iff $\theta \in \Theta$.

A *strategy* of Player 0 is a mapping $\tau_0 : \mathbb{M}(\Sigma, I) \to \mathcal{C}_I$, and that of Player 1 is a mapping $\tau_1 : \mathbb{M}(\Sigma, I) \to \mathcal{M}_I$, where $\mathcal{M}_I := \mathcal{V}^{\mathcal{C}_I}$ is set of all mappings from $\mathcal{C}_I$ to $\mathcal{V}$. Moreover, if for some $t \in \mathbb{M}(\Sigma, I)$, $[\tau_1(t)] \colon C \mapsto V$ then it must be the case that $|C| = |V|$ and the ordered pairs (a_i, x_i) of elements from C and V belong to Σ. Given a play prefix t and a round (C, V) involving the players' moves, we use the shorthand notation $t \odot (C, V)$ to denote the extension of the play t with all the letters (a_i, x_i). Since we are only interested in Player 1 strategies, we will henceforth ignore τ_0 and use the notation τ in place of τ_1.

Recall the definition of collective causal views from Section 2.2. A strategy τ of Player 1 is a *distributed strategy* if the following condition is satisfied:

> for a trace $t \in \mathbb{M}(\Sigma, I)$, a Player 0 move $C = \langle a_1, \ldots a_n\rangle$, and each $i \in [1, n]$, if $t_i = \bigsqcup_{p \in \mathsf{dom}(a_i)} t[\max_p(t)]$ is the collective causal view of processes p in $\mathsf{dom}(a_i)$, then $\tau(t) \colon C \mapsto \langle x_1, \ldots x_n\rangle$ iff $\forall i \in [1, n], \tau(t_i) \colon \langle a_i\rangle \mapsto \langle x_i\rangle$.

The above condition merely states that in order for τ to be a distributed strategy, owing to the mutual independence of all letters a_i in a Player 1 move C, it should be possible to define τ component-wise by referring independently to each a_i. In other words, the processes' responses depend only upon the causal view of the processes synchronizing at a_i. Now a play θ is said to be *consistent with the strategy* τ if every play prefix $t \sqsubset \theta$ is extended by referring to τ. That is, if $t \sqsubset t' \sqsubset \theta$ such that infix $t' \setminus t$ comprises solely of mutually independent letters $(a_i, x_i), \cdots (a_n, x_n) \in \Sigma$, then it must be the case that $\tau(t) \colon \langle a_1, \ldots a_n\rangle \mapsto \langle x_1, \ldots x_n\rangle$.

Furthermore, a single round C, V of any play which is consistent with a distributed strategy is equivalent to a sequence of rounds where each round involves a pair a_i, x_i, but not necessarily in the same order. This observation greatly simplifies the analyses of our claims in the subsequent sections. It goes without saying that we are only interested in distributed strategies.

A strategy for Player 1 is a *winning strategy* if all plays consistent with it belong to Θ. Now, the distributed Church's synthesis problem asks:

> *Let (Σ, I) be an independence alphabet where $\Sigma \subseteq \Sigma_0 \times \Sigma_1$ and $I \subseteq \Sigma \times \Sigma$ is induced by $I_0 \subseteq \Sigma_0 \times \Sigma_0$. Given a game $\Theta \subseteq \mathbb{R}(\Sigma, I)$, with Θ recognizable, does Player 1 have a winning, distributed strategy?*

While we are unable to provide mechanisms to decide the distributed Church's synthesis problem in general, we do demonstrate that it compares favorably with the two currently studied versions of distributed synthesis problems assuming recognizable specifications. In other words, we justify the utility of this variant of synthesis problem by showing reductions (under appropriate restrictions) to and from the two established distributed controller synthesis problems – namely, action-based and process-based controller synthesis [MWZ09].

5.2 Distributed Ramadge-Wonham Controller Synthesis

Ramadge and Wonham studied the problem of controller synthesis in another setting [RW89], which can be shown to be equivalent to the Church's setting in the case of word languages. In this setting, we are given a *plant*, which is usually represented by a deterministic, finite state transition system $\mathfrak{T}$ whose transitions are defined over two disjoint sets of actions – called the set Σ_{env} of environment actions and the set Σ_{sys} of system actions. A *specification* is a language $L \subseteq (\Sigma_{evn} \uplus \Sigma sys)^\omega$, comprising of behaviors that are deemed acceptable for the plant.

A *controller* in this setting is a mapping $\sigma\colon (\Sigma_{env} \uplus \Sigma_{sys})^* \to \Sigma_{sys}$, which describes a mechanism of extending finite play prefixes by enabling system actions. The intuition is that the controller cannot control any environment actions; and the synthesis problem asks, "Given a plant $\mathfrak{T}$ and a specification L, is there is a controller σ that chooses system actions so that the behavior of the plant satisfies the specification?"

We formalize these ideas by presenting two variants of Ramadge-Wonham controller synthesis problem for the distributed setting. It will be clear that if the independence relation over the letters is empty, then both these variants reduce to the Ramadge-Wonham synthesis problem as outlined above. The first variant is called the *process-based controller synthesis problem* [MTY05, MWZ09].

Over an independence alphabet (Σ, I), where $\Sigma \coloneqq \Sigma_{env} \uplus \Sigma_{sys}$, a *plant* is an asynchronous transition system $\mathfrak{T} = \big((X_p)_{p \in \mathcal{P}}, \Sigma, (\delta_a)_{a \in \Sigma}, \pi_0\big)$. With each plant, we associate an ω-trace language $\mathit{Plays}^\omega(\mathfrak{T}) = \{\theta \in \mathbb{R}(\Sigma, I) \mid \theta$ induces a valid run of $\mathfrak{T}\}$. In other words, $\mathit{Plays}^\omega(\mathfrak{T})$ consists of all traces θ that would be accepted by $\mathfrak{T}$ if it were a safety automaton with all states in $X_{2^\mathcal{P}}$ assumed to be safe. Similarly, define the set $\mathit{Plays}(\mathfrak{T}) \coloneqq \{t \in \mathbb{M}(\Sigma, I) \mid \exists \theta \in \mathit{Plays}^\omega(\mathfrak{T})\colon t \sqsubset \theta\}$ of finite traces that induce a valid finite run of $\mathfrak{T}$.

For $p \in \mathcal{P}$, define $\mathit{Plays}_p(\mathfrak{T}) \coloneqq \{t \in \mathbb{M}(\Sigma, I) \mid \exists s \in \mathit{Plays}(\mathfrak{T})\colon t = s[\max_p(s)]\}$. Note that $\mathit{Plays}_p(\mathfrak{T}) \subseteq \mathit{Plays}(\mathfrak{T})$, and that each trace $t \in \mathit{Plays}_p(\mathfrak{T})$ represents a causal view of p and contains a unique maximal event which is a p-event.

A *process-based strategy* $\sigma = (f_p)_{p \in \mathcal{P}}$ is a collection of functions $f_p\colon \mathit{Plays}_p(\mathfrak{T}) \to 2^{\Sigma_p}$, where $\Sigma_p \coloneqq \mathsf{dom}^{-1}(p)$. Furthermore, the functions f_p satisfy the constraint: if $f_p(t) = A$ for some $t \in \mathit{Plays}_p(\mathfrak{T})$ and if $t \odot a \in \mathit{Plays}(\mathfrak{T})$ for any $a \in \Sigma_{env}$, then it must hold that $a \in A$. The set $f_p(t)$ is referred to as the set of actions *enabled* by the process p; and the constraint mentioned here formalizes the intuition that environment actions are uncontrollable and therefore must always be enabled.

The set $\mathit{Plays}(\mathfrak{T}, \sigma)$ of finite play prefixes *consistent* with a process-based strategy σ is defined inductively as follows:

- $\varepsilon \in \mathit{Plays}(\mathfrak{T}, \sigma)$;
- for $a \in \Sigma_{env}$ and $t \in \mathit{Plays}(\mathfrak{T}, \sigma)$, if $t \odot a \in \mathit{Plays}(\mathfrak{T})$ then it must hold that $t \odot a \in \mathit{Plays}(\mathfrak{T}, \sigma)$; and
- for $a \in \Sigma_{sys}$ and $t \in \mathit{Plays}(\mathfrak{T}, \sigma)$, it holds that $t \odot a \in \mathit{Plays}(\mathfrak{T}, \sigma)$ only

if $t \odot a \in \mathit{Plays}(\mathfrak{T})$ and a is enabled by all processes in $\mathsf{dom}(a)$, that is $a \in \bigcap_{p\in\mathsf{dom}(a)} f_p(t[\max_p(t)])$.

In the presence of the above-mentioned constraints on functions f_p, the definition of $\mathit{Plays}(\mathfrak{T}, \sigma)$ subsumes the assumption that environment actions are uncontrollable; in addition to that, a system action $a \in \Sigma_{sys}$ is possible only if all the processes in $\mathsf{dom}(a)$ agree to it. Note that this agreement must already be present before these processes actually synchronize on a.

Using these definitions, we define the set of infinite plays *consistent* with σ as $\mathit{Plays}^\omega(\mathfrak{T}, \sigma) = \{a_1 \odot a_2 \ldots \in \mathbb{R}(\Sigma, I) \mid \forall i \in \mathbb{N}\colon a_1 \odot \ldots \odot a_i \in \mathit{Plays}(\mathfrak{T}, \sigma)\}$.

For a recognizable specification $\Theta \subseteq \mathbb{R}(\Sigma, I)$, a process-based strategy σ is called a *controller* if $\mathit{Plays}^\omega(\mathfrak{T}, \sigma) \subseteq \Theta$. As a guard against constructing trivial controllers, the controllers are expected to be *non-blocking.* A non-blocking controller guarantees that each play in $\mathit{Plays}(\mathfrak{T}, \sigma)$ that can be extended in $\mathit{Plays}(\mathfrak{T})$ can also be extended in $\mathit{Plays}(\mathfrak{T}, \sigma)$. That is, for any $t \in \mathit{Plays}(\mathfrak{T}, \sigma)$, if there exists $t' \sqsupset t$ such that $t' \in \mathit{Plays}(\mathfrak{T})$ then there must exist $t'' \in \mathit{Plays}(\mathfrak{T}, \sigma)$ such that $t'' \sqsupset t$.

For an ATS $\mathfrak{T}$ and recognizable ω-trace language Θ, the pair $(\mathfrak{T}, \Theta)$ is referred to as an *instance* of a distributed Ramadge-Wonham controller synthesis problem, and the process-based controller synthesis problems asks,

> *Given an instance $(\mathfrak{T}, \Theta)$, where $\mathfrak{T}$ is an ATS and Θ is a recognizable ω-trace language, does there exist a non-blocking, process-based controller for this instance?*

While this problem is open in general, it has been solved for specific restricted classes of plants $\mathfrak{T}$ [MTY05, MS13, MWZ09].

The *action-based controller synthesis problem* [GLZ04, MWZ09] is another variant of distributed Ramadge-Wonham synthesis problem, where a different kind of controllers are considered. Over an alphabet $\Sigma = \Sigma_{env} \uplus \Sigma_{sys}$, a plant $\mathfrak{T}$ is again an ATS, and $\mathit{Plays}^\omega(\mathfrak{T})$ and $\mathit{Plays}(\mathfrak{T})$ are also defined as before. For $a \in \Sigma$, we introduce $\mathit{Plays}_a(\mathfrak{T}) := \{t \in \mathbb{M}(\Sigma, I) \mid \exists t' \in \mathit{Plays}(\mathfrak{T})\colon t = \bigsqcup_{p\in\mathsf{dom}(a)} t'[\max_p(t')]\}$. Observe once again that $\mathit{Plays}_a(\mathfrak{T}) \subseteq \mathit{Plays}(\mathfrak{T})$, and that each trace $t \in \mathit{Plays}_a(\mathfrak{T})$ is a collective causal view of $\mathsf{dom}(a)$.

An *action-based strategy* $\sigma = (f_a)_{a\in\Sigma}$ is a collection of functions $f_a\colon \mathit{Plays}_a(\mathfrak{T}) \to \{\mathtt{tt}, \mathtt{ff}\}$. Once again, the functions f_a satisfy a constraint to ensure that environment actions are uncontrollable, viz. for all $a \in \Sigma_{env}$ and all $t \in \mathit{Plays}_a(\mathfrak{T})$, if $t \odot a \in \mathit{Plays}(\mathfrak{T})$ then it must hold that $f_a(t) = \mathtt{tt}$. This implies that environment actions are always enabled, and if a play can be extended with an environment action, then the function f_a always allows it. Therefore, it is often preferred to denote an action-based strategy σ simply as the collection $(f_a)_{a\in\Sigma_{sys}}$.

The set $\mathit{Plays}(\mathfrak{T}, \sigma)$ of finite play prefixes that are *consistent* with an action-based strategy σ is defined inductively as follows:

- $\varepsilon \in \mathit{Plays}(\mathfrak{T}, \sigma)$;

- for $a \in \Sigma$ and $t \in \mathit{Plays}(\mathfrak{T}, \sigma)$, it holds that $t \odot a \in \mathit{Plays}(\mathfrak{T}, \sigma)$ only if $f_a(t_a) = \mathtt{tt}$, where $t_a = \bigsqcup_{p \in \mathsf{dom}(a)} t[\max_p(t)]$.

The definitions of set $\mathit{Plays}^\omega(\mathfrak{T}, \sigma)$ and specification Θ, and the notion of non-blocking controllers are the same as those defined previously. The action-based controller synthesis problem now asks,

> *Given an instance $(\mathfrak{T}, \Theta)$, where $\mathfrak{T}$ is an ATS and Θ is a recognizable ω-trace language, does there exist a non-blocking, action-based controller for this instance?*

Once again, the decidability of this question remains open for arbitrary instances. With specific restrictions on $\mathfrak{T}$ and Θ, this problem has been solved in [GLZ04, GLZ05]. Moreover, it has been shown that the process-based synthesis problem can be reduced to the action-based synthesis problem.

Theorem 5.2 (Muscholl & Walukiewicz & Zeitoun [MWZ09]) *Given an instance $(\mathfrak{T}, \Theta)$ over an alphabet (Σ, I), one can construct an instance $(\mathfrak{T}', \Theta')$ over some alphabet (Σ', I') such that there exists a process-based controller for $(\mathfrak{T}, \Theta)$ if and only if there exists an action-based controller for $(\mathfrak{T}', \Theta')$.*

5.3 Distributed Synthesis Problems in Perspective

An intuition behind the reduction mentioned in Theorem 5.2 is that an action-based strategy has more information regarding the play prefix than a process-based strategy. The functions f_a can access the collective causal view $\bigsqcup_{p \in \mathsf{dom}(a)} t[\max_p(t)]$ for any prefix t, and then decide on whether or not to enable a. On the other hand, in a process-based strategy, the functions $f_p, p \in \mathsf{dom}(a)$, cannot access $t[\max_q(t)], q \neq p$, before determining whether or not to enable a. In this section, we investigate how process-based and action-based controller synthesis problems relate to the distributed Church's synthesis problem.

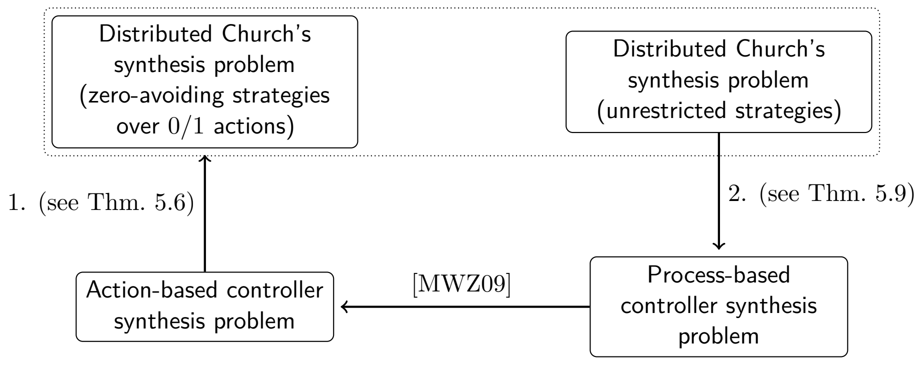

Distributed Church's synthesis problem (zero-avoiding strategies over 0/1 actions)
Distributed Church's synthesis problem (unrestricted strategies)
1. (see Thm. 5.6)
2. (see Thm. 5.9)
Action-based controller synthesis problem
[MWZ09]
Process-based controller synthesis problem

Figure 5.1: We complete this picture by demonstrating reductions 1 and 2.

We first introduce the concept of "zero-avoiding strategies" for distributed Church's synthesis problem. The definition introduced in Section 5.1 will be referred to as the distributed Church's synthesis problem with "unrestricted strategies". Although the former appears to be a subclass of the latter, we would like to indicate this without a formal argument, that imposing a restriction over the nature of strategies (e.g. by requiring the strategies to be non-blocking or zero-avoiding) makes the synthesis problem harder than the one where there is no restriction on distributed strategies.

The notion that one problem is harder than, or is at least as hard as, another problem is built upon the standard notion of "reduction" of one problem class to another. A problem Φ can be reduced to a problem Ψ if for every instance φ of Φ one can construct an instance ψ of Ψ, such that there exists a solution for φ if and only if there exists a solution for ψ. In this manner, a reduction implies that the task of developing solution techniques for Φ is at most as hard as the task of developing solution techniques for Ψ.

Zero-avoiding Strategies and 0/1 Games

We consider Church's controller synthesis problem where a special system action $0 \in \Sigma_1$ is identified as the "zero". A strategy τ is said to be *zero avoiding* if for each $t \in \mathbb{M}(\Sigma, I)$, there exists at least one environment action $a \in \Sigma_0$ such that it is *not* the case that $\tau(t)\colon \langle a \rangle \mapsto \langle 0 \rangle$. Note that a zero-avoiding strategy does not imply that the system will never play 0, it only means that there will always be some non-zero system event on offer. A zero-avoiding strategy thus avoids the constant "zero-mapping" which assigns each environment event $a \in \Sigma_0$ to 0.

Furthermore, Church's controller synthesis problem *over* 0/1 *actions* is one where the set Σ_1 of system actions consists of exactly two actions, one of which is the zero element 0. We refer to such a problem as simply a 0/1 *game* Θ. After restricting system actions to either 0 or 1 and the strategies to be zero-avoiding, the Church's controller synthesis problem begins to resemble the action-based synthesis problem very closely. We now formalize this notion.

Problem Reductions

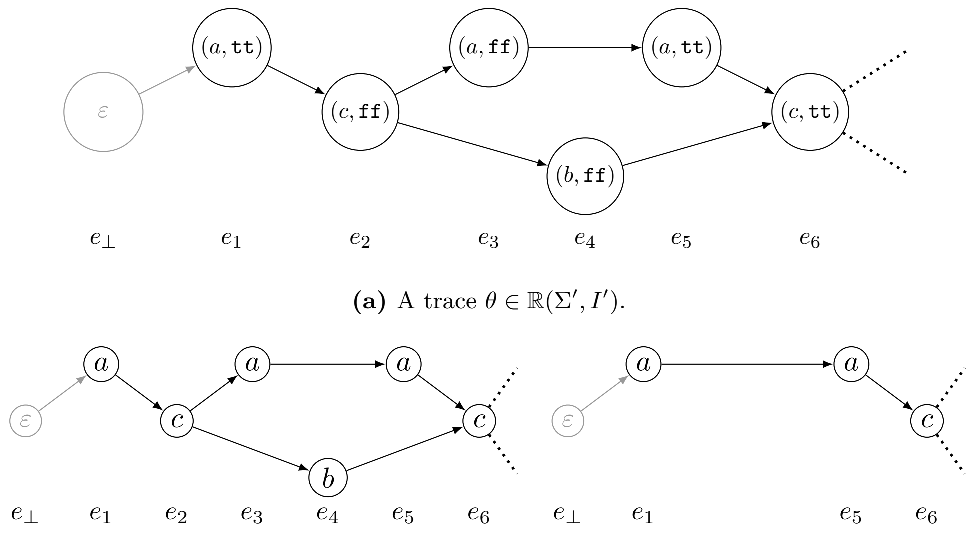
We begin with reducing action-based controller synthesis problem to distributed Church's synthesis problem over 0/1 actions with zero-avoiding strategies. Consider an alphabet $\Sigma = \Sigma_{sys} \uplus \Sigma_{env}$, an independence relation $I \subseteq \Sigma \times \Sigma$, a plant $\mathfrak{T}$, and a specification $\Theta \subseteq \mathbb{R}(\Sigma, I)$. Clearly, any play that is winning for the system belongs to the language $\mathit{Plays}^{\omega}(\mathfrak{T}) \cap \Theta$.

To describe the game specification $\Theta' \subseteq \mathbb{R}(\Sigma', I')$ for the Church's synthesis problem, we first define:

- $\Sigma'_0 := \Sigma = \Sigma_{env} \uplus \Sigma_{sys}$; $\Sigma'_1 := \{\mathtt{tt}, \mathtt{ff}\}$, with $\mathtt{ff}$ denoting the 0 of Σ'_1.
- $\Sigma' := (\Sigma_{sys} \times \{\mathtt{tt}, \mathtt{ff}\}) \cup (\Sigma_{env} \times \{\mathtt{tt}\})$; and

- $I' := \{((a,x),(b,y)) \in \Sigma' \times \Sigma' \mid (a,b) \in I\}$.

For a trace $\theta \in \mathbb{R}(\Sigma', I')$, we define its Σ-*projection* $\theta_{|\Sigma} \in \mathbb{R}(\Sigma, I)$ as the projection over the first component of events in θ; define the $\texttt{tt}$-Σ-*projection* $\theta_{|\{\Sigma,\texttt{tt}\}} \in \mathbb{M}(\Sigma, I) \cup \mathbb{R}(\Sigma, I)$ as the projection over the first component of θ, but only at those events where the second component is $\texttt{tt}$. See Figure 5.2 for example. For a trace $t = a_1 \odot a_2 \odot \cdots \in \mathbb{M}(\Sigma, I)$, we define the trace $t \times \texttt{tt} \in \mathbb{M}(\Sigma', I')$ as $t \times \texttt{tt} := (a_1, \texttt{tt}) \odot (a_2, \texttt{tt}) \odot \cdots$, and $\varepsilon \times \texttt{tt} := \varepsilon$.



(a) A trace $\theta \in \mathbb{R}(\Sigma', I')$.

(b) The Σ-projection $\theta_{|\Sigma} \in \mathbb{R}(\Sigma, I)$.

(c) For θ, the $\texttt{tt}$-Σ-projection $\theta_{|\{\texttt{tt},\Sigma\}}$ may even be a finite trace.

Figure 5.2: Illustrating Σ-projection and $\texttt{tt}$-Σ-projection.

The reduction to the distributed Church's synthesis problem is understood thus. The environment challenges the controller with a move C where C consists of mutually independent letters from $\Sigma_{env} \uplus \Sigma_{sys}$. To each $a \in C \cap \Sigma_{env}$, the processes in $\mathsf{dom}(a)$ can only respond with $\texttt{tt}$; whereas for $a \in C \cap \Sigma_{sys}$, the processes in $\mathsf{dom}(a)$ can choose between $\texttt{tt}$ and $\texttt{ff}$. Note that in action-based control, the play only proceeds if the controller enables an event. In Church's setting however, the play progresses irrespective of whether the strategy says $\texttt{tt}$ or $\texttt{ff}$. This implies that, for any play $\theta \in \mathbb{R}(\Sigma', I')$, the corresponding play in the action-based setting is obtain from the $\texttt{tt}$-Σ-projection $\theta_{|\{\Sigma,\texttt{tt}\}}$. Therefore, the winning condition Θ' for the Church's synthesis problem contains infinite plays that belong to the language $\Theta_1 := \{\theta \in \mathbb{R}(\Sigma', I') \mid \theta_{|\{\Sigma,\texttt{tt}\}} \in \mathit{Plays}^\omega(\mathfrak{T}) \cap \Theta\}$.

However, unlike the action-based setting where the systems can control system actions, it is the environment in the Chruch's setting that schedules all the actions. Therefore the environment has a possibility to cheat by withholding certain system actions that are enabled in the action-based setting and are crucial for the system to win. Consider, for example, that in the action-based setting, enabling a certain controllable action $a \in \Sigma_{sys}$ is crucial for the distributed controller. In the Church's setting however, the environment has no obligation to play a move $C \in \mathcal{C}_I$ containing a, in spite of the fact that $\langle a \rangle \mapsto \langle \mathtt{tt} \rangle$. Therefore, even though there may exist an action-based controller in the former setting, the zero-avoiding distributed strategy may not be a winning strategy in the latter. To counter this, we ensure that each such unfair plays are also winning for Player 1 by adding the language $\Theta_2 \coloneqq \{\theta \in \mathbb{M}(\Sigma', I') \mid \theta_{|\{\Sigma,\mathtt{tt}\}} \in \mathbb{M}(\Sigma, I)\}$ in the winning condition. By doing so, we guarantee that the environment either plays fairly, by allowing the system to make good use of its zero-avoiding strategy, or loses. One may view Θ_2 as an "insurance" for the system against an unfair environment.

We now define the distributed game $\Theta' \coloneqq \Theta_1 \cup \Theta_2$.

Remark 5.3 *If the ω-trace language $\Theta \subseteq \mathbb{R}(\Sigma, I)$ is recognizable then so is the ω-trace language $\Theta' \subseteq \mathbb{R}(\Sigma', I')$.*

Lemma 5.4 *If $(\mathfrak{T}, \Theta)$ has a non-blocking, action-based controller then there exists a winning, zero-avoiding distributed strategy for Player 1 in the 0/1 game Θ'.*

Proof Given a non-blocking, action-based controller $\sigma = \{f_a\}_{a\in\Sigma_{sys}}$, we construct a strategy τ for Player 1 as follows:

$$\text{for } a \in \Sigma'_0, \text{ for } t \in \mathbb{M}(\Sigma', I'), \text{ assign } \tau(t)\colon \langle a \rangle \overset{\text{def}}{\mapsto} \langle \mathtt{tt} \rangle \text{ iff } f_a(t_{|\{\Sigma,\mathtt{tt}\}}) = \mathtt{tt}.$$

Since the action-based controller σ is a distributed controller, the strategy τ defined above can be generalized in a straightforward manner to a distributed strategy $\tau\colon \mathbb{M}(\Sigma', I') \to \mathcal{V}^{\mathcal{C}_{I'}}$. Also, since σ is non-blocking, τ is zero-avoiding.

Now we need to show that the zero-avoiding distributed strategy τ is indeed a winning strategy for Player 1. For that, it suffices to show that any play θ that is consistent with τ is a winning play for Player 1. Since τ is a distributed strategy, without loss of generality, we assume that the play $\theta \in \Sigma^\omega$ consists of infinitely many rounds (C_i, V_i) where each of the environment and system moves are singleton tuples, i.e. for all $i \in \mathbb{N}, C_i = \langle a_i \rangle, a_i \in \Sigma'_0$ and $V_i = \langle x_i \rangle, x_i \in \Sigma'_1$.

We can assume that the environment plays fairly, and for a play $\theta \in \mathbb{R}(\Sigma', I')$, $\theta_{|\{\Sigma,\mathtt{tt}\}} \in \mathbb{R}(\Sigma, I)$. Now we need[1] to show that $\theta_{|\{\Sigma,\mathtt{tt}\}} \in \mathit{Plays}^\omega(\mathfrak{T}, \sigma)$. In order to show this, it suffices to show that for any prefix $t \sqsubset \theta$, there exists an extension $s \sqsupseteq t$ such that $s \sqsubset \theta$ and $s_{|\{\Sigma,\mathtt{tt}\}} \in \mathit{Plays}(\mathfrak{T}, \sigma)$. This can be done by induction over the size of the prefixes $s_{|\{\Sigma,\mathtt{tt}\}}$.

By Definition (cf. [MWZ09]), the empty prefix ε belongs to $\mathit{Plays}(\mathfrak{T}, \sigma)$. Therefore, at the base of the induction, we have all prefixes $t \in \mathbb{M}(\Sigma', I')$ with $t_{|\{\Sigma,\mathtt{tt}\}} = \varepsilon$. As the induction hypothesis, we have a trace $t_i \in \mathbb{M}(\Sigma', I')$ such that $t_{i|\{\Sigma,\mathtt{tt}\}} = s_i \in \mathit{Plays}(\mathfrak{T}, \sigma)$. Since σ is a non-blocking, action-based controller, we know that for each $a \in \Sigma_{env}, f_a(s_i) = \mathtt{tt}$, and that there exists $b \in \Sigma_{sys}, f_b(s_i) = \mathtt{tt}$; which implies that $s_i \odot a \in \mathit{Plays}(\mathfrak{T}, \sigma)$ and $s_i \odot b \in \mathit{Plays}(\mathfrak{T}, \sigma)$. Therefore, consider a

[1] By the definition of an action-based controller, $\mathit{Plays}^\omega(\mathfrak{T}, \sigma) \subseteq \mathit{Plays}^\omega(\mathfrak{T}) \cap \Theta$, and therefore we can follow the implications $\theta_{|\{\Sigma,\mathtt{tt}\}} \in \mathit{Plays}^\omega(\mathfrak{T}, \sigma) \Rightarrow \theta_{|\{\Sigma,\mathtt{tt}\}} \in \mathit{Plays}^\omega(\mathfrak{T}) \cap \Theta \Rightarrow \theta \in \Theta_1$.

shortest prefix $t'_i \sqsupseteq t_i$ after which Player 0 either challenges the system with an environment action $\langle a \rangle, a \in \Sigma_{env}$ or (by the assumption of this case) plays $\langle b \rangle$, where $b \in \Sigma_{sys}$ is an action enabled by σ.

Since $t_{i|\{\Sigma,\texttt{tt}\}} = t'_{i|\{\Sigma,\texttt{tt}\}}$, by the definition of the strategy τ, we have that $[\tau(t_i)](\langle a \rangle) = [\tau(t'_i)](\langle a \rangle) = \langle \texttt{tt} \rangle$ and $[\tau(t_i)](\langle b \rangle) = [\tau(t'_i)](\langle b \rangle) = \langle \texttt{tt} \rangle$. This implies that the play t'_i is extended either as $t^a_i := t'_i \odot (a, \texttt{tt})$ or as $t^b_i := t'_i \odot (b, \texttt{tt})$. Hence we have either $t^a_{i|\{\Sigma,\texttt{tt}\}} = s_i \odot a \in \mathit{Plays}(\mathfrak{T}, \sigma)$ or otherwise, $t^b_{i|\{\Sigma,\texttt{tt}\}} = s_i \odot b \in \mathit{Plays}(\mathfrak{T}, \sigma)$. This completes the induction. ■

Lemma 5.5 *If in the* 0/1 *game* Θ' *there exists a winning, zero-avoiding distributed strategy for Player* 1 *then there exists a non-blocking, action-based controller for the instance* $(\mathfrak{T}, \Theta)$.

Proof Similar to the proof of Lemma 5.4, we assume a winning, distributed strategy τ for Player 1 and construct a strategy $\sigma = \{f_a\}_{a \in \Sigma_{sys}}$ for the action-based game as follows:

> for $t \in \mathbb{M}(\Sigma, I)$ and $a \in \Sigma_{env} \uplus \Sigma_{sys}$, consider $t' = t \times \texttt{tt}$ and set $f_a(t) := \texttt{tt}$ if and only if $[\tau(t')] \colon \langle a \rangle \mapsto \langle \texttt{tt} \rangle$.

In the action-based strategy $\sigma = \{f_a\}_{a \in \Sigma_{sys}}$, the mappings f_a depend only on the causal views of processes in $\mathsf{dom}(a)$ because τ is a distributed strategy. Moreover, the fact that τ is zero-avoiding immediately implies that σ is a non-blocking, distributed strategy.

It remains to be shown that $\mathit{Plays}^\omega(\mathfrak{T}, \sigma) \subseteq \Theta$. Intuitively, we have defined the strategy σ under the assumption that in the Church's synthesis game with a zero-avoiding strategy, the Player 0 only plays actions $\langle a \rangle$ which are mapped to $\langle \texttt{tt} \rangle$ by τ. In this manner, the Player 0 prevents Player 1 from "waiting" and "gathering" arbitrary amount of information. But since τ is a winning strategy, the resulting play is still winning for Player 1.

Formally, we have that

1. for any trace $\kappa \in \mathit{Plays}^\omega(\mathfrak{T})$, if $s \sqsubset \kappa$ then $s \in \mathit{Plays}(\mathfrak{T})$;
2. since τ is a winning strategy, if θ' is consistent with τ and θ' can be expressed as $\theta \times \texttt{tt}$ for some $\theta \in \mathbb{R}(\Sigma, I)$ then $\theta \in \mathit{Plays}^\omega(\mathfrak{T}) \cap \Theta$;
3. from items 1 and 2 above, any prefix $t' \sqsubset \theta'$ can be expressed as $t \times \texttt{tt}$ for some $t \in \mathit{Plays}(\mathfrak{T})$;
4. since σ is obtained from τ, $\forall a \in \Sigma_{sys} \colon f_a(\varepsilon) = \texttt{tt} \Leftrightarrow [\tau(\varepsilon)] : \langle a \rangle \mapsto \langle \texttt{tt} \rangle \Rightarrow a \in \mathit{Plays}(\mathfrak{T})$; and therefore inductively, if $t \in \mathit{Plays}(\mathfrak{T})$ and $f_a(t) = \texttt{tt}$ then $t \odot a \in \mathit{Plays}(\mathfrak{T})$;
5. then, for any infinite extension θ of prefixes t in the manner of item 4, it holds that $\theta \in \mathit{Plays}^\omega(\mathfrak{T})$;

6. and finally, since σ is obtained from a winning strategy τ, it follows from item 2 that for any such θ from item 5, $\theta \in \mathit{Plays}^{\omega}(\mathfrak{T}) \cap \Theta$.

Item 5 in particular ensures that any play $\theta \in \mathbb{R}(\Sigma, I)$ that is consistent with the strategy σ belongs to $\mathit{Plays}^{\omega}(\mathfrak{T})$. Therefore, the set $\mathit{Plays}^{\omega}(\mathfrak{T}, \sigma)$ is well defined. From item 6 it follows that $\mathit{Plays}^{\omega}(\mathfrak{T}, \sigma) \subseteq \mathit{Plays}^{\omega}(\mathfrak{T}) \cap \Theta \subseteq \Theta$.

Hence, we conclude that σ is an action-based controller for $(\mathfrak{T}, \Theta)$. ■

From Lemmas 5.4 and 5.5, we obtain the following result.

Theorem 5.6 *For any instance $(\mathfrak{T}, \Theta)$ over an independence alphabet (Σ, I), one can construct a $0/1$ game Θ' over a suitably defined independence alphabet $(\Sigma_0 \times \Sigma_1, I')$ such that there exists an action-based controller for $(\mathfrak{T}, \Theta)$ if and only if there exists a winning, zero-avoiding distributed strategy for Player 1 in game Θ'.*

With Theorem 5.6, we have reached mid-way into proving the reductions depicted in Figure 5.1. Now we leave behind the case of zero-avoiding strategies over $0/1$ actions, and start afresh to demonstrate that solving the distributed Church's synthesis problem in its generality no more difficult than solving the process-based controller synthesis problem.

We begin with the alphabet $\Sigma' = \Sigma_0 \times \Sigma_1$, an independence relation $I_0 \subseteq \Sigma_0^2$, and derive the relation $I' \subseteq \Sigma' \times \Sigma'$ from I_0 (cf. Section 5.1). Let $\Theta' \subseteq \mathbb{R}(\Sigma', I')$ be a recognizable game specification for distributed Church's synthesis, and let $\mathfrak{T}'$ be a deterministic, asynchronous Muller automaton recognizing Θ' (see [DM94]). Note that in $\mathfrak{T}'$, for each $(a, x) \in \Sigma'$, the domain $\mathsf{dom}((a, x))$ is set by referring to $\mathsf{dom}(a)$ since the independence of letters in Σ' depends solely on the independence of letters in Σ_0.

To describe the process-based synthesis problem, we first define:

- $\Sigma_{env} := \Sigma_0$; $\Sigma_{sys} := \Sigma' = \Sigma_0 \times \Sigma_1$; and
- $I := I_0 \, \cup \, I' \, \cup \, \{(a, (b, x)) \in \Sigma_{env} \times \Sigma_{sys} \mid (a, b) \in I_0\}$.
- The domain mappings for Σ_{env} and Σ_{sys} are defined in a natural manner by referring to the corresponding mappings for Σ_0.

Now extend the Muller automaton $\mathfrak{T}'$ to construct a plant $\mathfrak{T}$, which has the following property. Each process in the plant has two types of states. Type-1 states have outgoing transitions on only Σ_{env} events, and Type-2 states have outgoing transitions on only Σ_{sys}. From a Type-1 state, upon making a transition on $a \in \Sigma_{env}$, the processes in $\mathsf{dom}(a)$ arrive at a Type-2 (partial) state from which the only outgoing transitions are on events of the form $(a, x) \in \Sigma_{sys}$, for some $x \in \Sigma_1$. After the second move, the processes arrive back in a Type-1 (partial) state. In this manner, the plant uses two transition to mimic the turn-based setting of the Church's synthesis game. The tight coupling of successive transitions on letters a and (a, x) avoids the possibility of transitions of over consecutive letters a and (b, y), where $a \neq b$ but $\mathsf{dom}(a) \cap \mathsf{dom}(b) \neq \emptyset$.

In other words, each round of play (a, x) in the synthesis game is mimicked by the the pair of events $a \in \Sigma_{env}$, $(a, x) \in \Sigma_{sys}$ in the process-based setting; and each process in the process-based setting can make consecutive moves on letter a followed by letter (b, x) only if $a = b$ and (a, x) is a transition in the original automaton $\mathfrak{T}'$.

Formally, let $\mathfrak{T}' = ((S'_p)_{p\in\mathcal{P}}, \Sigma', \{\delta'_{(a,x)}\}_{(a,x)\in\Sigma'}, \pi'_0, \mathcal{F})$, with the sets S'_p of local states of process p, set $\{\delta'_{(a,x)}\}_{(a,x)\in\Sigma'}$ of transition functions, a global initial state π'_0, and the Muller acceptance component $\mathcal{F}$. To define the plant $\mathfrak{T}$:

- define $S_p \coloneqq S'_p \cup (\Sigma_0 \times S'_p)$ for $p \in \mathcal{P}$
- (Type-1 transition) for $a \in \Sigma_{env}$ and $(s'_p)_{p\in\mathsf{dom}(a)} \in \prod_{p\in\mathsf{dom}(a)} S_p$, define a transition $\delta_a \colon (s'_p)_{p\in\mathsf{dom}(a)} \mapsto \big((a, s'_p)\big)_{p\in\mathsf{dom}(a)}$
- (Type-2 transition) for $(a, x) \in \Sigma_{sys}$ and $\big((a, s'_p)\big)_{p\in\mathsf{dom}(a)} \in \prod_{p\in\mathsf{dom}(a)} S_p$, define a transition $\delta_{(a,x)} \colon \big((a, s'_p)\big)_{p\in\mathsf{dom}(a)} \mapsto (r'_p)_{p\in\mathsf{dom}(a)}$ only if there exists a transition $\delta'_{(a,x)}\big((s'_p)_{p\in\mathsf{dom}(a)}\big) = (r'_p)_{p\in\mathsf{dom}(a)}$ in the Muller automaton $\mathfrak{T}'$
- $\pi_0 = \pi'_0$.

For a trace $\theta = (a_i \odot (a_i, x_i))_{i\in\mathbb{N}}$, define $\theta_{|\Sigma'} \coloneqq ((a_i, x_i))_{i\in\mathbb{N}}$. Similarly, define $t_{|\Sigma'} = (a_i, x_i)_{i=1}^{n}$ for $t \in \mathbb{M}(\Sigma, I)$ by removing any trailing letters $a_j \in t$ that are not followed by (a_j, x_j). Also, for $\theta' = ((a_i, x_i))_{i\in\mathbb{N}}$ define $\theta'_{\uparrow\Sigma} \coloneqq (a_i \odot (a_i, x_i))_{i\in\mathbb{N}}$; similarly define $t'_{\uparrow\Sigma}$.

Finally, we define $\Theta \coloneqq \{\theta \in (\Sigma_{env} \cup\!\!\!\cdot\, \Sigma_{sys})^\omega \mid \exists \theta' \in \Theta' \text{ such that } \theta = \theta'_{\uparrow\Sigma}\}$. Clearly, Θ is regular whenever Θ' is. Now we obtain an instance of the process-based synthesis problem $(\mathfrak{T}, \Theta)$.

Lemma 5.7 *If there exists a winning, distributed strategy for Player* 1 *for specification* Θ' *then there exists a non-blocking, process-based controller for* $(\mathfrak{T}, \Theta)$*.*

Proof Assuming that τ is a winning, distributed strategy for Player 1 in the Church's synthesis game, define a strategy $\sigma = \{f_p\}_{p\in\mathcal{P}}$ for the process-based game as follows:

1. $f_p(t) \coloneqq \{a \in \Sigma_{env} \mid a \in \mathsf{dom}^{-1}(p)\}$ iff $\max_p(t)$ is of the form $(b, x) \in \Sigma_{sys}$;
2. $f_p(t) \coloneqq \{(a, x)\}$ iff $\max_p(t) = a$ and $\tau(t_{|\Sigma'}) \colon \langle a\rangle \mapsto \langle x\rangle$.

The the first item sets the process-based strategy to allow for any environment action if the processes p are in Type-1 states; second item sets the process-based strategy to mimic the strategy τ if processes p are in Type-2 states. Recall that environment actions and system actions are enabled only from Type-1 states and Type-2 states respectively.

Since τ is a distributed strategy, for any $t \in \mathbb{M}(\Sigma, I)$ if $f_p(t) = \{(a, x)\}$ then for all $q \in \mathsf{dom}(a)$, $f_q(t) = \{(a, x)\}$. Therefore $\mathit{Plays}^\omega(\mathfrak{T}, \sigma)$ is well defined. Moreover,

$\theta \in \mathit{Plays}^{\omega}(\mathfrak{T}, \sigma)$ if and only if $\theta_{|\Sigma'}$ is a play that is consistent with the winning strategy τ. This implies that $\mathit{Plays}^{\omega}(\mathfrak{T}, \sigma) \subseteq \Theta$.

Hence we conclude that σ is a process-based controller for $(\mathfrak{T}, \Theta)$. ■

Lemma 5.8 *If $(\mathfrak{T}, \Theta)$ has a non-blocking process-based controller then there exists a winning, distributed strategy for Player 1 over the specification Θ'.*

Proof Starting from a process-based controller σ for $(\mathfrak{T}, \Theta)$, we define a strategy τ as follows:

$$\text{for } t' \in \mathbb{M}(\Sigma', I') \text{ assign } \tau(t') \colon \langle a \rangle \overset{\text{def}}{\mapsto} \langle x \rangle \text{ iff } (a, x) \in \bigcap_{p \in \mathsf{dom}(a)} f_p(t'_{\uparrow\Sigma} \odot a).$$

Since σ is a process-based controller, for every play $\theta' \in \mathbb{R}(\Sigma', I')$ that is consistent with τ, the trace $\theta'_{\uparrow\Sigma} \in \mathit{Plays}^{\omega}(\mathfrak{T}, \Theta)$. From the definitions, this implies that τ is a winning, distributed strategy for the game specification Θ'. ■

With the help of Lemmas 5.7 and 5.8, we can assert the following.

Theorem 5.9 *For any game Θ' over an independence alphabet $(\Sigma_0 \times \Sigma_1, I')$, one can construct an instance $(\mathfrak{T}, \Theta)$ over a suitable independence alphabet (Σ, I) such that there exists a winning, distributed strategy for Player 1 in Θ' if and only if there exists an process-based controller for $(\mathfrak{T}, \Theta)$.*

Finally, we complete the picture from Figure 5.1, with the help of Theorems 5.2, 5.6 and 5.9. We demonstrate that the distributed Church's synthesis problem in its unrestricted and restricted variants makes for the weakest and the strongest of classes of the distributed controller synthesis problem.

5.4 Discussion

The problem of distributed controller synthesis has been studied in various flavors, but in most of these flavors it is undecidable even for simple classes of instances whether there exists a distributed controller (see eg. [PR90, MT01]). In most of these cases, the discouraging undecidability arrives as a consequence of choosing plants and specifications in such a manner that a certain ordering of events is required by the specifications, but the processes of the plant are unable to synchronize to ensure this ordering. In other words, while the specification is over an independence alphabet (Σ, I), the plants are agnostic of the relation I.

A number of results have also appeared that restrict the plants or the specifications in a manner that makes the problem decidable. For example, very recently Finkbeiner-Schewe [FS13] and Fridman-Puchala [FP14] have characterized certain structural properties of the plants that are necessary for the distributed controller synthesis problem to be decidable. However, it boils down to near-trivial plant architectures for the reason cited above.

Most recently, Schewe [Sch14] has presented results regarding the undecidability of controller synthesis even for very simple recognizable ω-trace specifications, namely reachability and safety specifications described in simple temporal logics. Once again this appears discouraging, but it is not surprising. This is because plant architectures considered here are agnostic of the independence relation over the alphabet. Even if this dissociation were to be bridged, we still observe that it is hardly possible to construct weak asynchronous automata recognizing reachability and safety languages (see Proposition 4.2); and therefore, the outlook on synthesizing distributed safety controllers is unlikely to be any brighter.

In contrast to this, we are motivated by the observation that upon closely coupling the specifications and the plants with respect to the independence relation I, this problem becomes decidable for a number of interesting plant architectures. We find a specific solutions in [MT01, GLZ04, GLZ05, MTY05, MS13] (also see the references provided in [MWZ09]).

We therefore prefer the latter setting of considering recognizable ω-trace specifications because this is the closest to the classical settings for controller synthesis for sequential plants with specifications given as recognizable ω-languages. To advance this setting further, we have introduced here the definition of distributed Church's synthesis problem that is a generalization of that in the word case. Although we are no closer to deciding this problem for recognizable specifications, we have taken an important step in providing a perspective on the interrelationships of the various flavors – viz. process-based control, action-based control, and distributed Church's synthesis – that are pertinent to our preferred setting.

It remains a matter of further investigation whether one can demonstrate an equivalence between any of these problems, or obtain more refined reductions for each level of the Borel-like hierarchy that we have presented in the previous chapters. In the distributed Ramadge-Wonham setting, one can refine both at the level of the asynchronous transition systems and the specifications.

Turning to the subject of solution approaches for these problems, we believe that constructing distributed safety and reachability controllers will remain difficult for the very reason that makes the definition of reachability and safety asynchronous automata untenable – it is hard to decide global reachability by referring to local reachability. However, it might still be productive to consider synthesis problems for stronger specifications, like deterministic trace languages, which can be characterized in terms of deterministic asynchronous automata such as those presented in this thesis. It remains an area of future work to solve these problems by developing algorithms that are appropriate generalizations of the standard ones, for example, the ones for solving Büchi games over word languages.

Chapter 6

Concluding Remarks

The main contribution of this thesis is a two-pronged approach toward classifying recognizable ω-trace languages in a manner that closely resembles the Borel hierarchy of recognizable languages of ω-words. In the first approach, we introduce and build upon the definition of synchronization aware transition systems (SATS), which generalizes the notion of sequential transition systems. The SATS model provides a new insight into the behaviors of asynchronous automata by explicitly computing the amount of information exchanged at each synchronization. In the second approach, we investigate classes of ω-trace languages by means of classes of trace-closed ω-languages. We identify properties of ω-word languages that allow for efficient constructions of the matching deterministic ω-automata starting from automata over trace-closed finitary languages.

With respect to asynchronous automata, we present an algorithm to construct an SATS equivalent to any given asynchronous transition system. From the SATS model, we obtain a definition of deterministic, synchronization aware Büchi automata which generalizes the definition of deterministic Büchi automata over words: (a) it characterizes the class of deterministic trace languages; (b) this language class is closed under finite unions and intersections; and finally (c) this automaton model gives way to semi-decidability of deterministic trace languages in the manner of Landweber's celebrated result on deciding the levels of recognizable ω-languages. Furthermore, finite Boolean combinations of deterministic trace languages generates all recognizable ω-trace languages, which, as expected, are characterized by deterministic, synchronization aware Muller automata.

Once in the domain of trace-closed languages, we introduce the notions of I-suffix extension and I-limit-stability for finitary languages. These concepts are vital for obtaining trace-closed recognizable ω-languages that are recognized by specific classes of I-diamond deterministic weak and I-diamond deterministic Büchi automata respectively. In this setting, we obtain a full Borel-like hierarchy that classifies recognizable ω-trace languages by classifying trace-closed recognizable ω-languages. On the other hand, we do not yet know if it is decidable whether a given trace-closed recognizable ω-language is also recognized by an I-diamond deterministic Büchi automaton.

It must be duly noted that the classes of deterministic Büchi recognizable languages in the above two settings are strictly different – the former subsuming the latter. However, just like the former class, the class of I-diamond deterministic

Büchi recognizable trace-closed languages is expressive enough to describe all recognizable ω-trace languages.

At the end, we make an excursion into the field of the distributed controller synthesis problem. Here we recall the two distributed variants of Ramadge-Wonham controller synthesis problem that are well known in the literature. We then introduce the distributed Church's synthesis problem and compare these varieties. In the process, we enrich the results pertaining to the relative inclusions of the different variants of this problem.

Open Issues

The distributed controller synthesis problem, as considered in this thesis, has long resisted any attempts toward establishing its decidability. Progress in this direction was the main motivation for developing the ideas regarding classifications of recognizable ω-trace languages. It is now an endeavor to follow this trail, and to explore whether or not a structured approach for deterministic ω-automata can offer any insights into proving decidability or undecidability of distributed controller synthesis problem for Ramadge-Wonham and Church variants.

Independently of the synthesis problem, it has been left open in this thesis whether or not deterministic trace languages are a decidable subclass of recognizable ω-trace languages; although we have a weaker result confirming semi-decidability. Also, in the case of trace-closed languages, it is open whether or not I-diamond deterministic Büchi recognizable languages are a decidable subclass of trace-closed recognizable ω-languages. Clarifications of these questions will go a long way toward establishing the study of recognizable ω-trace languages as a generalization of the study of recognizable ω-languages.

Bibliography

[BE58] J. Richard Büchi and Calvin C. Elgot. Decision problems of weak second-order arithmetics and finite automata. 5(834), 1958.

[BL69] J. Richard Büchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.

[CG14a] Namit Chaturvedi and Marcus Gelderie. Classifying regular infinitary trace langauges using word automata. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *MFCS 2014: 39th International Symposium on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, pages 171–182. Springer-Verlag, 2014.

[CG14b] Namit Chaturvedi and Marcus Gelderie. Weak ω-regular trace languages. *CoRR*, abs/1402.3199, 2014.

[Cha14a] Namit Chaturvedi. Languages of infinite traces and deterministic asynchronous automata. Technical Report AIB-2014-04, RWTH Aachen University, February 2014.

[Cha14b] Namit Chaturvedi. Toward a structure theory of regular infinitary trace languages. In Elias Koutsoupias, Javier Esparza, and Pierre Fraigniaud, editors, *Proceedings of the 41st Colloquium on Automata, Languages, and Programming*, Lecture Notes in Computer Science, pages 134–145. Springer-Verlag, 2014.

[Chu57] Alonzo Church. Applications of recursive arithmetic to the problem of circuit synthesis. *Summaries of the Summer Institute of Symbolic Logic*, 1:3–50, 1957.

[CPP08] Olivier Carton, Dominique Perrin, and Jean-Éric Pin. Automata and semigroups recognizing infinite words. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives*, volume 2 of *Texts in Logic and Games*, pages 133–168. Amsterdam University Press, 2008.

[DM94] Volker Diekert and Anca Muscholl. Deterministic asynchronous automata for infinite traces. *Acta Informatica*, 31(4):379–397, 1994.

[DR95] Volker Diekert and Grzegorz Rozenberg, editors. *The Book of Traces.* World Scientific Publishing Co., River Edge, NJ, USA, 1995.

[EM93] Werner Ebinger and Anca Muscholl. Logical definability on infinite traces. In Andrzej Lingas, Rolf Karlsson, and Svante Carlsson, editors, *Automata, Languages and Programming*, volume 700 of *Lecture Notes in Computer Science*, pages 335–346. Springer-Verlag, 1993.

[FP14] Wladimir Fridman and Bernd Puchala. Distributed synthesis for regular and contextfree specifications. *Acta Informatica*, 51(3-4):221–260, 2014.

[FS13] Bernd Finkbeiner and Sven Schewe. Bounded synthesis. *International Journal on Software Tools for Technology Transfer*, 15(5-6):519–539, 2013.

[GLZ04] Paul Gastin, Benjamin Lerman, and Marc Zeitoun. Distributed games and distributed control for asynchronous systems. In Martín Farach-Colton, editor, *LATIN 2004: Theoretical Informatics*, volume 2976 of *Lecture Notes in Computer Science*, pages 455–465. Springer-Verlag, 2004.

[GLZ05] Paul Gastin, Benjamin Lerman, and Marc Zeitoun. Distributed games with causal memory are decidable for series-parallel systems. In Kamal Lodaya and Meena Mahajan, editors, *FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science*, volume 3328 of *Lecture Notes in Computer Science*, pages 275–286. Springer-Verlag, 2005.

[GP92a] 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-Verlag, 1992.

[GP92b] Paul Gastin and Antoine Petit. A survey of recognizable languages of infinite traces. In Grzegorz Rozenberg, editor, *Advances in Petri Nets 1992*, volume 609 of *Lecture Notes in Computer Science*, pages 392–409. Springer-Verlag, 1992.

[Löd01] Christof Löding. Efficient minimization of deterministic weak ω-automata. *Information Processing Letters*, 79(3):105 – 109, 2001.

[Mad12] 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 Publishing Co., 2012.

[Maz87] 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.

[McN66] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. 9(5):521–530, 1966.

[MS13] Anca Muscholl and Sven Schewe. Unlimited decidability of distributed synthesis with limited missing knowledge. In Krishnendu Chatterjee and Jirí Sgall, editors, *Mathematical Foundations of Computer Science 2013*, volume 8087 of *Lecture Notes in Computer Science*, pages 691–703. Springer-Verlag, 2013.

[MT01] P. Madhusudan and P.S. Thiagarajan. Distributed controller synthesis for local specifications. In Fernando Orejas, PaulG. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming*, volume 2076 of *Lecture Notes in Computer Science*, pages 396–407. Springer-Verlag, 2001.

[MTY05] P. Madhusudan, P.S. Thiagarajan, and Shaofa Yang. The mso theory of connectedly communicating processes. In Sundar Sarukkai and Sandeep Sen, editors, *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science*, volume 3821 of *Lecture Notes in Computer Science*, pages 201–212. Springer-Verlag, 2005.

[Mus94] Anca Muscholl. *Über die Erkennbarkeit unendlicher Spuren.* PhD thesis, Universität Stuttgart, 1994.

[MWZ09] Anca Muscholl, Igor Walukiewicz, and Marc Zeitoun. A look at the control of asynchronous automata. In Kamal Lodaya, Madhavan Mukund, and R Ramanujam, editors, *Perspectives in Concurrency Theory*. Universities Press, 2009.

[PP04] Dominique Perrin and Jean-Éric Pin. *Infinite Words: Automata, Semigroups, Logic and Games*, volume 141 of *Pure and Applied Mathematics*, chapter Automata and Infinite Words. Elsevier, 2004.

[PR90] Amir Pneuli and Roni Rosner. Distributed reactive systems are hard to synthesize. In *Foundations of Computer Science, 1990. Proceedings., 31st Annual Symposium on*, pages 746–757. IEEE, 1990.

[Rab69] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. 141:1–35, 1969.

[RW89] Peter J.G. Ramadge and W. Murray Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[Sch14] Sven Schewe. Distributed synthesis is simply undecidable. *Information Processing Letters*, 114(4):203–207, 2014.

[Sta83] Ludwig Staiger. Subspaces of $GF(q)^{\omega}$ and convolutional codes. *Information and Control*, 59(1-3):148–183, 1983.

[Tho90a] Wolfgang Thomas. Automata on infinite objects. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 133–192. Elsevier, 1990.

[Tho90b] Wolfgang Thomas. Logical definability of trace languages. In Volker Diekert, editor, *Proceedings of the ASMICS-Workshop "Free Partially Commutative Monoids"*, pages 172–182. TU München, 1990. Rep. TUM-I9002.

[Tho97] Wolfgang Thomas. Languages, automata, and logic. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 389–455. Springer-Verlag, 1997.

[Tho08] Wolfgang Thomas. Church's problem and a tour through automata theory. In Arnon Avron, Nachum Dershowitz, and Alexander Rabinovich, editors, *Pillars of computer science*, pages 635–655. Springer-Verlag, 2008.

[Zie87] Wieslaw Zielonka. Notes on Finite Asynchronous Automata. *R.A.I.R.O. – Informatique théorique et applications*, 21:99–135, 1987.

Index of Definitions