Fakultät für Informatik
Technische Universität München

A Unified Translation of Linear Temporal Logic to ω-Automata

Ben Salomon Maria Sickert

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:
Prof. Tobias Nipkow, Ph.D.

Prüfende der Dissertation:
1. Prof. Dr. Francisco Javier Esparza Estaun
2. Prof. Orna Kupferman, Ph.D.,
   The Hebrew University of Jerusalem

Die Dissertation wurde am 30.04.2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 13.07.2019 angenommen.

Abstract

One of the predominant specification languages for reactive systems is linear temporal logic (LTL) which is used to describe infinite traces of these systems. The leading techniques for verification and synthesis of reactive systems using LTL as a specification language are automata-based. The automata-theoretic approach fundamentally relies on the translation of specifications given in LTL into a suitable type of ω-automaton. These translations and the resulting automata are a major factor for the applicability of these automata-based algorithms: the number of states, the type and the branching degree of the transition relation, the type and the size of the acceptance condition, and the representation of the automata all influence the runtime of these algorithms and thus the optimisation of such factors is an important goal.

Nondeterministic Büchi automata (NBA) gained considerable popularity amongst the different types of ω-automata, since they elegantly balance a simple acceptance condition with fairly efficient algorithms and are still expressive enough to capture all ω-regular languages. Consequently, most of the effort has been spent on translating to NBAs and the optimisations of these constructions. However, the simplicity of the Büchi condition is inseparably linked to the nondeterminism: this makes NBAs unsuitable for some algorithms used for verification of probabilistic systems or synthesis. Thus NBAs are often just an intermediate step, followed by additional constructions, on the way to the desired automaton model. But this detour comes at a price. Since these subsequent steps operate on NBAs, they cannot take advantage of the formula structure and the fact that LTL only captures a subset of the ω-regular languages.

We present a unified translation of LTL formulas into nondeterministic Büchi automata, limit-deterministic Büchi automata (LDBA), and deterministic Rabin automata (DRA). The translations yield automata of asymptotically optimal size (double or single exponential, respectively). All three translations are derived from one single Master Theorem of purely logical nature. The Master Theorem decomposes the language of a formula into a positive Boolean combination of languages that can be translated into ω-automata by elementary means. All three translations are direct and no intermediate constructions are used. In particular, Safra's construction, ranking constructions, and breakpoint constructions used in other translations are not needed. Further, states are semantically labelled, meaning the language of each state can be derived as an LTL formula from the state label. Lastly, the special structure of the constructed LDBAs make them suitable for the verification of probabilistic models and for the synthesis of reactive systems.

Zusammenfassung

Einer der vorherrschenden Formalismen für die Spezifikation reaktiver Systeme ist lineare temporale Logik (LTL), die benutzt wird um unendliche Abläufe solcher Systeme zu beschreiben. Die führenden Techniken für die Verifikation und die Synthese reaktiver Systeme mit LTL als Spezifikation sind automatenbasiert. Der automatentheoretische Ansatz hängt essentiell von der Übersetzung der LTL Spezifikation in ein passendes ω-Automatenmodell ab. Die Übersetzungen und die daraus resultierenden Automaten sind ein ernstzunehmender Faktor für die Anwendbarkeit der automatenbasierten Algorithmen: die Anzahl der Zustände, die Art und der Verzweigungsgrad der Übergangsrelation, die Art und die Größe der Akzeptanzbedingung und die Darstellung der Automaten, all diese Faktoren beeinflussen die Laufzeit der Algorithmen und deshalb sind Optimierungen dieser Eigenschaften ein wichtiges Ziel.

Nichtdeterministische Büchi-Automaten (NBA) haben eine beträchtliche Popularität unter den verschiedenen Arten von ω-Automaten erreicht, da sie auf eine elegante Art und Weise eine simple Akzeptanzbedingung mit relativ effizienten Algorithmen vereinen und dabei immer noch ausdrucksstark genug bleiben, so dass sie alle ω-regulären Sprachen darstellen können. Daher ist ein Großteil des wissenschaftlichen Aufwands in die Entwicklung und die Verbesserung von Verfahren, die LTL zu NBAs übersetzen, geflossen. Jedoch ist die Schlichtheit der Büchi-Bedingung untrennbar mit dem Nichtdeterminismus verbunden und dies macht NBAs ungeeignet für einige Algorithmen, die für die Verifikation von probabilistischen Systemen oder für die Synthese verwendet werden. Deshalb sind NBAs oft nur ein Zwischenschritt auf dem Weg zu dem eigentlich verwendetem Automatenmodel. Da die Konstruktionen, die der LTL-Übersetzung folgen, auf nichtdeterministischen Büchi-Automaten arbeiten, können sie nicht die Struktur der Formel und den Fakt, dass LTL nur einen Teil der ω-regulären Sprachen erkennt, zu ihrem Vorteil nutzen.

Wir präsentieren eine uniforme Übersetzung von LTL Formeln zu nichtdeterministischen Büchi-Automaten (NBA), limit-deterministischen Büchi-Automaten (LDBA) und deterministischen Rabin-Automaten (DRA). Die Übersetzungen erzeugen Automaten von asymptotisch optimaler Größe (doppelt bzw. einfach exponentiell). Alle drei Übersetzungen werden aus einem einzigen Master Theorem abgeleitet. Das Master Theorem zerlegt die Sprache einer Formel in eine Boolesche Kombination von Sprachen, die mit elementaren Techniken zu ω-Automaten übersetzt werden können. Alle drei Übersetzungen sind direkt und benötigen keine Zwischenkonstruktionen. Insbesondere sind Ansätze, wie die „Safra“-Konstruktion, „Ranking“-Konstruktionen oder „Breakpoint“-Konstruktionen, wie sie normalerweise in anderen Übersetzungen verwendet werden, nicht notwendig. Darüberhinaus sind die Zustände mit semantischen Informationen versehen, so dass die Sprache jedes Zustandes als LTL Formel abgeleitet werden kann. Die spezielle Struktur der konstruierten limit-deterministischen Automaten macht diese auch nutzbar für die Verifikation von probabilistischen Systemen und für die Synthese reaktiver Systeme.

Preface

I want to express my sincere gratitude to my doctoral advisor Javier Esparza for his invaluable guidance and support. I always could rely on his mentorship and helpful advice. I am also thankful to my undergraduate advisor Jan Křetínský for the productive academic discourse with him and for my early introduction to the world of research. His work on LTL translations paved the way for the results presented in this dissertation. Further, I want to thank Orna Kupferman and Jean-François Raskin for hosting me in Jerusalem and Brussels. The trip to Jerusalem, on which I accompanied Javier to meet Orna and Moshe Vardi, gave me the initial ideas and insights that made the following results possible.

Over the years I had the pleasure of working with researchers from various institutions on projects related to my dissertation. I am thankful for these collaborations and the things I learned from my co-authors, namely: Julian Brunner, Javier Esparza, Stefan Jaax, Jan Křetínský, Michael Luttenberger, Tobias Meggendorfer, Philipp Meyer, David Müller, Jean-François Raskin, and Benedikt Seidl. Moreover, I want to thank my colleagues at the chair for the great atmosphere and the good times. A special thank you goes to all colleagues that proofread bits and pieces of my dissertation and gave my valuable feedback.

The moral support and the backing from my friends and my family was as essential as the academic help and mentorship I received. I feel gratitude towards all those people who accompanied me on this journey. Finally, I have to thank Tyche making me cross paths with Micha who managed to catch a thief.

Contents

1 Introduction

Logics and automata theory, two fundamental fields of theoretical computer science, are remarkably intertwined. On the one hand, we use logical frameworks as a natural way to reason about computations and their properties. On the other hand, automata with their clear operational focus are a well-suited abstraction for computations themselves. Even though they look remote to each other at first sight, it has been shown time and time again that they are closely related: One of the earliest results that fall into this category is due to Büchi. In his seminal work he used automata to characterise the expressive power of monadic second-order logic with 1 successor (S1S) [Büc60; Büc66] and settle the decidability of the logic. In the following decades more results connecting automata theory and logics on other structures were discovered, e.g. see [Lee90] for some examples.

Verification and Synthesis. Software, hardware circuits, and more generally computer systems often exhibit erroneous behaviour. Usually this is not detected during the development of the system, and therefore it is necessary to spend a considerable amount of resources for debugging and testing such systems. Nevertheless, these steps do not guarantee correctness in general. Thus two fundamental questions arise when we build systems that need to be reliable and are mission-critical:

First, the *verification problem*: Given a program, e.g. represented as state machine, and a specification, e.g. expressed in a suitable logic, decide if the behaviour of the program satisfies the specification. This idea can be traced back to Turing, who investigated the halting problem, which is termination of a computation, on Turing machines [Tur37].

Second, the *synthesis problem*, already defined by Church and Kleene [Kle56; Chu57]: Given a specification find an implementation, e.g. a circuit or a finite state machine, that satisfies the specification, provided such an implementation exists. This goes a step further than the verification problem. A synthesis procedure frees the implementor from building the system and it is only necessary to come up with a description of the desired behaviour. However, in practice just writing a complete specification is already a difficult task in itself.

Note that we abstained in our descriptions from fixing a logic or representation of programs. In fact the hardness and complexity of the two problems greatly depend on the chosen formalisms. Initially the focus for the verification problem laid on proof-like systems: Sequential programs were analysed line-by-line using annotations with pre- and postconditions. This so called Floyd-Hoare paradigm [Flo67; Hoa69] was an immense intellectual success for reasoning about programs. However, it requires at some steps manual interaction for constructing the proof, and thus poses a scalability issue for large programs. Further, it does not translate well to *reactive systems*. Reactive systems are non-terminating, typically concurrent, programs that interact with an open environment. Examples of reactive systems are circuits, resource arbiters, or distributed protocols. This nonterminating, infinite behaviour make them incompatible with the Floyd-Hoare paradigm. *Temporal logics*, by contrast, focus on the behavioural aspects of the system,

and thus overcome some of the shortcomings of the Floyd-Hoare approach, as these logics made it possible to reason about the infinite nature of such systems. Algorithmic progress was then achieved due to the small model property, i.e. the model of a formula can be represented by a 'small' and finite structure, that holds for several of these temporal logics and the term *model checking* [CE81; EC82; CHVB18] was coined. This instance of the verification problem usually only considers programs abstracted to finite state graph and logics that are effectively and efficiently decidable.

Let us now focus on temporal logics, which come in two flavours: linear and branching time. In the linear-time setting we have a single timeline and the logic can only talk about one future. By contrast, in the branching-time setting we can reason about different possible futures. The most prominent logics of these two categories are Linear Temporal Logic (LTL) [Pnu77; GPSS80; Pnu81] and Computation Tree Logic (CTL) [CE81]. An in-depth discussion and a detailed overview on different temporal logics can be found in [Eme90]. LTL and in particular its propositional variant gained immense popularity, since it finds a good balance between expressiveness, and algorithmic complexity, and is similar enough to natural language.

Automata-Theoretic Approach. The predominant techniques for automatic verification and synthesis of reactive systems are automata-based and follow the same scheme: First, the specification, given in a temporal logic, is translated to an automaton; second, the graph abstraction of the reactive system and the automaton for the specification are combined and analysed, e.g. via an intersection and an emptiness check. In the synthesis case only the specification automaton is processed. The automata-theoretic approach [VW86a] proved to be so successful because it separates the logic and its interpretation from the graph-related questions. This approach is effectively robust to changes in the logic, since most of time only the translation step needs to be adapted.

Depending on the inputs (reactive system and specification) we might have different requirements on the resulting automaton for the specification. For example, model checking probabilistic systems, such as Markov chains or Markov decision processes (MDP), the specification automaton typically needs to be deterministic (or 'almost' deterministic). Thus a translation from LTL to a nondeterministic automaton does not suffice and additional steps are required to obtain an automaton with suitable properties. However, all extra steps incur extra costs, e.g. in the size of the automata or in the running time, and since almost all algorithms explicitly or implicitly construct a product automaton, the size of the specification automaton is a crucial factor. Hence optimised and efficient translations from the logic to the specification automaton are paramount to the success of these approaches. Therefore, a substantial amount of research has gone into the translation of LTL to automata, which we will summarise in the next section. We then outline the main contribution of this thesis: a logical characterisation of LTL from which compositional constructions for a wide variety of different automata are derived.

1.1 A Short History of LTL Translations

The Automata Zoo. In the realm of regular languages over finite words we differentiate automata by their branching mode, e.g. nondeterministic, deterministic, universal, or alternating. However, the acceptance condition is always the same: a word is accepted if depending on the branching mode a sufficient amount of runs end up in a final state.

Automata over infinite words inherit this differentiation, but add a new dimension to it. For ω-automata the choice of the acceptance condition, e.g. Büchi, co-Büchi, Rabin, Streett, and parity[1], is crucial and can make a difference in expressiveness and size. The Büchi automaton can be seen as the archetype of ω-automata with its simple acceptance structure: a run has to visit at least one accepting state infinitely often. However, this simplicity comes at a price: nondeterminism is required to be able to recognise the whole class of ω-regular languages, and therefore deterministic Büchi automata (DBA) are strictly less expressive than nondeterministic Büchi automata (NBA).

While NBAs can be used for model checking nondeterministic systems, other applications such as model checking probabilistic systems or synthesis usually require automata with a certain degree of determinism, such as deterministic parity automata (DPA) [PR89] or deterministic Rabin automata (DRA) [BK08; CGK13], limit-deterministic Büchi automata (LDBA) [Var85; CY95; HLS+15; SEJK16], or unambiguous Büchi automata [BKK+16]. The essential point here is that there exists no automaton model that is suitable for all applications and it is unavoidable to have different models with different trade-offs and translations.

The Central Hub: Nondeterministic Büchi Automata. The usual solution to this multiplicity of choice is to define translations from LTL to nondeterministic Büchi automata and then use further automata conversions to arrive at the desired automaton model. However, this approach has the inherent weakness that LTL cannot express all ω-regular languages, while NBAs can, and thus the subsequent conversions might have to solve a harder problem.

The discovery of a translation of asymptotical optimal size from LTL to NBAs [VW86a][2] together with the automata-theoretic approach established the central role of NBAs. Since the characteristics, e.g. size and representation, of the resulting NBA has a direct effect on the subsequent steps, a lot of effort has been spent on finding translations with better practical characteristics, e.g. smaller size [GPVW95; Cou99; DGV99; EH00; SB00; GO01; GL02; Fri03; BKRS12; DLF+16], symbolic representation [Sch01], or a compositional structure [KPR98; PZ08]. Interestingly, it was discovered that LTL and very-weak alternating co-Büchi automata (VWAA) are closely related – they capture the same class of languages – and the translation from LTL to VWAA is straight-forward [MSS88; Var94]. This in combination with the classic result of [MH84] also yields a LTL to NBA translation. In fact [GO01] follows this approach and adds several optimisations on top of it.

As noted before we sometimes need to apply a conversion step to arrive at the desired automaton class. The most studied conversion in this context is determinisation: The first translation [McN66] from NBAs to deterministic ω-automata (with a different acceptance condition) dates back to the 60's. However, this construction left an exponential complexity gap open. This gap was closed by Safra with an asymptotic optimal construction [Saf88], which has been refined several times, e.g. in [Pit07; Sch09]. In parallel, conceptually different approaches to determinisation have been proposed by [MS95; KW08; FKVW15]. Interestingly, [LP19] unifies the constructions from Safra and Muller-Schupp into a single meta-construction. So far we talked about determinisation constructions that are correct for all NBAs. However, by restricting to certain subclasses,

[1] For a definition of these different acceptance types we refer to reader to Chapter 2.

[2] In fact the translation presented there is a special case of the technique shown in [WVS83].

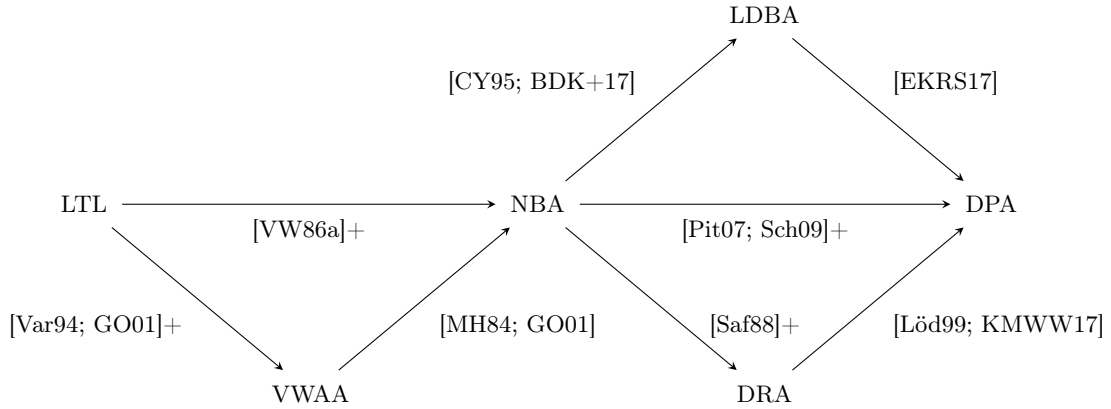


Figure 1.1: Map of non-trivial translations centred around NBAs. Note that we do not distinguish between state- and transition-based acceptance and between generalised and non-generalised acceptance conditions, e.g. a generalised nondeterministic Büchi automaton is contained in the group NBA. We indicate by a '+' if we left out some references appearing in the text.

simpler constructions are possible, e.g. the construction of [MS08] operates on confluent NBAs. Alternatively, one can also consider fragments of LTL such that a translation to deterministic automata is simpler, as implied by [KV01].

Continuing on the theme of relaxing requirements: for qualitative probabilistic model checking it is not necessary to use deterministic automata and limit-deterministic suffice [Var85; CY95].[3] Thus we can use translations presented in [CY95; BDK+17] to obtain LDBAs from NBAs. Similarly, for some scenarios it is enough to consider unambiguous [BKK+16] or Good-for-Games automata [HP06; KMBK14]. Finally, one can also translate LDBAs to DPAs [EKRS17] and DRAs to DPAs [Löd99; KMWW17]. An overview of these translations is depicted in Figure 1.1.

Short-Cuts Around the Central Hub. Even though determinisation procedures were successfully implemented for some applications such as probabilistic model checking [KNP11], the situation remains unsatisfactory, since the combination of LTL-to-NBA translations and of determinisation constructions does not scale well on larger specifications in practice. This is believed to be due to the complexity of the state structure that is well-known to make efficient implementation difficult, e.g. [KPV06]. This poses a challenging problem, since specifications for interesting and non-trivial properties tend to grow quite fast and the corresponding deterministic automata become quickly too large. Consequently, the research community designed more efficient translations from LTL to deterministic [KE12; KL13; BBKS13; EK14; EKS16; MS17; EKS18] or limit-deterministic automata [KV15; SEJK16; KV17; EKS18], skipping the intermediate step through NBAs. It is has to be said that the early constructions of this series only apply to 'simple' fragments of LTL and a complete translation was elusive at that time. One can also derive deterministic parity automata (DPA) using the construction of [EKRS17] that exploits the special structure of [SEJK16] to obtain small deterministic automata.

[3] [SEJK16] shows how to use a special family of LDBAs for quantitive probabilistic model checking, see also Chapter 9.

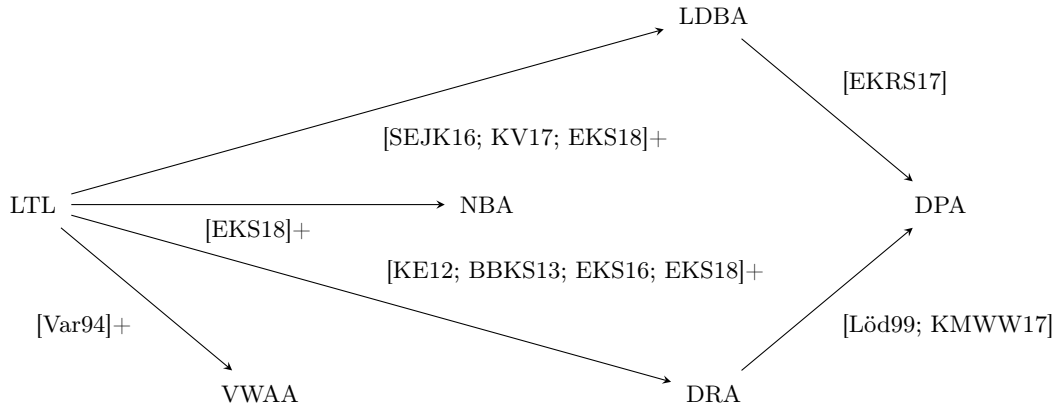


Figure 1.2: Map of direct translations. Note that we do not distinguish between state- and transition-based acceptance and between generalised and non-generalised acceptance conditions, e.g. a generalised Rabin automaton is contained in the group DRA. We indicate by a '+' if we left out some references appearing in the text.

This translation is used with great success in [MSL18]. A map relating these translations is displayed in Figure 1.2.

1.2 A Unified Translation of LTL to ω-Automata

The contribution of this thesis is a unified approach to the translation of LTL to NBAs, LDBAs, and DRAs enjoying the following properties that are absent in former translations: They are based on a unified logical framework, use semantic state labels, have asymptotic optimality, are symmetric, are independent of syntactic restrictions, and have practical relevance. The content of this thesis is a substantially extended version of our results found in [EKS18] and rephrases our earlier results found in [SEJK16].

Unified Approach. Our translations rely on a novel *Master Theorem*, which decomposes the language of a formula into a positive boolean combination of 'simple' languages[4], in the sense that they are easy to translate into automata. This approach is arguably simpler than previous ones [EKS16; SEJK16]. Moreover, it provides a unified treatment of DRAs, NBAs, and LDBAs, differing only in the translations of the 'simple' languages. The automaton for the formula is obtained from the automata for the 'simple' languages by means of standard operations for closure under union and intersection. It can be thought of as a black box, where the user of the theorem provides translations to the target language for simple fragments of LTL and obtains a compositional translation for all LTL formulas.

Semantic Translation: Rediscovering Derivatives. Regular-expression derivatives are an elegant, but seemingly forgotten, technique for translating regular expressions to deterministic [Brz64] as well as nondeterministic [Ant96] finite automata. In order to

[4]Some of the languages can be attributed to the 'recurrence', 'persistence', and 'safety' class of LTL according to [MP90; Sis94].

distinguish the two types of derivates, the ones used for constructing nondeterministic automata are called *partial* derivatives. The translations we present for the 'simple' languages are conceptually similar to these derivative constructions: States are essentially LTL formulas and the successors are computed locally from the state-labelling formula. The important point is that the language of a product state in the compositional construction can be derived as an LTL formula by the simple means of conjunction and disjunction. This is the pre-requisite for efficient semantics-based state reductions. For example, based on this information states corresponding to equivalent formulas can be merged. Note that these semantic-based reductions cannot be applied in general to Safra-based constructions, because this semantic structure gets lost in translation. Furthermore, this makes it easier to define a symbolic variant of the translation.

There have been several attempts to apply the idea of derivatives to ω-regular expressions in the deterministic setting [Red99; Red12], but it seems that derivatives resist a simple generalisation to an infinite setting [Par81] and major additional structure is necessary for a correct construction. However, partial derivatives have been adapted to ω-regular expressions [TS15] with more success and in [ST18] well-known results for LTL were reformulated using partial derivatives. In contrast, we apply the derivative constructions only to fragments of LTL, where they can be easily be applied without additional structure, and we then delegate the decomposition into the simple fragments to the Master Theorem.

Asymptotic Optimality. Deterministic generalised Rabin automata are the most compact among the deterministic automata used in practice[5], in particular compared to DPA. Previous translations to D(G)RA were either limited to fragments of LTL [KE12; KL13; BBKS13], or only shown to be triply exponential [EK14; EKS16]. Here we provide constructions for all mentioned types of automata matching the asymptotic double exponential size for D(G)RAs and LDBAs, and the exponential size for NBAs. Thus the construction we present here is the first direct translation from LTL to D(G)RAs with a proven double exponential size bound.

Symmetry. The first direct translations [KE12; KL13] to deterministic automata used auxiliary automata to monitor each **F**- (*Finally*) and **G**- (*Globally*) subformula. While this approach worked for fragments of LTL, subsequent constructions for full LTL [EK14; EKS16; SEJK16] could not preserve the symmetric treatment, thus we consider these constructions to be *asymmetric*. They only used auxiliary automata for **G**-subformulas, at the price of more complex constructions, e.g. break-point constructions similar to [MH84]. Our translation re-establishes the symmetry and it treats **F** and **G** equally (actually, and more generally, it treats each operator and its dual equally), which results into simpler automata constructions.

Independence of Syntax. Previous translations were quite sensitive to the operators used in the syntax of LTL. In particular, the only greatest-fixed-point operator they allowed was **G** and operators such as **R** (*Release*) needed to be removed. Since formulas also had to be in negation normal form, pre-processing of the input often led to unne-

[5] In theory Emerson-Lei automata can be more succinct [MS17], but they are not yet used widely in practice.

cessarily large formulas. While our translations still requires negation normal form, it allows for direct treatment of **R**, **W** (*Weak Until*), and other operators.

Practical Relevance. On top of its theoretical advantages, our translation is comparable to previous NBA, LDBA, and DRA translations in practice.

Summarising, we think this work finally achieves the goals formulated in [KE12], where the first translation of this kind – valid only for what is in comparison only a small fragment of LTL – was presented.

1.3 Structure of the Thesis

The thesis is organised as follows: Chapter 2 introduces fundamental terminology and results about ω-automata and LTL. Chapter 3 lays the theoretical groundwork for the derivative-based constructions for fragments of LTL based on the 'after'-function from [EK14; EKS16; SEJK16]. Chapter 4 presents the central result – the Master Theorem – that decomposes LTL formulas into simpler fragments. Chapters 5 and 6 derive from the Master Theorem the promised automata constructions. These chapters follow the same structure. First, we illustrate how to use the results from Chapter 3 to translate four simple fragments of LTL to deterministic automata and nondeterministic automata, respectively. Second, these translations are then used as building blocks to obtain DRAs, LDBAs, and NBAs, respectively. Chapter 7 introduces a collection of possible optimisations to the presented constructions that are relevant for using these translations in practice. Chapter 8 compares experimentally the newly defined translations with other approaches and Chapter 9 discusses two possible applications for the proposed constructions. In Chapter 10 we look at open questions and further ideas.

1.4 Preceding Publications

This dissertation is a comprehensive presentation of the results published in [EKS18], but extends that material considerably. Thus several chapters are in some parts textually based on [EKS18], but have been adapted to fit in the general presentation. The

Publication	[EKS18]	[EKS16]	[SEJK16]	[LMS19]
Chapter	Abstract, 1 - 6, 10	3	9	9

(a) Publications by the author containing results described in this dissertation and which served as a textual basis.

Publication	[KMS18]	[KMSZ18]	[SK16]	[MSL18]	[LMS19]
Chapter	8	8	9	9	9

(b) Publications by the author describing implementations that either have been used in a chapter to perform experiments or contain implementations of the proposed applications.

Table 1.1: Preceding publications.

most notable changes and additions are: the reduction of the LTL syntax; the introduction of *af*- and $\cdot[\cdot]_\nu$-congruences to make it clear which equivalence relations can be used for constructing deterministic automata; a restricted version of the Master Theorem to remove redundant guesses; a completely revised NBA construction; the inclusion of correctness proofs or proof sketches for all constructions; an extensive list of optimisation for constructions derived from the Master Theorem; a detailed experimental evaluation; and a adaption of the results from [SEJK16] to the LDBAs constructed through the Master Theorem. Thus this dissertation contains text fragments and a selection of results from other publications of the author. Furthermore, the experiments presented in Chapter 8 rely on [KMS18; KMSZ18] and the applications outlined in Chapter 9 have been implemented in [SK16; MSL18; LMS19]. All implementations described in [KMS18; KMSZ18; SK16; MSL18; LMS19] have been either developed or co-developed by the author. Tables 1.1a and 1.1b give an overview of the relation between each chapter and each publication.

2 Preliminaries

Let $\sim$ be an equivalence relation – a reflexive, symmetric, and transitive binary relation – over a fixed set X. The *equivalence class* of an element $x \in X$ is denoted $[x]_\sim$ and defined as $[x]_\sim = \{y \in X : x \sim y\}$. The *quotient set* of a subset $Y \subseteq X$ is denoted $Y_{/\sim}$ and defined as $Y_{/\sim} = \{[y]_\sim : y \in Y\}$. Let $\sim$ and $\approx$ be two equivalence relations over the same set X. We write $\sim \preccurlyeq \approx$ if $x \sim y$ implies $x \approx y$ for all x, y in X. We write $\sim \prec \approx$ if $\sim \preccurlyeq \approx$ and $\sim \neq \approx$. Further let $f : X \to X$ be a function on the set X. If for each $x, y \in X$ with $x \sim y$ the function f satisfies $f(x) \sim f(y)$, then there exists the *canonical lifting* $f_\sim : X_{/\sim} \to X_{/\sim}$ defined as $f_\sim([x]_\sim) = [f(x)]_\sim$.

2.1 ω-Languages and ω-Automata

Let Σ be a finite alphabet. An ω-word w over Σ is an infinite sequence of letters $a_0 a_1 a_2 \dots$ with $a_i \in \Sigma$ for all $i \geq 0$, and an ω-language is a set of ω-words. The set of all ω-words is denoted Σ^ω. We denote the i-th letter of an ω-word w (starting at 0) by $w[i]$. The finite infix $w[i]w[i+1]\dots w[j-1]$ is abbreviated with w_{ij} and the infinite suffix $w[i]w[i+1]\dots$ with w_i. The following identities are useful when working with ω-words: $w_{i(i+1)} = w[i]$, $w_{(i+j)i} = \epsilon$ and $w_{0i}w_i = w$ for all $i, j \geq 0$. Finally, we denote the infinite repetition of a finite word $\sigma_1 \dots \sigma_n$ by $(\sigma_1 \dots \sigma_n)^\omega = \sigma_1 \dots \sigma_n \sigma_1 \dots \sigma_n \sigma_1 \dots$.

For the sake of presentation we focus on ω-automata with acceptance conditions defined on states rather than on transitions. Since nowadays the preferred type of acceptance condition for implementations is transition-based [GL02; KK14; BBD+15; DLF+16], we sketch in Chapter 7 how to adapt the presented constructions to the transition-based setting.

Let Σ be a finite alphabet. A *nondeterministic pre-automaton* over Σ is a tuple $\mathcal{P} = (Q, \Delta, Q_0)$ where Q is a finite set of states, $\Delta\colon Q \times \Sigma \to 2^Q$ is a transition function, and Q_0 is a set of initial states. A transition is a triple (q, a, q') such that $q' \in \Delta(q, a)$. Depending on the context we might also view $\Delta \subseteq Q \times \Sigma \times Q$ as a relation, which is an equivalent description. A pre-automaton $\mathcal{P}$ is deterministic if Q_0 is a singleton and $\Delta(q, a)$ is a singleton for every $q \in Q$ and every $a \in \Sigma$.

Acceptance Conditions over States. A *state-run* of $\mathcal{P}$ on an ω-word w is an infinite sequence of states $r = q_0 q_1 q_2 \dots$ such that $q_0 \in Q_0$ and $q_{i+1} \in \Delta(q_i, w[i])$ for all i. We denote the set of states occurring infinitely often in r by $\mathit{inf}(r)$. An *acceptance condition on states* is an expression over the syntax:

$$\alpha ::= \mathit{inf}(S) \mid \mathit{fin}(S) \mid \alpha_1 \vee \alpha_2 \mid \alpha_1 \wedge \alpha_2 \quad \text{with } S \subseteq Q$$

Acceptance conditions are evaluated on runs and the satisfaction relation $r \models \alpha$ is defined as follows:

$$\begin{array}{lll} r \models \mathit{inf}(S) & \mathit{iff} & \mathit{inf}(r) \cap S \neq \emptyset \\ r \models \mathit{fin}(S) & \mathit{iff} & \mathit{inf}(r) \cap S = \emptyset \\ r \models \alpha_1 \vee \alpha_2 & \mathit{iff} & r \models \alpha_1 \ \mathit{or}\ r \models \alpha_2 \\ r \models \alpha_1 \wedge \alpha_2 & \mathit{iff} & r \models \alpha_1 \ \mathit{and}\ r \models \alpha_2 \end{array}$$

An acceptance condition α is a

- Büchi condition if $\alpha = \mathit{inf}(S)$ for some set $S \subseteq Q$ of states.
- co-Büchi condition if $\alpha = \mathit{fin}(S)$ for some set $S \subseteq Q$ of states.
- generalised Büchi condition if $\alpha = \bigwedge_{i=1}^{k} \mathit{inf}(I_i)$ for some $k \geq 1$ and some sets $I_1, I_2, \ldots, I_k \subseteq Q$ of states.
- Rabin condition if $\alpha = \bigvee_{i=1}^{k} (\mathit{fin}(F_i) \wedge \mathit{inf}(I_i))$ for some $k \geq 1$ and some sets F_1, $I_1, \ldots, F_k$, $I_k \subseteq Q$ of states.
- generalised Rabin condition if $\alpha = \bigvee_{i=1}^{k} (\mathit{fin}(F_i) \wedge \bigwedge_{j=1}^{l_i} \mathit{inf}(I_{ij}))$ for some k, l_1, $l_2, \ldots, l_k \geq 1$ and some sets $F_1, I_{1,1}, I_{1,2}, \ldots, I_{1,l_1}, \ldots, F_k, I_{k,1}, \ldots I_{k,l_k} \subseteq Q$ of states.

An ω-automaton over Σ is a tuple $\mathcal{A} = (Q, \Delta, Q_0, \alpha)$ where (Q, Δ, Q_0) is a pre-automaton over Σ and α is an acceptance condition. A state-run r of $\mathcal{A}$ is *accepting* if $r \models \alpha$. A word w is accepted by $\mathcal{A}$ if some run of $\mathcal{A}$ on w is accepting. The language $\mathcal{L}(\mathcal{A})$ of an automaton $\mathcal{A}$ is defined as the set $\mathcal{L}(\mathcal{A}) \coloneqq \{w \in \Sigma^\omega : w \text{ is accepted by } \mathcal{A}\}$. An ω-automaton is a (generalised) Büchi (co-Büchi, Rabin) automaton if its acceptance condition is a (generalised) Büchi (co-Büchi, Rabin) condition.

Acceptance Conditions over Transitions. We only need to slightly change the previous definitions. A *transition-run* of $\mathcal{P}$ on an ω-word w is an infinite sequence of transitions $r = t_0 t_1 t_2 \ldots = (q_0, w[0], q_1)(q_1, w[1], q_2)(q_2, w[2], q_3) \ldots$ such that $q_0 \in Q_0$ and $q_{i+1} \in \Delta(q_i, w[i])$ for all i. We denote by $\mathit{inf}(r)$ the set of transitions occurring infinitely often in r. An *acceptance condition on transitions* has the same syntax as before and is an expression over the syntax

$$\alpha ::= \mathit{inf}(T) \mid \mathit{fin}(T) \mid \alpha_1 \vee \alpha_2 \mid \alpha_1 \wedge \alpha_2 \quad \text{with } T \subseteq \Delta$$

Acceptance of an ω-word w and the acceptance conditions are defined analogously to the acceptance conditions on states. Defining acceptance on transitions does not change the expressivity of the ω-automaton, but automata with transition-based acceptance can be more succinct compared to automata with state-based acceptance. In the following we just use the term *run* for both types of runs, state-run and transition-run.

Limit-Deterministic Büchi Automata. Intuitively, a nondeterministic Büchi automaton (NBA) is limit-deterministic if it can be split into a nondeterministic component without accepting states (or transitions), and a deterministic component. The automaton can only accept by 'jumping' from the nondeterministic to the deterministic component, but after the jump it must stay in the deterministic component forever. Formally, an NBA $\mathcal{B} = (Q, \Delta, Q_0, \mathit{inf}(S))$ is a *limit-deterministic Büchi automaton* (LDBA) if Q can be partitioned into two disjoint sets $Q = Q_\mathcal{N} \uplus Q_\mathcal{D}$ such that

1. $\Delta(q, a) \subseteq Q_{\mathcal{D}}$ and $|\Delta(q, a)| = 1$ for every $q \in Q_{\mathcal{D}}$, $a \in \Sigma$, and
2. $S \subseteq Q_{\mathcal{D}}$.

The definition for LDBAs with transition acceptance is analogous.

Notation for ω-Automata Classes. We abbreviate the type of an ω-automaton using a simple, well-known scheme. We denote the branching mode by D (deterministic), LD (limit-deterministic), or N (nondeterministic) and follow this by the acceptance condition: B (Büchi), C (co-Büchi), R (Rabin), GB (generalised Büchi), or GR (generalised Rabin). We suffix this string by A which stands for automaton.

Further Acceptance Conditions and Branching Modes. In this thesis we solely focus on the deterministic and the nondeterministic (also called existential) branching modes of ω-automata. Other branching modes such as as alternating branching or universal branching, which is dual to the existential branching mode and where every run needs to be accepting, are not studied here. In the universal branching case a simple dualisation of the results for NBAs is sufficient and yields universal co-Büchi automata. In the alternating branching case, there exists already a very natural relationship between LTL and very-weak alternating co-Büchi automata [MSS88; Var94] and the approach we outline later does not add an interesting perspective.

There are also more acceptance conditions than the ones we listed above, but beyond the focus of this dissertation: parity acceptance, which can be seen as a restricted form of a Rabin acceptance condition; Streett acceptance, which is the dual of the Rabin acceptance condition; or Emerson-Lei acceptance, which is a symbolic encoding of the Muller acceptance condition.

2.1.1 Boolean Operations

In the later chapters we rely on Boolean operations in the chosen automaton model to assemble the final automaton. We survey several standard constructions for binary intersection and union for deterministic and nondeterministic automata with acceptance condition on states. Since these constructions are folklore, the correctness proofs are skipped. Further, the generalisation to the n-ary case and to the case with acceptance defined on transitions is straight-forward and left as an exercise for the reader.

We write $\mathcal{A} \cap \mathcal{B}$ and $\mathcal{A} \cup \mathcal{B}$ to denote the intersection and the union construction, respectively, for the automata $\mathcal{A}$ and $\mathcal{B}$. Since these constructions are not uniform over all involved automata models, we always assume that the best construction or the best combination of constructions from the following is chosen. Further, because we allow blocking states in our automata definition, we assume that in the (synchronous) product construction a trap state is implicitly added, when needed.

Proposition 2.1. *Let $\mathcal{A}_1 = (Q^1, \Delta^1, Q^1_0, \alpha^1)$ and $\mathcal{A}_2 = (Q^2, \Delta^2, Q^2_0, \alpha^2)$ be two ω-automata over the same alphabet Σ. We define the product pre-automaton $\mathcal{P}_\times$ as:*

$$\mathcal{P}_\times \coloneqq (Q^1 \times Q^2, \Delta_\times, Q^1_0 \times Q^2_0)$$

where the transition function $\Delta_\times$ is given by:

$$\Delta_\times(\langle q_1, q_2\rangle, a) \coloneqq \{\langle q'_1, q'_2\rangle : q'_1 \in \Delta^1(q_1, a), q'_2 \in \Delta^2(q_2, a)\}$$

The intersection-automaton $\mathcal{A}_1 \cap \mathcal{A}_2$ over the alphabet Σ recognising $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$ is $\mathcal{P}_\times$ equipped with the acceptance condition $\alpha^1 \wedge \alpha^2$. The union-automaton $\mathcal{A}_1 \cup \mathcal{A}_2$ over the alphabet Σ recognising $\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$ is $\mathcal{P}_\times$ equipped with the acceptance condition $\alpha^1 \vee \alpha^2$. To make this operation well-defined we need to replace all occurrences of $inf(S)$ and $fin(S)$ within α^1 by $inf(S \times Q^2)$ and $fin(S \times Q^2)$. Analogously, all occurrences within α^2 need to be replaced by $inf(Q^1 \times S)$ and $fin(Q^1 \times S)$.

Note that if both automata, $\mathcal{A}_1$ and $\mathcal{A}_2$, are deterministic, then also $\mathcal{A}_1 \cap \mathcal{A}_2$ and $\mathcal{A}_1 \cup \mathcal{A}_2$ are deterministic. Further, observe that for Büchi automata the acceptance condition obtained from this construction is not necessarily again Büchi. For the union-automaton we simply use the following rewrite rule:

$$\bigvee_{i=1}^{k} inf(I_i) \rightsquigarrow inf\left(\bigcup_{i=1}^{k} I_i\right)$$

But for the intersection-automaton we need to apply a degeneralisation construction to obtain a Büchi automaton:

Proposition 2.2. *Let $\mathcal{A} = (Q, \Delta, Q_0, \bigwedge_{i=1}^{k} inf(I_i))$ be a generalised Büchi automaton. Then the following Büchi automaton over the alphabet Σ recognises $\mathcal{L}(\mathcal{A})$:*

$$\mathcal{A}' \coloneqq \left(\bigcup_{i=1}^{k} \{i\} \times Q, \Delta', \{1\} \times Q_0, inf(\{k\} \times I_k)\right)$$

where the transition function Δ' is given by:

$$\Delta'(\langle i, q\rangle, a) \coloneqq \begin{cases} \{\langle (i \bmod k) + 1, p\rangle : p \in \Delta(q, a)\} & \text{if } q \in I_i \\ \{\langle i, p\rangle : p \in \Delta(q, a)\} & \text{otherwise.} \end{cases}$$

Note that, again, if the automaton $\mathcal{A}$ is deterministic, then also $\mathcal{A}'$ is deterministic. Finally, if we want to construct the union of two nondeterministic Büchi automata, then we can use a simpler construction than the product construction from before:

Proposition 2.3. *Let $\mathcal{A}_1 = (Q^1, \Delta^1, Q_0^1, \alpha^1)$ and $\mathcal{A}_2 = (Q^2, \Delta^2, Q_0^2, \alpha^2)$ be two nondeterministic automata over the same alphabet Σ and a disjoint set of states. Then the following automaton over the alphabet Σ recognises $\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$:*

$$\mathcal{A}_1 \cup \mathcal{A}_2 \coloneqq (\{q_0\} \uplus Q^1 \uplus Q^2, \Delta_\cup, \{q_0\}, \alpha^1 \vee \alpha^2)$$

where we define the transition function $\Delta_\cup$ as follows:

$$\begin{aligned} \Delta_\cup \coloneqq & \Delta^1 \cup \{(q_0, a, p^1) : q_0^1 \in Q_0^1, a \in \Sigma, p^1 \in \Delta^1(q_0^1, a)\} \\ & \cup \Delta^2 \cup \{(q_0, a, p^2) : q_0^2 \in Q_0^2, a \in \Sigma, p^2 \in \Delta^2(q_0^2, a)\} \end{aligned}$$

2.1.2 Visual Representation

In the following chapters we will deal with automata over alphabets of the structure $\Sigma = 2^X$ for some finite set X. In this case we simplify the graphical representation of transitions by the following symbolic notation:

$$q \xrightarrow{\emptyset,\{b\},\{c\},\{b,c\},\{a,b,c\}} p \qquad \text{is replaced by} \qquad q \xrightarrow{\overline{a}+bc} p$$

We write a for all sets containing a, and $\overline{a}$ for all sets *not* containing a. Furthermore, we write $\psi\chi$ for the intersection and $\psi + \chi$ for the union of the sets represented by ψ and χ.

2.2 Linear Temporal Logic

Most authors introduce linear temporal logic (LTL) [Pnu77; GPSS80; Pnu81] with the following reduced syntax:

$$\varphi ::= \mathbf{tt} \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\varphi \mid \varphi\mathbf{U}\varphi$$

Left-out, but often used LTL operators are then added as abbreviations. This reduced syntax captures the essence of LTL and has the advantage that only a few cases have to be considered, e.g. in an induction. However, this reduced syntax can also be an obstacle. The result we present requires two characteristics from an LTL syntax: First, it is essential that in addition to **U** (until) we have a 'weak' version of it: **W** (weak until). Second, all formulas need to be in negation-normal-form, i.e. negations only occur in front of atomic propositions. Thus we introduce **ff**, $\neg a$, $\vee$ and the temporal operators **R** (release) and **M** (strong release) in order to remove the arbitrary negations ($\neg\varphi$) from the syntax:

Definition 2.4 (LTL)**.**

$$\varphi ::= \mathbf{tt} \mid \mathbf{ff} \mid a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \varphi\mathbf{U}\varphi \mid \varphi\mathbf{M}\varphi \mid \varphi\mathbf{R}\varphi \mid \varphi\mathbf{W}\varphi \quad \textit{with } a \in Ap$$

Let w be a word over the alphabet 2^{Ap} and let φ be a formula. The satisfaction relation $w \models \varphi$ is inductively defined as follows:

$w \models \mathbf{tt}$		$w \models \mathbf{X}\varphi$	*iff* $w_1 \models \varphi$
$w \not\models \mathbf{ff}$			
$w \models a$	*iff* $a \in w[0]$	$w \models \varphi\mathbf{U}\psi$	*iff* $\exists k.\, w_k \models \psi$ *and* $\forall j < k.\, w_j \models \varphi$
$w \models \neg a$	*iff* $a \notin w[0]$	$w \models \varphi\mathbf{M}\psi$	*iff* $\exists k.\, w_k \models \varphi$ *and* $\forall j \leq k.\, w_j \models \psi$
$w \models \varphi \wedge \psi$	*iff* $w \models \varphi$ *and* $w \models \psi$	$w \models \varphi\mathbf{R}\psi$	*iff* $\forall k.\, w_k \models \psi$ *or* $w \models \varphi\mathbf{M}\psi$
$w \models \varphi \vee \psi$	*iff* $w \models \varphi$ *or* $w \models \psi$	$w \models \varphi\mathbf{W}\psi$	*iff* $\forall k.\, w_k \models \varphi$ *or* $w \models \varphi\mathbf{U}\psi$

We denote by $\mathcal{L}(\varphi) \coloneqq \{w \in (2^{Ap})^\omega : w \models \varphi\}$ the language of φ. Two formulas φ, ψ are equivalent*, denoted $\varphi \sim_l \psi$, if their languages are equal. Formally:*

$$\varphi \sim_l \psi \coloneqq (\mathcal{L}(\varphi) = \mathcal{L}(\psi))$$

The semantics make it clear why **W** is called *weak until*: it behaves exactly as **U**, but does not enforce that ψ is eventually satisfied. Similarly, **M** is called *strong release*, because φ needs to be satisfied eventually. It is easy to see that every LTL formula (of the reduced syntax) can be translated to an equivalent LTL formula in negation-normal-form without an increase in size[1]. Lastly, we use the two common abbreviations $\mathbf{F}\varphi \coloneqq \mathbf{tt}\mathbf{U}\varphi$ (eventually) and $\mathbf{G}\varphi \coloneqq \mathbf{ff}\mathbf{R}\varphi$ (always) with well-known semantics:

$$w \models \mathbf{F}\varphi \;\; \textit{iff} \;\; \exists k.\, w_k \models \varphi \qquad w \models \mathbf{G}\varphi \;\; \textit{iff} \;\; \forall k.\, w_k \models \varphi$$

[1] Assuming we consider a and $\neg a$ being of the same size.

2.2.1 Propositional Semantics

A subformula ψ of φ is called *proper* if it is neither a conjunction nor a disjunction, i.e., if the root of its syntax tree is labelled by either a, $\neg a$, or a temporal operator ($\mathbf{U}$, $\mathbf{R}$, $\mathbf{M}$, $\mathbf{W}$, or $\mathbf{X}$). We denote by $sf(\varphi)$ the set of proper subformulas of φ.

In addition to the interpretation over infinite words we often regard LTL formulas as propositional formulas. Intuitively, this can be described with the following transformation: Let φ be a formula. We replace every maximal proper subformula ψ by a propositional variable x_ψ to obtain a propositional formula φ_p. We write $\mathcal{I} \models_p \varphi$ if the propositional assignment $\mathcal{I}\colon \mathrm{LTL} \to \{\mathbf{tt}, \mathbf{ff}\}$ satisfies φ_p. Note that depending on the context we take the equivalent view of $\mathcal{I}$ being a set of LTL formulas. Formally:

Definition 2.5 (Propositional Semantics of LTL)**.** *Let $\mathcal{I}$ be a set of LTL formulas and let φ be an LTL formula. The satisfaction relation $\mathcal{I} \models_p \varphi$ is inductively defined as follows:*

$$
\begin{array}{lll}
\mathcal{I} \models_p \mathbf{tt} & & \mathcal{I} \models_p \mathbf{X}\varphi \quad \textit{iff} \quad \mathbf{X}\varphi \in \mathcal{I} \\
\mathcal{I} \not\models_p \mathbf{ff} & & \\
\mathcal{I} \models_p a \quad \textit{iff} \quad a \in \mathcal{I} & & \mathcal{I} \models_p \varphi\mathbf{U}\psi \quad \textit{iff} \quad \varphi\mathbf{U}\psi \in \mathcal{I} \\
\mathcal{I} \models_p \neg a \quad \textit{iff} \quad \neg a \in \mathcal{I} & & \mathcal{I} \models_p \varphi\mathbf{M}\psi \quad \textit{iff} \quad \varphi\mathbf{M}\psi \in \mathcal{I} \\
\mathcal{I} \models_p \varphi \wedge \psi \quad \textit{iff} \quad \mathcal{I} \models_p \varphi \textit{ and } \mathcal{I} \models_p \psi & & \mathcal{I} \models_p \varphi\mathbf{R}\psi \quad \textit{iff} \quad \varphi\mathbf{R}\psi \in \mathcal{I} \\
\mathcal{I} \models_p \varphi \vee \psi \quad \textit{iff} \quad \mathcal{I} \models_p \varphi \textit{ or } \mathcal{I} \models_p \psi & & \mathcal{I} \models_p \varphi\mathbf{W}\psi \quad \textit{iff} \quad \varphi\mathbf{W}\psi \in \mathcal{I}
\end{array}
$$

Two formulas φ, ψ are propositionally equivalent*, denoted $\varphi \sim_p \psi$, if*

$$\mathcal{I} \models_p \varphi \iff \mathcal{I} \models_p \psi$$

holds for all propositional assignments $\mathcal{I}$.

Example 2.6. *Let $\varphi = \psi_1 \vee (\psi_2 \wedge \psi_1)$ with $\psi_1 = \mathbf{X}b$ and $\psi_2 = \mathbf{G}(a \vee \mathbf{X}b)$. Let $\mathcal{I} = \{\psi_1\}$ and let $\mathcal{J} = \{\psi_2\}$ be two propositional assignments. We then have $\mathcal{I} \models_p \varphi$, but $\mathcal{J} \not\models_p \varphi$.*

Further, we have $\varphi \sim_p \psi_1$. Thus $\mathbf{X}b$ is propositionally equivalent to φ and $\mathbf{X}b$ is an element of the equivalence class $[\varphi]_{\sim_p}$.

Notice that this definition maps LTL formulas to monotone Boolean functions. Thus $\emptyset \models_p \varphi$ holds if and only if $\varphi \sim_p \mathbf{tt}$ and $\mathcal{U} \not\models_p \varphi$ where $\mathcal{U}$ denotes the universe containing all variables holds if and only if $\varphi \sim_p \mathbf{ff}$. Furthermore, $\models$ and $\models_p$ interpret $\mathbf{tt}$, $\mathbf{ff}$, $\wedge$, and $\vee$ in the same way:

Lemma 2.7. *Let φ be a formula and let w be word. Then:*

$$w \models \varphi \iff \{\psi : \psi \in sf(\varphi) \wedge w \models \psi\} \models_p \varphi$$

Proof sketch. This follows from a straight-forward structural induction on φ. □

Example 2.8. *Let $\varphi = a \vee \mathbf{F}b$ and $w = (\emptyset\{b\})^\omega$. In this case we have $w \models \varphi$ and $\{\mathbf{F}b\} \models_p \varphi$.*

2.2.2 Notable Fragments: μLTL, νLTL, $\mathbf{GF}(\mu LTL)$, and $\mathbf{FG}(\nu LTL)$

In the upcoming chapters the following four fragments of LTL will play a central role, taking the role of in the introduction mentioned 'simple' languages:

– The fragment μLTL and the fragment νLTL.

  μLTL is the fragment of LTL restricted to temporal operators **U** and **M**, on top of Boolean connectives $(\wedge, \vee)$, literals $(a, \neg a)$, and the next operator (**X**). νLTL is defined analogously, but with the operators **R** and **W**. In the literature μLTL is also called syntactic co-safety and νLTL syntactic safety.

– The fragments $\mathbf{GF}(\mu LTL)$ and $\mathbf{FG}(\nu LTL)$.

  These fragments contain the formulas of the form $\mathbf{GF}\varphi$, where $\varphi \in \mu LTL$, and $\mathbf{FG}\varphi$, where $\varphi \in \nu LTL$, respectively.

We substantiate the claim that these languages are 'simple' by giving in Sections 5.1 and 6.2 straight-forward constructions. The reason for the names μLTL and νLTL is that **U** and **M** are least-fixed-point operators, in the sense that their semantics is naturally formulated by least fixed-points, e.g. in the μ-calculus, while the semantics of **R** and **W** is naturally formulated by greatest fixed-points.

3 The 'after'-Function

3.1 Definition and Properties

The 'after function' $af(\varphi, w)$, read 'φ after w', [EK14; EKS16; EKS18] is the foundation for the translations to automata presented in the following chapters. The function assigns to a formula φ and a finite word w another formula such that, intuitively, φ holds for ww' if and only if $af(\varphi, w)$ holds 'after reading w', that is, if and only if $w' \models af(\varphi, w)$.

Definition 3.1. *Let φ be a formula and $\sigma \in 2^{Ap}$ a single letter. The formula $af(\varphi, \sigma)$ is inductively defined as follows:*

$$
\begin{array}{lll}
af(\mathbf{tt}, \sigma) & = \mathbf{tt} & af(\mathbf{X}\varphi, \sigma) = \varphi \\
af(\mathbf{ff}, \sigma) & = \mathbf{ff} & \\
af(a, \sigma) & = \textit{if } a \in \sigma \textit{ then } \mathbf{tt} \textit{ else } \mathbf{ff} & af(\varphi \mathbf{U} \psi, \sigma) = af(\psi, \sigma) \vee (af(\varphi, \sigma) \wedge \varphi \mathbf{U} \psi) \\
af(\neg a, \sigma) & = \textit{if } a \notin \sigma \textit{ then } \mathbf{tt} \textit{ else } \mathbf{ff} & af(\varphi \mathbf{M} \psi, \sigma) = af(\psi, \sigma) \wedge (af(\varphi, \sigma) \vee \varphi \mathbf{M} \psi) \\
af(\varphi \wedge \psi, \sigma) & = af(\varphi, \sigma) \wedge af(\psi, \sigma) & af(\varphi \mathbf{R} \psi, \sigma) = af(\psi, \sigma) \wedge (af(\varphi, \sigma) \vee \varphi \mathbf{R} \psi) \\
af(\varphi \vee \psi, \sigma) & = af(\varphi, \sigma) \vee af(\psi, \sigma) & af(\varphi \mathbf{W} \psi, \sigma) = af(\psi, \sigma) \vee (af(\varphi, \sigma) \wedge \varphi \mathbf{W} \psi)
\end{array}
$$

Furthermore, we generalise af to finite words by defining $af(\varphi, \epsilon) \coloneqq \varphi$ and $af(\varphi, \sigma w) \coloneqq af(af(\varphi, \sigma), w)$ for every $\sigma \in 2^{Ap}$ and every finite word w. Finally, we define the set of formulas reachable from *φ as $Reach(\varphi) \coloneqq \{af(\varphi, w) : w \in (2^{Ap})^*\}$.*

Example 3.2. *Let $\varphi = a \vee (b \mathbf{U} c)$ be a formula. We then have $af(\varphi, \{a\}) = \mathbf{tt} \vee (\mathbf{ff} \vee (\mathbf{ff} \wedge b \mathbf{U} c))$, $af(\varphi, \{b\}) = \mathbf{ff} \vee (\mathbf{ff} \vee (\mathbf{tt} \wedge b \mathbf{U} c))$, and $af(\varphi, \emptyset) = \mathbf{ff} \vee (\mathbf{ff} \vee (\mathbf{ff} \wedge b \mathbf{U} c))$. We discuss in Section 3.2 how to deal with the large amount of added constants and this example is revised in Example 3.5 using such a method.*

Lemma 3.3. *Let φ be a formula, let $w \in (2^{Ap})^*$ be a finite word, and let $w' \in (2^{Ap})^\omega$ be an infinite word. Then:*

1. $ww' \models \varphi \iff w' \models af(\varphi, w)$
2. $sf(af(\varphi, w)) \subseteq sf(\varphi)$

Lemma 3.3 was introduced in [EK14; EKS16] and describes the main purpose of *af*: The first part can be read as $\mathcal{L}(\varphi)^w = \mathcal{L}(af(\varphi, w))$, where L^w denotes the derivate under w of the language L, defined as $L^w \coloneqq \{w' : ww' \in L\}$. Thus *af* computes the derivative of a language, represented by a formula. The second part states that $af(\varphi, w)$ is always a positive Boolean combination of proper subformulas of φ and *af* does not create new proper subformulas. Thus propositional logic offers a way to approximate equivalence of derivatives.

Proof of Lemma 3.3. (1) We show by induction on φ that for a single letter $\sigma \in 2^{Ap}$ the property holds, where we just show two representative cases of the induction. The result

for arbitrary w is then proven by induction on the length of w. Let us now proceed with proving the single-letter case:

$$\sigma w' \models \varphi \iff w' \models \mathit{af}(\varphi, \sigma)$$

– Case $\varphi = a$.

$$\sigma w' \models a \iff a \in \sigma \iff \mathit{af}(a, \sigma) = \mathbf{tt} \iff \sigma w \models \mathit{af}(a, \sigma)$$

– Case $\varphi = \psi_1 \mathbf{U} \psi_2$.

$$\begin{array}{lll} & \sigma w' \models \varphi & \\ \iff & \sigma w' \models \psi_2 \vee (\psi_1 \wedge \mathbf{X}\varphi) & \text{(LTL expansion)} \\ \iff & w' \models \mathit{af}(\psi_2, \sigma) \vee (\mathit{af}(\psi_1, \sigma) \wedge \varphi) & \text{(induction hypothesis)} \\ \iff & w' \models \mathit{af}(\varphi, \sigma) & \end{array}$$

(2) Intuitively this holds, since *af* does not create new elements in the syntax tree except Boolean combinations of existing temporal subformulas. The formal proof proceeds by induction on φ and then on the length of w. □

Related Work. The definition of *af* is almost identical to the transition relation of very-weak alternating automata constructed from LTL formulas [MSS88; Var94]. Thus *af* can be interpreted as a mechanism to track a run on such an automaton. This close relationship stems from the fact that both approaches use 'LTL expansion laws' [BK08] to construct the final result.

As mentioned in Chapter 1, regular-expression derivatives are an elegant, but seemingly forgotten, technique for translating regular expressions to deterministic [Brz64] as well as nondeterministic [Ant96] finite automata. The function *af* follows in these footsteps for LTL and we will identify more conceptual similarities in the rest of this chapter.

Definitions for F and G. In examples we often use formulas containing $\mathbf{F}$ and $\mathbf{G}$. In order to keep the examples readable we add the following two shortened definitions:

$$\mathit{af}(\mathbf{F}\varphi, \sigma) \coloneqq \mathit{af}(\varphi, \sigma) \vee \mathbf{F}\varphi \qquad \mathit{af}(\mathbf{G}\varphi, \sigma) \coloneqq \mathit{af}(\varphi, \sigma) \wedge \mathbf{G}\varphi$$

Notational Conventions. In the dissertation we use the following notational convention for *af*. We denote a change to the object it is applied by a subscript, e.g. $\mathit{af}_\sim$ signals that *af* operates on equivalence classes of $\sim$ instead of formulas. We use a superscript to denote changes to the computation of *af* itself, e.g. $\mathit{af}^{\mathbf{F}\psi}$ hints at the fact that we added a special case, e.g. $\mathit{af}^{\mathbf{F}\psi}(\mathbf{tt}, \sigma) = \mathbf{F}\psi$.

3.2 *af*-Congruences

The automata constructions we define use formulas as states and *af* as the transition relation. However, blindly applying *af* has two problems: it might yield an infinite state space and, more importantly, it is unclear when a state should be considered accepting or rejecting, since a simple test for $\mathbf{tt}$ or $\mathbf{ff}$ might not suffice, e.g. $\mathit{af}(\mathbf{F}a, \{a\}) = \mathbf{tt} \vee \mathbf{F}a \neq \mathbf{tt}$. Let us focus on the second question first: we want to determine in a simple way if $\mathcal{L}(\varphi)$

is either $\emptyset$, $(2^{Ap})^\omega$, or something in-between. For this we define the *eval* function which returns $\mathbf{tt}$ if the language is $(2^{Ap})^\omega$, returns $\mathbf{ff}$ if it is $\emptyset$, and returns $\mathbf{?}$ if it is something in between:

Definition 3.4. *Let φ be a formula and let $\{\mathbf{tt}, \mathbf{ff}, \mathbf{?}\}$ be the domain of a three-valued logic. Then the* constants-value *eval(φ) is inductively defined as follows:*

$$
\begin{array}{lll}
\mathit{eval}(\mathbf{tt}) & = \mathbf{tt} & \mathit{eval}(\mathbf{X}\varphi) = \mathbf{?} \\
\mathit{eval}(\mathbf{ff}) & = \mathbf{ff} & \\
\mathit{eval}(a) & = \mathbf{?} & \mathit{eval}(\varphi\mathbf{U}\psi) = \mathbf{?} \\
\mathit{eval}(\neg a) & = \mathbf{?} & \mathit{eval}(\varphi\mathbf{M}\psi) = \mathbf{?} \\
\mathit{eval}(\varphi \wedge \psi) & = \mathit{eval}(\varphi) \sqcap \mathit{eval}(\psi) & \mathit{eval}(\varphi\mathbf{R}\psi) = \mathbf{?} \\
\mathit{eval}(\varphi \vee \psi) & = \mathit{eval}(\varphi) \sqcup \mathit{eval}(\psi) & \mathit{eval}(\varphi\mathbf{W}\psi) = \mathbf{?}
\end{array}
$$

where we define $\sqcap$ and $\sqcup$ on the three-valued logic as:

$\sqcap$	$\mathbf{ff}$	$\mathbf{?}$	$\mathbf{tt}$
$\mathbf{ff}$	$\mathbf{ff}$	$\mathbf{ff}$	$\mathbf{ff}$
$\mathbf{?}$	$\mathbf{ff}$	$\mathbf{?}$	$\mathbf{?}$
$\mathbf{tt}$	$\mathbf{ff}$	$\mathbf{?}$	$\mathbf{tt}$

$\sqcup$	$\mathbf{ff}$	$\mathbf{?}$	$\mathbf{tt}$
$\mathbf{ff}$	$\mathbf{ff}$	$\mathbf{?}$	$\mathbf{tt}$
$\mathbf{?}$	$\mathbf{?}$	$\mathbf{?}$	$\mathbf{tt}$
$\mathbf{tt}$	$\mathbf{tt}$	$\mathbf{tt}$	$\mathbf{tt}$

Two formulas φ, ψ are constants-equivalent*, denoted $\varphi \sim_c \psi$, if $\varphi = \psi$ or $\mathit{eval}(\varphi) = \mathit{eval}(\psi) \in \{\mathbf{tt}, \mathbf{ff}\}$.*

Example 3.5. *Let $\varphi = a \vee (b\mathbf{U}c)$ be a formula. We then have $\mathit{af}(\varphi, \{a\}) \sim_c \mathbf{tt}$, $\mathit{af}(\varphi, \{b\}) \nsim_c b\mathbf{U}c$, $\mathit{af}(\varphi, \{b\}\{c\}) \sim_c \mathbf{tt}$, and $\mathit{af}(\varphi, \emptyset) \sim_c \mathbf{ff}$.*

Note the conceptual similarity to Brzozowski derivatives [Brz64]: In that construction states are marked as final if the regular expression r labelling the state recognises the empty word ϵ. For this [Brz64] introduces a syntactic check that works in the same way as *eval* does.

Now let us reconsider the first problem: We need a way to decide if two computed formulas $\mathit{af}(\varphi, w)$ and $\mathit{af}(\varphi, w')$ are equivalent and can be collapsed to the same state. For this we coin the term *af-congruence*: An equivalence relation $\sim$ is an *af*-congruence if *af* (for a fixed σ) can be applied to any element of an equivalence class *af* and the resulting formulas are again all in the same equivalence class. This ensures the existence of a canonical lifting of *af* to $\sim$. Further, since we want that every *af*-congruence embeds *eval* we require that $\sim_c$ is a subset of $\sim$. Finally, for obvious reasons we only consider equivalence relations that are under-approximations of language equivalence ($\sim_l$).

Definition 3.6. *Let $\sim$ be an equivalence relation on formulas. We call the equivalence relation $\sim$ an af-*congruence *if the following holds:*

1. $\sim_c \preccurlyeq \sim \preccurlyeq \sim_l$
2. $\varphi \sim \psi \implies \mathit{af}(\varphi, \sigma) \sim \mathit{af}(\psi, \sigma)$ *for all formulas φ, ψ and letters σ.*

Archetypical *af*-Congruences. All equivalence relations ($\sim_c$, $\sim_p$, and $\sim_l$) we have seen so far are in fact *af*-congruences. The relation $\sim_c$ marks one end of the spectrum. It embodies a minimal check to determine if a state should be accepting or rejecting. However, while being cheap to compute, it might generate an infinite state space. The

relation $\sim_l$ marks the other end of the spectrum. It collapses as many formulas as possible to one equivalence class, but deciding $\sim_l$ is expensive and depending on the decision procedure we might need to implicitly build an automaton. Thus we obtain a circular dependency: to build the automaton we need to build an automaton. The equivalence relation $\sim_p$ balances this and provides an efficiently equivalence relation, allows for a finite automaton construction, and embodies *eval*.

Lemma 3.7.

$$\sim_c \prec \sim_p \prec \sim_l$$

Proof. We need to prove (1) $\sim_c \preccurlyeq \sim_p$, (2) $\sim_p \preccurlyeq \sim_l$, and the matching inequalities (3).

(1) It is sufficient to show $[\mathbf{tt}]_{\sim_c} = [\mathbf{tt}]_{\sim_p}$ and $[\mathbf{ff}]_{\sim_c} = [\mathbf{ff}]_{\sim_p}$, since formulas not covered by these two equivalence classes form singleton equivalence classes $[\psi]_{\sim_c} = \{\psi\}$ for $\sim_c$. We observed before that $\varphi \sim_p \mathbf{tt}$ is equivalent to $\emptyset \models_p \varphi$. We continue with $\emptyset \models_p \varphi$ if and only if $\mathit{eval}(\varphi) = \mathbf{tt}$ which can be shown by a straight-forward induction on φ. Taking these two steps together we have $[\mathbf{tt}]_{\sim_c} = [\mathbf{tt}]_{\sim_p}$. The second part ($[\mathbf{ff}]_{\sim_c} = [\mathbf{ff}]_{\sim_p}$) is then proven in the same way.

(2) Let φ, ψ be two formulas and assume $\varphi \sim_p \psi$. We need to show $\varphi \sim_l \psi$. For this let w be a word such that $w \models \varphi$. With Lemma 2.7 we follow $\{\chi \in \mathit{sf}(\varphi) : w \models \chi\} \models_p \varphi$. Due to the monotonicity of $\models_p$ we have $\{\chi \in \mathit{sf}(\varphi) \cup \mathit{sf}(\psi) : w \models \chi\} \models_p \psi$ and we can restrict it again to the proper subformulas of ψ: $\{\chi \in \mathit{sf}(\psi) : w \models \chi\} \models_p \psi$. We apply Lemma 2.7 again and are done with this direction. The other direction is proven analogously.

(3) Finally, we have $\sim_c \neq \sim_p$ and $\sim_p \neq \sim_l$ due to $a \wedge b \not\sim_c b \wedge a$, $a \wedge b \sim_p b \wedge a$, $(\mathbf{F}a) \vee (\mathbf{F}\neg a) \not\sim_p \mathbf{tt}$, and $(\mathbf{F}a) \vee (\mathbf{F}\neg a) \sim_l \mathbf{tt}$. □

Lemma 3.8. *$\sim_c$, $\sim_p$, and $\sim_l$ are af-congruences. The set $\mathit{Reach}(\varphi)_{/\sim_c}$ can be infinite. The sets $\mathit{Reach}(\varphi)_{/\sim_p}$ and $\mathit{Reach}(\varphi)_{/\sim_l}$ are finite and if the formula φ has n proper subformulas, then $\mathit{Reach}(\varphi)_{/\sim_p}$ and $\mathit{Reach}(\varphi)_{/\sim_l}$ have at most cardinality $M(n) \leq 2^{2^n}$, where $M(n)$ denotes the number of monotonic Boolean functions of n variables (Dedekind number).*

Proof. We first observe that $\sim_c$, $\sim_p$, and $\sim_l$ all satisfy the first part of Definition 3.6 due to Lemma 3.7. For the second part we begin with $\sim_l$. Let φ, ψ be two formulas such that $\varphi \sim_l \psi$, let σ be a letter, and let w be a word. Then:

$$\begin{aligned} w \models \mathit{af}(\varphi, \sigma) &\iff \sigma w \models \varphi && \text{(Lemma 3.3)} \\ &\iff \sigma w \models \psi && (\varphi \sim_l \psi) \\ &\iff w \models \mathit{af}(\psi, \sigma) && \text{(Lemma 3.3)} \end{aligned}$$

Hence $\mathit{af}(\varphi, \sigma) \sim_l \mathit{af}(\psi, \sigma)$ and thus $\sim_l$ is an *af*-congruence. We continue with $\sim_p$:

Let φ, ψ be two formulas such that $\varphi \sim_p \psi$ and let σ be a letter. Note that *af* is a substitution that only replaces literals (a, $\neg a$) and modal operators ($\mathbf{M}$, $\mathbf{U}$, $\mathbf{R}$, $\mathbf{W}$, $\mathbf{X}$). These substitutions are a congruence on $\sim_p$ and thus we have in this particular case: $\mathit{af}(\varphi, \sigma) \sim_p \mathit{af}(\psi, \sigma)$. Hence $\sim_p$ is an *af*-congruence.

For the equivalence relation $\sim_c$ we showed in the proof for Lemma 3.7 that $[\mathbf{tt}]_{\sim_c} = [\mathbf{tt}]_{\sim_p}$ and $[\mathbf{ff}]_{\sim_c} = [\mathbf{ff}]_{\sim_p}$ hold. Either an equivalence class $[\varphi]_{\sim_c}$ is a singleton, in which case the second condition trivially holds, or it is one of $[\mathbf{tt}]_{\sim_c}$ and $[\mathbf{ff}]_{\sim_c}$, in which case we simply use the fact $\sim_p$ is an *af*-congruence to show that the second condition holds. Thus $\sim_c$ is an *af*-congruence.

We show that $\mathit{Reach}(\mathbf{F}a)_{/\sim_c}$ is infinite. Note that the set contains for each $\emptyset^n$ a distinct equivalence class ($\mathit{Reach}(\varphi)_{/\sim_c} \supseteq \{[\mathbf{F}a]_{\sim_c}, [\mathbf{ff} \vee \mathbf{F}a]_{\sim_c}, [\mathbf{ff} \vee (\mathbf{ff} \vee \mathbf{F}a)]_{\sim_c}, \dots\}$) and thus is infinite.

Let φ be a formula with n proper subformulas. *af* does not create new temporal operators and maps only to Boolean combinations of existing proper subformulas (Lemma 3.3). Thus each $[\psi]_{\sim_p} \in \mathit{Reach}(\varphi)_{/\sim_p}$ can be interpreted as a monotonic Boolean function over n variables. There are at most $M(n)$ many of these functions and thus $|\mathit{Reach}(\varphi)_{/\sim_p}| \leq M(n)$. Furthermore, $M(n)$ can be bounded by 2^{2^n}, since there exist at most that many Boolean functions over n variables.

The last missing piece – $\mathit{Reach}(\varphi)_{/\sim_l}$ has at most $M(n)$ elements – is an immediate consequence of the bounds for $\mathit{Reach}(\varphi)_{/\sim_p}$ and Lemma 3.7. Thus we conclude that $\mathit{Reach}(\varphi)_{/\sim_l}$ is also finite and has at most $M(n)$ many elements. □

3.3 Logical Characterisations of μLTL, νLTL, $\mathbf{GF}(\mu LTL)$, and $\mathbf{FG}(\nu LTL)$

We now detail how an *af*-congruence (Definition 3.6) can be used as mechanism to check if a word w satisfies formulas from 'simple' LTL fragments. This generalises the corresponding result of [EKS16].

Lemma 3.9. *Let $\sim$ be an af-congruence and let w be a word.*

- *Let $\varphi \in \mu LTL$.*
  1. $w \models \varphi \iff \exists i.\ \mathit{af}(\varphi, w_{0i}) \sim \mathbf{tt}$
  2. $w \models \mathbf{GF}\varphi \iff \forall i.\ \exists j \geq i.\ \exists k.\ \mathit{af}(\mathbf{F}\varphi, w_{jk}) \sim \mathbf{tt}$
- *Let $\varphi \in \nu LTL$.*
  3. $w \models \varphi \iff \forall i.\ \mathit{af}(\varphi, w_{0i}) \nsim \mathbf{ff}$
  4. $w \models \mathbf{FG}\varphi \iff \exists i.\ \forall j \geq i.\ \forall k.\ \mathit{af}(\mathbf{G}\varphi, w_{jk}) \nsim \mathbf{ff}$

Proof. Let $\sim$ be an *af*-congruence, let w be a word, and let $\varphi \in \mu LTL$.

($\Rightarrow_1$) Assume $w \models \varphi$. We proceed by structural induction on φ to prove the existence of an i such that $\mathit{af}(\varphi, w_{0i}) \sim_c \mathbf{tt}$, which implies $\mathit{af}(\varphi, w_{0i}) \sim \mathbf{tt}$ by $\sim_c \preccurlyeq \sim$. We only consider two representative cases of the possible cases: $\mathbf{tt}, \mathbf{ff}, a, \neg a, \psi_1 \wedge \psi_2, \psi_1 \vee \psi_2, \mathbf{X}\psi, \psi_1 \mathbf{U} \psi_2, \psi_1 \mathbf{M} \psi_2$.

- Case $\varphi = a$. Since $w \models \varphi$ we have $a \in w[0] = w_{01}$ and we get $\mathit{af}(\varphi, w_{01}) = \mathbf{tt} \sim_c \mathbf{tt}$.
- Case $\varphi = \psi_1 \mathbf{U} \psi_2$. By the semantics of LTL there is a k such that $w_k \models \psi_2$ and $w_\ell \models \psi_1$ for every $0 \leq \ell < k$. By induction hypothesis there exists for every $0 \leq \ell < k$ an $i \geq \ell$ such that $\mathit{af}(\psi_1, w_{\ell i}) \sim_c \mathbf{tt}$ and there exists an $i \geq k$ such that $\mathit{af}(\psi_2, w_{ki}) \sim_c \mathbf{tt}$. Let j be the maximum of all those i's. We prove $\mathit{af}(\psi_1 \mathbf{U} \psi_2, w_{0j}) \sim_c \mathbf{tt}$ via induction on k.
  - $k = 0$.

$$
\begin{aligned}
& \mathit{af}(\psi_1 \mathbf{U} \psi_2, w_{0j}) \\
= \ & \mathit{af}(\psi_2, w_{0j}) \vee (\mathit{af}(\psi_1, w_{0j}) \wedge \mathit{af}(\psi_1 \mathbf{U} \psi_2, w_{1j})) && \text{(Definition 3.1)} \\
\sim_c \ & \mathbf{tt} \vee (\mathit{af}(\psi_1, w_{0j}) \wedge \mathit{af}(\psi_1 \mathbf{U} \psi_2, w_{1j})) && (k = 0 \to \mathit{af}(\psi_2, w_{0j}) \sim_c \mathbf{tt}) \\
\sim_c \ & \mathbf{tt}
\end{aligned}
$$

– $k > 0$.

$$\begin{array}{lll} & af(\psi_1 \mathbf{U} \psi_2, w_{0j}) & \\ = & af(\psi_2, w_{0j}) \vee (af(\psi_1, w_{0j}) \wedge af(\psi_1 \mathbf{U} \psi_2, w_{1j})) & \text{(Definition 3.1)} \\ \sim_c & af(\psi_2, w_{0j}) \vee (\mathbf{tt} \wedge af(\psi_1 \mathbf{U} \psi_2, w_{1j})) & (k > 0 \rightarrow af(\psi_1, w_{0j}) \sim_c \mathbf{tt}) \\ \sim_c & af(\psi_2, w_{0j}) \vee (\mathbf{tt} \wedge \mathbf{tt}) & \text{(induction hypothesis)} \\ \sim_c & \mathbf{tt} & \end{array}$$

($\neg \Rightarrow_1 \neg$) Assume $w \not\models \varphi$. By Lemma 3.3 we have $w_i \not\models af(\varphi, w_{0i})$ for all i, thus $af(\varphi, w_{0i}) \nsim_l \mathbf{tt}$ for all i, and thus $af(\varphi, w_{0i}) \nsim \mathbf{tt}$ which we needed to prove.

(2) We derive this part by unfolding the LTL semantics of $\mathbf{GF}\varphi$ and applying (1):

$$w \models \mathbf{GF}\varphi \iff \forall i.\, \exists j \geq i.\, w_j \models \mathbf{F}\varphi \iff \forall i.\, \exists j \geq i.\, \exists k.\, af(\mathbf{F}\varphi, w_{jk}) \sim \mathbf{tt}$$

($\Rightarrow_3$) Assume $w \models \varphi$. By Lemma 3.3 we have $w_i \models af(\varphi, w_{0i})$ for all i, thus $af(\varphi, w_{0i}) \nsim_l \mathbf{ff}$ holds for any i, and $af(\varphi, w_{0i}) \nsim \mathbf{ff}$ which we needed to prove.

($\neg \Rightarrow_3 \neg$) Assume $w \not\models \varphi$. We proceed by structural induction on φ to prove the existence of an i such that $af(\varphi, w_{0i}) \sim_c \mathbf{ff}$. We only consider two representative cases of the possible cases: $\mathbf{tt}, \mathbf{ff}, a, \neg a, \psi_1 \wedge \psi_2, \psi_1 \vee \psi_2, \mathbf{X}\psi, \psi_1 \mathbf{W} \psi_2, \psi_1 \mathbf{R} \psi_2$.

– Case $\varphi = a$. Since $w \not\models \varphi$ we have $a \notin w[0] = w_{01}$ and we get $af(\varphi, w_{01}) = \mathbf{ff} \sim_c \mathbf{ff}$.

– Case $\varphi = \psi_1 \mathbf{W} \psi_2$. By the semantics of LTL there is a k such that $w_k \not\models \psi_1$ and $w_\ell \not\models \psi_2$ for every $0 \leq \ell \leq k$. By induction hypothesis there exists for every $0 \leq \ell \leq k$ an $i \geq \ell$ such that $af(\psi_2, w_{\ell i}) \sim_c \mathbf{ff}$ and there exists an $i \geq k$ such that $af(\psi_1, w_{ki}) \sim_c \mathbf{ff}$. Let j be the maximum of all those i's. We prove $af(\psi_1 \mathbf{W} \psi_2, w_{0j}) \sim_c \mathbf{ff}$ via induction on k.

– $k = 0$.

$$\begin{array}{lll} & af(\psi_1 \mathbf{W} \psi_2, w_{0j}) & \\ = & af(\psi_2, w_{0j}) \vee (af(\psi_1, w_{0j}) \wedge af(\psi_1 \mathbf{W} \psi_2, w_{1j})) & \text{(Definition 3.1)} \\ \sim_c & \mathbf{ff} \vee (af(\psi_1, w_{0j}) \wedge af(\psi_1 \mathbf{W} \psi_2, w_{1j})) & (k = 0 \rightarrow af(\psi_2, w_{kj}) \sim_c \mathbf{ff}) \\ \sim_c & \mathbf{ff} \vee (\mathbf{ff} \wedge af(\psi_1 \mathbf{W} \psi_2, w_{1j})) & (k = 0 \rightarrow af(\psi_1, w_{kj}) \sim_c \mathbf{ff}) \\ \sim_c & \mathbf{ff} & \end{array}$$

– $k > 0$.

$$\begin{array}{lll} & af(\psi_1 \mathbf{W} \psi_2, w_{0j}) & \\ = & af(\psi_2, w_{0j}) \vee (af(\psi_1, w_{0j}) \wedge af(\psi_1 \mathbf{W} \psi_2, w_{1j})) & \text{(Definition 3.1)} \\ \sim_c & af(\psi_2, w_{0j}) \vee (af(\psi_1, w_{0j}) \wedge af(\psi_1 \mathbf{W} \psi_2, w_{1j})) & (k > 0 \rightarrow af(\psi_2, w_{0j}) \sim_c \mathbf{ff}) \\ \sim_c & \mathbf{ff} \vee (af(\psi_1, w_{0,j}) \wedge \mathbf{ff}) & \text{(induction hypothesis)} \\ \sim_c & \mathbf{ff} & \end{array}$$

(4) We derive this part by unfolding the LTL semantics of $\mathbf{FG}\varphi$ and applying (3):

$$w \models \mathbf{FG}\varphi \iff \exists i.\, \forall j \geq i.\, w_j \models \mathbf{G}\varphi \iff \exists i.\, \forall j \geq i.\, \forall k.\, af(\mathbf{G}\varphi, w_{jk}) \nsim \mathbf{ff}$$

□

4 The Master Theorem

In this chapter we present and prove the Master Theorem: A characterisation of the words satisfying a given formula from which we can easily extract deterministic, limit-deterministic, and nondeterministic automata of asymptotically optimal size.

The essential insight of the Master Theorem is that given a formula φ we can partition the universe into finitely many sets $L_1 \uplus L_2 \uplus \cdots \uplus L_n = (2^{Ap})^\omega$, where two words w and w' are in the same partition L_i if

1. each subformula of φ with the shape $\psi_1 \mathbf{U} \psi_2$ or $\psi_1 \mathbf{M} \psi_2$ is either satisfied infinitely often by w and w', or is violated almost always by w and w', and

2. each subformula of φ with the shape $\psi_1 \mathbf{W} \psi_2$ or $\psi_1 \mathbf{R} \psi_2$ is either satisfied almost always by w and w', or is violated infinitely often by w and w'.

Intuitively, this means two words $w, w' \in L_i$ behave identically with regards to the subformulas of φ when we consider the 'limit' of w and w'. How can we make use of this way to partition the universe? Let us provide some intuition with the help of an example.

Consider the formula $\varphi = \mathbf{G}((a \mathbf{R} b) \vee (c \mathbf{U} d))$. Observe that φ does not belong to any of the fragments mentioned in Section 2.2.2, namely μLTL, νLTL, $\mathbf{GF}(\mu LTL)$, $\mathbf{FG}(\nu LTL)$, for which one can easily obtain automata.

Let L_i be a partition containing words that satisfy infinitely often $c \mathbf{U} d$. Assume we now are promised that the word w is in L_i, meaning that we are promised that along the word w the subformula $c \mathbf{U} d$ holds infinitely often. For example, this is the case for $w = (\{a, b\}\{d\})^\omega$. In particular, we then know that d holds infinitely often, and so we can 'reduce' checking $w \models \varphi$ to checking $w \models \mathbf{G}((a \mathbf{R} b) \vee (c \mathbf{W} d))$, which belongs to the fragment νLTL.

Let L_j be a partition containing words that violate $c \mathbf{U} d$ almost always. Assume we now are promised that the word w is in L_j, meaning that we are promised that $c \mathbf{U} d$ only holds finitely often. For example, because $w = \{d\}^4\{b, c\}^\omega$. Furthermore, we get an additional promise that along the suffix w_5 the formula $c \mathbf{U} d$ never holds again. How can we use this promise? First, $w \models \varphi$ reduces to $w_5 \models \mathit{af}(\varphi, w_{05})$ by the fundamental property of af, Lemma 3.3. Further, a little computation shows that $\mathit{af}(\varphi, w_{05}) \sim_l \varphi$, and so $w \models \varphi$ reduces to $w_5 \models \varphi$. Finally, using that $c \mathbf{U} d$ never holds again, we reduce $w \models \varphi$ to $w_5 \models \mathbf{G}(a \mathbf{R} b \vee \mathbf{ff}) \sim_l \mathbf{G}(a \mathbf{R} b)$ which belongs to the fragment νLTL.

This example suggests a general strategy for translating φ to automata:

– Guess the partition w belongs to. To be more precise, guess the set of least-fixed-point subformulas ($\psi_1 \mathbf{M} \psi_2$, $\psi_1 \mathbf{U} \psi_2$) of φ that hold infinitely often, denoted by $\mathcal{GF}_w^\varphi$, and the set of greatest-fixed-point subformulas ($\psi_1 \mathbf{R} \psi_2$, $\psi_1 \mathbf{W} \psi_2$) that hold almost always, denoted by $\mathcal{FG}_w^\varphi$.

- Guess a *stabilisation point* after which the least-fixed-point subformulas outside $\mathcal{GF}_w^\varphi$ do not hold any more, and the greatest-fixed-point subformulas of $\mathcal{FG}_w^\varphi$ hold forever.
- Use these guesses to reduce $w \models \varphi$ to problems $w \models \psi$ for formulas ψ that belong to the fragments introduced in Section 2.2.2 and for which we know how to build automata as outlined in Sections 5.1 and 6.2 relying on *af* from Chapter 3.
- Check that the guesses are correct.

Since the number of such partitions is finite, we can replace guessing a partition by enumerating all possible ones and 'trying them all out' in parallel. In the rest of the section we develop the logical component of this strategy. In Section 4.1 we introduce the terminology needed to formalise the *stabilisation point*. Section 4.2 shows how to use a guess X for $\mathcal{GF}_w^\varphi$ or a guess Y for $\mathcal{FG}_w^\varphi$ to reduce $w \models \varphi$ to a simpler problem $w \models \varphi[X]_\nu$ or $w \models \varphi[Y]_\mu$, where $\varphi[X]_\nu$ and $\varphi[Y]_\mu$ are read as 'φ with **GF**-advice X' and 'φ with **FG**-advice Y', respectively. Section 4.3 shows how to use the advice to decide $w \models \varphi$. Section 4.4 shows how to check that the advice is correct. The Master Theorem is stated and proved in Section 4.5. We provide a useful restriction of the Master Theorem in Section 4.6.1 and relate to preceding results in Section 4.6.2.

4.1 μ- and ν-Stability

Fix a formula φ. The set of subformulas of φ of the form $\psi_1 \mathbf{U} \psi_2$ and $\psi_1 \mathbf{M} \psi_2$ is denoted by $\mu(\varphi)$. So, loosely speaking, $\mu(\varphi)$ contains the set of subformulas of φ with a least-fixed-point operator at the top of their syntax tree. Given a word w, we are interested in which of these formulas hold infinitely often, and which ones hold at least once, i.e., we are interested in the previously mentioned set $\mathcal{GF}_w^\varphi$ and the now to be defined set $\mathcal{F}_w^\varphi$:

Definition 4.1 (μ-stability)**.** *Let φ be a formula and let w be a word. The set $\mu(\varphi)$ is inductively defined as follows:*

$$
\begin{array}{lll}
\mu(\mathbf{tt}) = \emptyset & & \mu(\mathbf{X}\varphi) = \mu(\varphi) \\
\mu(\mathbf{ff}) = \emptyset & & \\
\mu(a) = \emptyset & & \mu(\varphi \mathbf{U} \psi) = \{\varphi \mathbf{U} \psi\} \cup \mu(\varphi) \cup \mu(\psi) \\
\mu(\neg a) = \emptyset & & \mu(\varphi \mathbf{M} \psi) = \{\varphi \mathbf{M} \psi\} \cup \mu(\varphi) \cup \mu(\psi) \\
\mu(\varphi \wedge \psi) = \mu(\varphi) \cup \mu(\psi) & & \mu(\varphi \mathbf{R} \psi) = \mu(\varphi) \cup \mu(\psi) \\
\mu(\varphi \vee \psi) = \mu(\varphi) \cup \mu(\psi) & & \mu(\varphi \mathbf{W} \psi) = \mu(\varphi) \cup \mu(\psi)
\end{array}
$$

Then the sets $\mathcal{GF}_w^\varphi$ and $\mathcal{F}_w^\varphi$ are defined as follows:

$$
\begin{aligned}
\mathcal{GF}_w^\varphi &= \{\psi : \psi \in \mu(\varphi) \wedge w \models \mathbf{GF}\psi\} \\
\mathcal{F}_w^\varphi &= \{\psi : \psi \in \mu(\varphi) \wedge w \models \mathbf{F}\psi\}
\end{aligned}
$$

We say that w is μ-stable with respect to φ if $\mathcal{GF}_w^\varphi = \mathcal{F}_w^\varphi$.

Example 4.2. *For $\varphi = \mathbf{G}a \vee b\mathbf{U}c$ we have $\mu(\varphi) = \{b\mathbf{U}c\}$. Let $w = \{a\}^\omega$ and $w' = \{b\}\{c\}\{a\}^\omega$. We have $\mathcal{F}_w^\varphi = \emptyset = \mathcal{GF}_w^\varphi$ and $\mathcal{GF}_{w'}^\varphi = \emptyset \subset \{b\mathbf{U}c\} = \mathcal{F}_{w'}^\varphi$. So w is μ-stable with respect to φ, but w' is not.*

Dually, the set of subformulas of φ of the form $\psi_1 \mathbf{R} \psi_2$ and $\psi_1 \mathbf{W} \psi_2$ is denoted by $\nu(\varphi)$. This time we are interested in whether these formulas hold everywhere or almost everywhere, i.e., we are interested in the now to be defined set $\mathcal{G}_w^\varphi$ and the previously mentioned set $\mathcal{FG}_w^\varphi$:

Definition 4.3 (ν-stability)**.** *Let φ be a formula and let w be a word. The set $\nu(\varphi)$ is inductively defined as follows:*

$$
\begin{array}{llll}
\nu(\mathbf{tt}) & = \emptyset & \nu(\mathbf{X}\varphi) & = \nu(\varphi) \\
\nu(\mathbf{ff}) & = \emptyset & & \\
\nu(a) & = \emptyset & \nu(\varphi \mathbf{U} \psi) & = \nu(\varphi) \cup \nu(\psi) \\
\nu(\neg a) & = \emptyset & \nu(\varphi \mathbf{M} \psi) & = \nu(\varphi) \cup \nu(\psi) \\
\nu(\varphi \wedge \psi) & = \nu(\varphi) \cup \nu(\psi) & \nu(\varphi \mathbf{R} \psi) & = \{\varphi \mathbf{R} \psi\} \cup \nu(\varphi) \cup \nu(\psi) \\
\nu(\varphi \vee \psi) & = \nu(\varphi) \cup \nu(\psi) & \nu(\varphi \mathbf{W} \psi) & = \{\varphi \mathbf{W} \psi\} \cup \nu(\varphi) \cup \nu(\psi)
\end{array}
$$

Then the sets $\mathcal{FG}_w^\varphi$ and $\mathcal{G}_w^\varphi$ are defined as follows:

$$
\begin{aligned}
\mathcal{FG}_w^\varphi &= \{\psi : \psi \in \nu(\varphi) \wedge w \models \mathbf{FG}\psi\} \\
\mathcal{G}_w^\varphi &= \{\psi : \psi \in \nu(\varphi) \wedge w \models \mathbf{G}\psi\}
\end{aligned}
$$

We say that w is ν-stable with respect to φ if $\mathcal{FG}_w^\varphi = \mathcal{G}_w^\varphi$.

Example 4.4. *Let φ, w and w' as in Example 4.2. We have $\nu(\varphi) = \{\mathbf{G}a\}$. The word w is ν-stable, but w' is not, because $\mathcal{FG}_{w'}^\varphi = \{\mathbf{G}a\} \supset \emptyset = \mathcal{G}_{w'}^\varphi$.*

So not every word is μ-stable or ν-stable. However, notice that the inclusions $\mathcal{GF}_w^\varphi \subseteq \mathcal{F}_w^\varphi$ and $\mathcal{FG}_w^\varphi \supseteq \mathcal{G}_w^\varphi$ hold for all formulas φ and words w. Moreover, as shown by the following lemma, all but finitely many suffixes of a word are μ- and ν-stable. If the word w_i is μ-stable and ν-stable with respect to φ, then we call i a *stabilisation point* for w with respect to φ.

Lemma 4.5. *Let φ be a formula and let w be a word. Then there exist indices $i, j \geq 0$ such that for every $k \geq 0$ the suffix w_{i+k} is μ-stable and the suffix w_{j+k} is ν-stable with respect to φ.*

Proof. We only prove the μ-stability part; the proof of the other part is similar. Let φ be a formula and let w be a word. For the following proof we will omit the superscript φ, as it is always φ.

Since $\mathcal{GF}_{w_i} \subseteq \mathcal{F}_{w_i}$ for every $i \geq 0$, it suffices to exhibit an index i such that $\mathcal{GF}_{w_{i+k}} \supseteq \mathcal{F}_{w_{i+k}}$ for every $k \geq 0$. If $\mathcal{GF}_w \supseteq \mathcal{F}_w$ then we can choose $i := 0$. So assume $\mathcal{F}_w \setminus \mathcal{GF}_w \neq \emptyset$. By definition, every $\psi \in \mathcal{F}_w \setminus \mathcal{GF}_w$ holds only finitely often along w. So for every $\psi \in \mathcal{F}_w \setminus \mathcal{GF}_w$ there exists an index i_ψ such that $w_{i_\psi + k} \not\models \psi$ for every $k \geq 0$. Let $i := \max\{i_\psi : \psi \in \mathcal{F}_w\}$, which exists because $\mathcal{F}_w$ is a finite set. It follows $\mathcal{GF}_{w_{i+k}} \supseteq \mathcal{F}_{w_{i+k}}$ for every $k \geq 0$, and so every w_{i+k} is μ-stable. □

Example 4.6. *Let again $\varphi = \mathbf{G}a \vee b\mathbf{U}c$. The word $w' = \{b\}\{c\}\{a\}^\omega$ is neither μ-stable nor ν-stable with respect to φ, but all suffixes $w'_{(2+k)}$ of w' are both μ-stable and ν-stable with respect to φ.*

4.2 The Formulas $\varphi[X]_\nu$ and $\varphi[Y]_\mu$

Assume we have to determine if a word w satisfies φ, and we are told that w is μ-stable. Further, we are given the set $X \subseteq \mu(\varphi)$ such that $\mathcal{GF}_w^\varphi = X = \mathcal{F}_w^\varphi$. We use this oracle information to reduce the problem $w \models \varphi$ to a 'simpler' problem $w \models \varphi[X]_\nu$, where 'simpler' means that $\varphi[X]_\nu$ is a formula of νLTL, for which we have simple automata constructions (Sections 5.1 and 6.2) based on Lemma 3.9. In other words, we define a formula $\varphi[X]_\nu \in \nu LTL$ such that:

$$(\mathcal{GF}_w^\varphi = X = \mathcal{F}_w^\varphi) \implies (w \models \varphi \longleftrightarrow w \models \varphi[X]_\nu)$$

Observe that $X \subseteq \mu(\varphi)$ but $\varphi[X]_\nu \in \nu LTL$, and so the latter, not the former, is the reason for the ν-subscript in the notation $\varphi[X]_\nu$.

The definition of $\varphi[X]_\nu$ is purely syntactic, and the intuition behind it is very simple. All the main ideas are illustrated by the following examples, where we assume $\mathcal{GF}_w^\varphi = X = \mathcal{F}_w^\varphi$:

- $\varphi = \mathbf{F}a \wedge \mathbf{G}b$ and $X = \{\mathbf{F}a\}$. Since $X = \mathcal{GF}_w^\varphi$, we have then $\mathbf{F}a \in \mathcal{GF}_w^\varphi$, which implies in particular $w \models \mathbf{F}a$. So we can reduce $w \models \mathbf{F}a \wedge \mathbf{G}b$ to $w \models \mathbf{G}b$, and so $\varphi[X]_\nu \coloneqq \mathbf{tt} \wedge \mathbf{G}b$.
- $\varphi = \mathbf{F}a \wedge \mathbf{G}b$ and $X = \emptyset$. Since $X = \mathcal{F}_w^\varphi$, we have then $\mathbf{F}a \notin \mathcal{F}_w^\varphi$, and so $w \not\models \mathbf{F}a$. So we can reduce $w \models \mathbf{F}a \wedge \mathbf{G}b$ to the trivial problem $w \models \mathbf{ff}$, and so $\varphi[X]_\nu \coloneqq \mathbf{ff} \wedge \mathbf{G}b$.
- $\varphi = \mathbf{G}(b\mathbf{U}c)$ and $X = \{b\mathbf{U}c\}$. Since $X = \mathcal{GF}_w^\varphi$, we have then $b\mathbf{U}c \in \mathcal{GF}_w^\varphi$, and so $w \models \mathbf{GF}(b\mathbf{U}c)$. This does not imply $w_i \models b\mathbf{U}c$ for all suffixes w_i of w, but it implies that c will hold infinitely often in the future. So we can reduce $w \models \mathbf{G}(b\mathbf{U}c)$ to $w \models \mathbf{G}(b\mathbf{W}c)$, a formula of νLTL, and so we define $\varphi[X]_\nu \coloneqq \mathbf{G}(b\mathbf{W}c)$.

The formal definition is now as follows:

Definition 4.7. *Let φ be a formula and let $X \subseteq \mu(\varphi)$ be a set of formulas. The formula $\varphi[X]_\nu$ is inductively defined as follows for the interesting cases* $\mathbf{U}$ *and* $\mathbf{M}$*:*

$$(\varphi\mathbf{U}\psi)[X]_\nu = \begin{cases} (\varphi[X]_\nu)\mathbf{W}(\psi[X]_\nu) & \text{if } \varphi\mathbf{U}\psi \in X \\ \mathbf{ff} & \text{otherwise.} \end{cases}$$

$$(\varphi\mathbf{M}\psi)[X]_\nu = \begin{cases} (\varphi[X]_\nu)\mathbf{R}(\psi[X]_\nu) & \text{if } \varphi\mathbf{M}\psi \in X \\ \mathbf{ff} & \text{otherwise.} \end{cases}$$

and for all other cases as a simple recursive descent:

$$\begin{array}{lll} \mathbf{tt}[X]_\nu = \mathbf{tt} & & (\mathbf{X}\varphi)[X]_\nu = \mathbf{X}(\varphi[X]_\nu) \\ \mathbf{ff}[X]_\nu = \mathbf{ff} & & \\ a[X]_\nu = a & & (\varphi\mathbf{U}\psi)[Y]_\nu = \ldots \\ \neg a[X]_\nu = \neg a & & (\varphi\mathbf{M}\psi)[Y]_\nu = \ldots \\ (\varphi \wedge \psi)[X]_\nu = \varphi[X]_\nu \wedge \psi[X]_\nu & & (\varphi\mathbf{R}\psi)[X]_\nu = (\varphi[X]_\nu)\mathbf{R}(\psi[X]_\nu) \\ (\varphi \vee \psi)[X]_\nu = \varphi[X]_\nu \vee \psi[X]_\nu & & (\varphi\mathbf{W}\psi)[X]_\nu = (\varphi[X]_\nu)\mathbf{W}(\psi[X]_\nu) \end{array}$$

We now introduce, in a dual way, a formula $\varphi[Y]_\mu \in \mu LTL$ such that $\mathcal{FG}_w^\varphi = Y = \mathcal{G}_w^\varphi$ implies $w \models \varphi \longleftrightarrow w \models \varphi[Y]_\mu$.

Definition 4.8. *Let φ be a formula and let $Y \subseteq \nu(\varphi)$ be a set of formulas. The formula $\varphi[Y]_\mu$ is inductively defined as follows for the interesting cases $\mathbf{R}$ and $\mathbf{W}$:*

$$(\varphi \mathbf{R} \psi)[Y]_\mu = \begin{cases} \mathbf{tt} & \text{if } \varphi \mathbf{R} \psi \in Y \\ (\varphi[Y]_\mu)\mathbf{M}(\psi[Y]_\mu) & \text{otherwise.} \end{cases}$$

$$(\varphi \mathbf{W} \psi)[Y]_\mu = \begin{cases} \mathbf{tt} & \text{if } \varphi \mathbf{W} \psi \in Y \\ (\varphi[Y]_\mu)\mathbf{U}(\psi[Y]_\mu) & \text{otherwise.} \end{cases}$$

and for all other cases as a simple recursive descent:

$$\begin{array}{lll} \mathbf{tt}[Y]_\mu = \mathbf{tt} & & (\mathbf{X}\varphi)[Y]_\mu = \mathbf{X}(\varphi[Y]_\mu) \\ \mathbf{ff}[Y]_\mu = \mathbf{ff} & & \\ a[Y]_\mu = a & & (\varphi \mathbf{U} \psi)[Y]_\mu = (\varphi[Y]_\mu)\mathbf{U}(\psi[Y]_\mu) \\ \neg a[Y]_\mu = \neg a & & (\varphi \mathbf{M} \psi)[Y]_\mu = (\varphi[Y]_\mu)\mathbf{M}(\psi[Y]_\mu) \\ (\varphi \wedge \psi)[Y]_\mu = \varphi[Y]_\mu \wedge \psi[Y]_\mu & & (\varphi \mathbf{R} \psi)[Y]_\mu = \ldots \\ (\varphi \vee \psi)[Y]_\mu = \varphi[Y]_\mu \vee \psi[Y]_\mu & & (\varphi \mathbf{W} \psi)[Y]_\mu = \ldots \end{array}$$

Definitions for F and G. We will often use in examples formulas containing $\mathbf{F}$ and $\mathbf{G}$. In order to keep examples readable we add the following two additional, shorter, and language-equivalent[1] definitions:

$$(\mathbf{F}\varphi)[X]_\nu \coloneqq \begin{cases} \mathbf{tt} & \text{if } \mathbf{F}\varphi \in X \\ \mathbf{ff} & \text{otherwise.} \end{cases}$$

$$(\mathbf{G}\varphi)[Y]_\mu \coloneqq \begin{cases} \mathbf{tt} & \text{if } \mathbf{G}\varphi \in Y \\ \mathbf{ff} & \text{otherwise.} \end{cases}$$

Example 4.9. *Let $\varphi = ((a\mathbf{W}b) \wedge \mathbf{F}c) \vee a\mathbf{U}d$. We have:*

$$\begin{array}{lcll} \varphi[\{\mathbf{F}c\}]_\nu & = & ((a\mathbf{W}b) \wedge \mathbf{tt}) \vee \mathbf{ff} & \sim_p \; a\mathbf{W}b \\ \varphi[\{a\mathbf{U}d\}]_\nu & = & ((a\mathbf{W}b) \wedge \mathbf{ff}) \vee a\mathbf{W}d & \sim_p \; a\mathbf{W}d \\ \varphi[\emptyset]_\nu & = & ((a\mathbf{W}b) \wedge \mathbf{ff}) \vee \mathbf{ff} & \sim_p \; \mathbf{ff} \\ \varphi[\{a\mathbf{W}b\}]_\mu & = & (\mathbf{tt} \wedge \mathbf{F}c) \vee a\mathbf{U}d & \sim_p \; \mathbf{F}c \vee a\mathbf{U}d \\ \varphi[\emptyset]_\mu & = & (a\mathbf{U}b \wedge \mathbf{F}c) \vee a\mathbf{U}d & \end{array}$$

4.3 Utilising $\varphi[X]_\nu$ and $\varphi[Y]_\mu$

The following lemma states the fundamental properties of $\varphi[X]_\nu$ and $\varphi[Y]_\mu$. As announced above, for a μ-stable word w we can reduce the problem $w \models \varphi$ to $w \models \varphi[X]_\nu$, and for a ν-stable word to $w \models \varphi[Y]_\mu$. However, there is more: If we only know $X \subseteq \mathcal{GF}_w$, then we can still infer $w \models \varphi$ from $w \models \varphi[X]_\nu$, only the implication in the other direction fails.

[1] $\mathbf{tt}\mathbf{W}\varphi \sim_l \mathbf{tt}$ and $\mathbf{ff}\mathbf{M}\varphi \sim_l \mathbf{ff}$

Lemma 4.10. *Let φ be a formula and let w be a word.*

- *Let $X \subseteq \mu(\varphi)$ be a set of formulas.*
  1. *If $\mathcal{F}_w^\varphi \subseteq X$ and $w \models \varphi$, then $w \models \varphi[X]_\nu$.*
  2. *If $X \subseteq \mathcal{GF}_w^\varphi$ and $w \models \varphi[X]_\nu$, then $w \models \varphi$.*

  In particular:
  3. *If $\mathcal{F}_w^\varphi = X = \mathcal{GF}_w^\varphi$, then $w \models \varphi \iff w \models \varphi[X]_\nu$.*
- *Let $Y \subseteq \nu(\varphi)$ be a set of formulas.*
  4. *If $\mathcal{FG}_w^\varphi \subseteq Y$ and $w \models \varphi$, then $w \models \varphi[Y]_\mu$.*
  5. *If $Y \subseteq \mathcal{G}_w^\varphi$ and $w \models \varphi[Y]_\mu$, then $w \models \varphi$.*

  In particular:
  6. *If $\mathcal{FG}_w^\varphi = Y = \mathcal{G}_w^\varphi$, then $w \models \varphi \iff w \models \varphi[Y]_\mu$.*

Proof. All parts are proved by a straightforward structural induction on φ. We consider for (1), (2), (4), and (5) only two representative cases of the induction. We moved the parts for (2), (4), and (5) to Section 4.A, since the proofs are repetitive and do not give additional insight.

(1) Assume $\mathcal{F}_w^\varphi \subseteq X$. Then $\mathcal{F}_{w_i}^\varphi \subseteq X$ for all $i \geq 0$. We prove the following stronger statement via structural induction on φ, where we consider one representative of the 'interesting' cases and one of the 'straightforward' cases:

$$\forall i.\, (\, (w_i \models \varphi) \implies (w_i \models \varphi[X]_\nu)\,)$$

- Case $\varphi = \psi_1 \mathbf{U} \psi_2$: Let $i \geq 0$ arbitrary and assume $w_i \models \psi_1 \mathbf{U} \psi_2$. Then $\psi_1 \mathbf{U} \psi_2 \in \mathcal{F}_{w_i}^\varphi$ and so $\varphi \in X$. We prove $w_i \models (\psi_1 \mathbf{U} \psi_2)[X]_\nu$:

$$\begin{aligned}
& w_i \models \psi_1 \mathbf{U} \psi_2 \\
\implies & w_i \models \psi_1 \mathbf{W} \psi_2 \\
\iff & \forall j.\, w_{i+j} \models \psi_1 \vee \exists k \leq j.\, w_{i+k} \models \psi_2 \\
\implies & \forall j.\, w_{i+j} \models \psi_1[X]_\nu \vee \exists k \leq j.\, w_{i+k} \models \psi_2[X]_\nu && \text{(induction hypothesis)} \\
\implies & w_i \models (\psi_1[X]_\nu) \mathbf{W} (\psi_2[X]_\nu) \\
\iff & w_i \models (\psi_1 \mathbf{U} \psi_2)[X]_\nu && (\varphi \in X, \text{ Definition 4.7})
\end{aligned}$$

- Case $\varphi = \psi_1 \vee \psi_2$: Let $i \geq 0$ arbitrary and assume $w_i \models \psi_1 \vee \psi_2$:

$$\begin{aligned}
& w_i \models \psi_1 \vee \psi_2 \\
\iff & w_i \models \psi_1 \vee w_i \models \psi_2 \\
\implies & w_i \models \psi_1[X]_\nu \vee w_i \models \psi_2[X]_\nu && \text{(induction hypothesis)} \\
\iff & w_i \models (\psi_1 \vee \psi_2)[X]_\nu && \text{(Definition 4.7)}
\end{aligned}$$

□

Lemma 4.10 suggests to decide $w \models \varphi$ by 'trying out' all possible sets X. In fact (2) shows that the strategy of checking for every set X if both $X \subseteq \mathcal{GF}_w^\varphi$ and $w \models \varphi[X]_\nu$ hold is sound.

Example 4.11. *Consider* $\varphi = \mathbf{GF}a \vee \mathbf{GF}(b \wedge \mathbf{G}c)$. *Since* $\mu(\varphi) = \{\mathbf{F}a, \mathbf{F}(b \wedge \mathbf{G}c)\}$, *there are four possible* X*'s to be tried out:* $\emptyset$, $\{\mathbf{F}a\}$, $\{\mathbf{F}(b \wedge \mathbf{G}c)\}$, *and* $\{\mathbf{F}a, \mathbf{F}(b \wedge \mathbf{G}c)\}$. *For* $X = \emptyset$ *we get* $\varphi[X]_\nu \sim_l \mathbf{ff}$, *indicating that if neither* a *nor* $b \wedge \mathbf{G}c$ *hold infinitely often, then* φ *cannot hold. For the other three possibilities (*a *holds infinitely often,* $b \wedge \mathbf{G}c$ *holds infinitely often, or both) there are words satisfying* φ, *like* a^ω, $\{b, c\}^\omega$, *and* $\{a, b, c\}^\omega$.

However, there are still two questions open. First, is this strategy complete? Part (3) of Lemma 4.10 shows that it is complete for μ-stable words: Indeed, in this case there is a set X such that $\mathcal{GF}^\varphi_w = X = \mathcal{F}^\varphi_w$, and for this particular set $w \models \varphi[X]_\nu$ holds. For words that are not μ-stable, we will use the existence of μ-stable suffixes: Instead of checking $w \models \varphi[X]_\nu$, we will check the existence of a suffix w_i such that $w_i \models \mathit{af}(\varphi, w_{0i})[X]_\nu$. This will happen in Section 4.5. The second open question is simply how to check $X \subseteq \mathcal{GF}^\varphi_w$ and we deal with it in Section 4.4.

4.4 Checking $X \subseteq \mathcal{GF}$ and $Y \subseteq \mathcal{FG}$

Consider again the formula $\varphi = \mathbf{GF}a \vee \mathbf{GF}(b \wedge \mathbf{G}c)$ from Example 4.11. If $X = \{\mathbf{F}a\}$, then checking whether X is a correct advice (i.e., whether $X \subseteq \mathcal{GF}^\varphi_w$ holds) is easy, because $\mathbf{GFF}a \in \mathbf{GF}(\mu LTL)$ and we have several automata constructions – deterministic, as well as nondeterministic – on our hand for this fragment. In contrast, for $X = \{\mathbf{F}(b \wedge \mathbf{G}c)\}$ this is not so. In this case it would turn out to be useful if we had an advice $Y = \{\mathbf{G}c\}$ promising that $\mathbf{G}c$ holds almost always, as is the case for e.g. $\emptyset^5(\{b, c\}\{c\})^\omega$. Indeed, we could easily check correctness of this advice, because $\mathbf{FGG}c \in \mathbf{FG}(\nu LTL)$, and with its help checking $\mathbf{GF}(b \wedge \mathbf{G}c)$ reduces to checking $\mathbf{GF}(b \wedge \mathbf{tt}) \sim_l \mathbf{GF}b$, which is also easy.

One of the main ingredients of our approach is that in order to verify a promise $X \subseteq \mathcal{GF}^\varphi_w$ we can rely on a promise $Y \subseteq \mathcal{FG}^\varphi_w$ about subformulas of X, and vice versa. There is no circularity in this rely/guarantee reasoning because the subformula order is well founded, and we eventually reach formulas ψ such that $\psi[X]_\nu = \psi$ or $\psi[Y]_\mu = \psi$. This argument is formalised in the next lemma. The first part of the lemma states that mutually assuming correctness of the other promise is correct. The second part states that, loosely speaking, this rely/guarantee method is complete: it can prove that $X = \mathcal{GF}^\varphi_w$ and $Y = \mathcal{FG}^\varphi_w$ hold.

Lemma 4.12. *Let* φ *be a formula and let* w *be a word.*

1. *For every* $X \subseteq \mu(\varphi)$ *and* $Y \subseteq \nu(\varphi)$, *if*

$$\forall \psi \in X.\ w \models \mathbf{GF}(\psi[Y]_\mu)$$
$$\forall \psi \in Y.\ w \models \mathbf{FG}(\psi[X]_\nu)$$

   then $X \subseteq \mathcal{GF}^\varphi_w$ *and* $Y \subseteq \mathcal{FG}^\varphi_w$.

2. *If* $X = \mathcal{GF}^\varphi_w$ *and* $Y = \mathcal{FG}^\varphi_w$, *then:*

$$\forall \psi \in X.\ w \models \mathbf{GF}(\psi[Y]_\mu)$$
$$\forall \psi \in Y.\ w \models \mathbf{FG}(\psi[X]_\nu)$$

Proof. (1) Let $X \subseteq \mu(\varphi)$ and $Y \subseteq \nu(\varphi)$. Observe that $X \cap Y = \emptyset$. Let $n \coloneqq |X \cup Y|$. Let $\psi_1, \ldots, \psi_n$ be an enumeration of $X \cup Y$ compatible with the subformula order, i.e., if ψ_i is a subformula of ψ_j, then $i \leq j$. Finally, let $(X_0, Y_0), (X_1, Y_1), \ldots, (X_n, Y_n)$ be the unique sequence of pairs satisfying:

- $(X_0, Y_0) = (\emptyset, \emptyset)$ and $(X_n, Y_n) = (X, Y)$.
- For every $0 < i \leq n$, if $\psi_i \in X$ then $X_i \setminus X_{i-1} = \{\psi_i\}$ and $Y_i = Y_{i-1}$, and if $\psi_i \in Y$, then $X_i = X_{i-1}$ and $Y_i \setminus Y_{i-1} = \{\psi_i\}$.

We prove $X_i \subseteq \mathcal{GF}^{\varphi}_w$ and $Y_i \subseteq \mathcal{FG}^{\varphi}_w$ for every $0 \leq i \leq n$ by induction on i. For $i = 0$ the result follows immediately from $X_0 = \emptyset = Y_0$. For $i > 0$ we consider two cases:

- $\psi_i \in Y$, i.e., $X_i = X_{i-1}$ and $Y_i \setminus Y_{i-1} = \{\psi_i\}$.

  By induction hypothesis and $X_i = X_{i-1}$ we have $X_i \subseteq \mathcal{GF}^{\varphi}_w$ and $Y_{i-1} \subseteq \mathcal{FG}^{\varphi}_w$. We prove $\psi_i \in \mathcal{FG}^{\varphi}_w$, i.e., $w \models \mathbf{FG}\psi_i$, in three steps.

  - Claim 1: $\psi_i[X]_\nu = \psi_i[X_i]_\nu$.

    By the definition of the $\cdot[\cdot]_\nu$ mapping, $\psi_i[X]_\nu$ is completely determined by the μ-subformulas of ψ_i that belong to X. By the definition of the sequence $(X_0, Y_0), \ldots, (X_n, Y_n)$, a μ-subformula of ψ_i belongs to X if and only if it belongs to X_i, and we are done.
  - Claim 2: $X_i \subseteq \mathcal{GF}^{\varphi}_{w_k}$ for every $k \geq 0$.

    Follows immediately from $X_i \subseteq \mathcal{GF}^{\varphi}_w$.
  - Proof of $w \models \mathbf{FG}\psi_i$.

    By the assumption of the lemma we have $w \models \mathbf{FG}(\psi_i[X]_\nu)$, and so, by Claim 1, $w \models \mathbf{FG}(\psi_i[X_i]_\nu)$. So there exists an index j such that $w_{j+k} \models \psi_i[X_i]_\nu$ for every $k \geq 0$. By Claim 2 we further have $X_i \subseteq \mathcal{GF}^{\varphi}_{w_{j+k}}$ for every $j, k \geq 0$. So we can apply part (2) of Lemma 4.10 to X_i, w_{j+k}, and ψ_i, which yields $w_{j+k} \models \psi_i$ for every $k \geq 0$. So $w \models \mathbf{FG}\psi_i$.

- $\psi_i \in X$, i.e., $X_i \setminus X_{i-1} = \{\psi_i\}$ and $Y_i = Y_{i-1}$.

  By induction hypothesis we have in this case $X_{i-1} \subseteq \mathcal{GF}^{\varphi}_w$ and $Y_i \subseteq \mathcal{FG}^{\varphi}_w$. We prove $\psi_i \in \mathcal{GF}^{\varphi}_w$, i.e., $w \models \mathbf{GF}\psi_i$ in three steps.

  - Claim 1: $\psi_i[Y]_\mu = \psi_i[Y_i]_\mu$.

    The claim is proved as in the previous case.
  - Claim 2: There is an $j \geq 0$ such that $Y_i \subseteq \mathcal{G}^{\varphi}_{w_k}$ for every $k \geq j$.

    Follows immediately from $Y_i \subseteq \mathcal{FG}^{\varphi}_w$.
  - Proof of $w \models \mathbf{GF}\psi_i$.

    By the assumption of the lemma we have $w \models \mathbf{GF}(\psi_i[Y]_\mu)$. Let j be the index of Claim 2. By Claim 1 we have $w \models \mathbf{GF}(\psi_i[Y_i]_\mu)$, and so there exist infinitely many $k \geq j$ such that $w_k \models \psi_i[Y_i]_\mu$. By Claim 2 we further have $Y_i \subseteq \mathcal{G}^{\varphi}_{w_k}$. So we can apply part (5) of Lemma 4.10 to Y_i, w_k, and ψ_i, which yields $w_k \models \psi_i$ for infinitely many $k \geq j$. So $w \models \mathbf{GF}\psi_i$.

(2) Let $\psi \in \mathcal{GF}^{\varphi}_w$. We have $w \models \mathbf{GF}\psi$, and so $w_i \models \psi$ for infinitely many $i \geq 0$. Since $\mathcal{FG}^{\varphi}_{w_i} = \mathcal{FG}^{\varphi}_w$ for every $i \geq 0$, part (4) of Lemma 4.10 can be applied to w_i, $\mathcal{FG}^{\varphi}_{w_i}$, and ψ. This yields $w_i \models \psi[\mathcal{FG}^{\varphi}_w]_\mu$ for infinitely many $i \geq 0$ and thus $w \models \mathbf{GF}(\psi[\mathcal{FG}^{\varphi}_w]_\mu)$.

Let $\psi \in \mathcal{FG}^{\varphi}_w$. Since $w_i \models \mathbf{FG}\psi$, there is an index j such that $w_{j+k} \models \psi$ for every $k \geq 0$. By Lemma 4.5 the index j can be chosen so that it also satisfies $\mathcal{GF}^{\varphi}_w = \mathcal{F}^{\varphi}_{w_{j+k}} = \mathcal{GF}^{\varphi}_{w_{j+k}}$

for every $k \geq 0$. So part (1) of Lemma 4.10 can be applied to $\mathcal{F}^{\varphi}_{w_{j+k}}$, w_{j+k}, and ψ. This yields $w_{j+k} \models \psi[\mathcal{GF}^{\varphi}_{w}]_{\nu}$ for every $k \geq 0$ and thus $w \models \mathbf{FG}(\psi[\mathcal{GF}^{\varphi}_{w}]_{\nu})$. □

Example 4.13. *Let $\varphi = \mathbf{F}(a \wedge \mathbf{G}(b \vee \mathbf{F}c))$, $X = \{\varphi\}$, and $Y = \{\mathbf{G}(b \vee \mathbf{F}c)\}$.*

- *The condition $\forall \psi \in X.\ w \models \mathbf{GF}(\psi[Y]_{\mu})$ becomes*

$$w \models \mathbf{GF}\,(\varphi[Y]_{\mu}) = \mathbf{GF}(\mathbf{F}a \wedge \mathbf{tt}) \sim_{l} \mathbf{GF}a$$

- *The condition $\forall \psi \in Y.\ w \models \mathbf{FG}(\psi[X]_{\nu})$ becomes*

$$w \models \mathbf{FG}\,((\mathbf{G}(b \vee \mathbf{F}c))[X]_{\nu}) = \mathbf{FG}(\mathbf{G}(b \vee \mathbf{ff})) \sim_{l} \mathbf{FG}b$$

Applying Lemma 4.12 (1) to this we then obtain that $w \models \mathbf{GF}a \wedge \mathbf{FG}b$ implies $\varphi \in \mathcal{GF}^{\varphi}_{w}$ and $\mathbf{G}(b \vee \mathbf{F}c) \in \mathcal{FG}^{\varphi}_{w}$.

4.5 The Master Theorem: Logical Characterisation of LTL

Putting together Lemma 4.10 and Lemma 4.12, we arrive at the core of the dissertation: a decomposition of LTL formulas into simpler fragments, which we will use as 'Master Theorem' for the construction of automata.

Theorem 4.14 (Master Theorem)**.** *Let φ be a formula and let w be a word. Then $w \models \varphi$ if and only if there exist $X \subseteq \mu(\varphi)$ and $Y \subseteq \nu(\varphi)$ satisfying:*

1. $\exists i.\, w_i \models af(\varphi, w_{0i})[X]_{\nu}$
2. $\forall \psi \in X.\, w \models \mathbf{GF}(\psi[Y]_{\mu})$
3. $\forall \psi \in Y.\ w \models \mathbf{FG}(\psi[X]_{\nu})$

Before proving the theorem, let us interpret it in informal terms. The Master Theorem states that in order to decide $w \models \varphi$ we can guess two sets $X \subseteq \mu(\varphi)$ and $Y \subseteq \nu(\varphi)$ and an index i, and then proceed as follows: verify $Y \subseteq \mathcal{FG}^{\varphi}_{w}$ assuming that $X \subseteq \mathcal{GF}^{\varphi}_{w}$ holds (3), verify $X \subseteq \mathcal{GF}^{\varphi}_{w}$ assuming that $Y \subseteq \mathcal{FG}^{\varphi}_{w}$ holds (2), and verify $w_i \models af(\varphi, w_{0i})[X]_{\nu}$ assuming that $X \subseteq \mathcal{GF}^{\varphi}_{w}$ holds (1). The procedure is sound by Lemmas 4.10 and 4.12, and complete because the guess where $X := \mathcal{GF}^{\varphi}_{w}$, $Y := \mathcal{FG}^{\varphi}_{w}$, and i is a stabilisation point of w, is guaranteed to succeed.

Example 4.15. *Let $\varphi = \mathbf{F}(a \wedge \mathbf{G}(b \vee \mathbf{F}c))$ as in Example 4.13, and let $\varphi' = d\mathbf{U}(e \wedge \varphi)$. For $X = \{\varphi, \varphi'\}$, $Y = \{\mathbf{G}(b \vee \mathbf{F}c)\}$, and $i = 0$ the Master Theorem yields that $w \models \varphi'$ is implied by*

1. $w \models (d\mathbf{U}(e \wedge \varphi))[X]_{\nu} = d\mathbf{W}(e \wedge \varphi[X]_{\nu}) = d\mathbf{W}(e \wedge \mathbf{tt}) \sim_{l} d\mathbf{W}e$,
2. $w \models \mathbf{GF}(\varphi[Y]_{\mu}) = \mathbf{GF}(\mathbf{F}(a \wedge \mathbf{tt})) \sim_{l} \mathbf{GF}a$,
   $w \models \mathbf{GF}(\varphi'[Y]_{\mu}) = \mathbf{GF}(d\mathbf{U}(e \wedge \mathbf{F}(a \wedge \mathbf{tt}))) \sim_{l} \mathbf{GF}(e \wedge \mathbf{F}a)$, *and*
3. $w \models \mathbf{FG}((\mathbf{G}(b \vee \mathbf{F}c))[X]_{\nu}) = \mathbf{FG}(\mathbf{G}(b \vee \mathbf{ff})) \sim_{l} \mathbf{FG}b$.

For $X' = \{\varphi\}$, $Y' = \{\mathbf{G}(b \vee \mathbf{F}c)\}$, and $i' = 0$, condition (1) is not fulfilled, since $w \not\models (d\mathbf{U}(e \wedge \varphi))[X']_\nu = \mathbf{ff}$. Logically this is sound, because $w \models \mathbf{ff}$ implies everything. We just do not derive any useful information from this particular instance. The Master Theorem does not promise that every *correct guess for X and Y will give us an actual chance to prove the property. Let us now choose $i' = 1$ and let us assuming the word is $w[0] = \{e\}$. Then the Master Theorem says that $w \models \varphi'$ is implied by*

1. $w \models (af(d\mathbf{U}(e \wedge \varphi), \{e\})[X']_\nu = ((\mathbf{tt} \wedge (\varphi \vee \dots)) \vee (\mathbf{ff} \wedge \varphi'))\,[X']_\nu \sim_l \mathbf{tt}$,
2. $w \models \mathbf{GF}(\varphi[Y']_\mu) = \mathbf{GF}(\mathbf{F}(a \wedge \mathbf{tt})) \sim_l \mathbf{GF}a$,
3. $w \models \mathbf{FG}((\mathbf{G}(b \vee \mathbf{F}c))[X']_\nu) = \mathbf{FG}(\mathbf{G}(b \vee \mathbf{ff})) \sim_l \mathbf{FG}b$.

Thus theorem is offering us yet another possibility to prove $w \models \varphi$. One that is even simpler than the first one.

Proof of the Master Theorem. ($\Rightarrow$) Assume $w \models \varphi$, and set $X := \mathcal{GF}_w^\varphi$ and $Y := \mathcal{FG}_w^\varphi$. Properties (2) and (3) follow from Lemma 4.12. For property (1), let i be an index such that $\mathcal{F}_{w_i}^\varphi = \mathcal{GF}_{w_i}^\varphi$; this index exists by Lemma 4.5. By Lemma 3.3 we have $w_i \models af(\varphi, w_{0i})$, and by Lemma 4.10 (1) $w_i \models af(\varphi, w_{0i})[X]_\nu$.

($\Leftarrow$) Assume that properties (1-3) hold for sets $X \subseteq \mu(\varphi)$, $Y \subseteq \nu(\varphi)$ and an index i. By Lemma 4.12 (1) we have $X \subseteq \mathcal{GF}_w^\varphi$, and so $X \subseteq \mathcal{GF}_{w_i}^\varphi$. From $w_i \models af(\varphi, w_{0i})[X]_\nu$ we obtain $w_i \models af(\varphi, w_{0i})$ with Lemma 4.10 (2). Finally, Lemma 3.3 yields then $w \models \varphi$. □

A blue-print for Automata Constructions. Observe that $af(\varphi, w_{0i})[X]_\nu$, $\mathbf{GF}(\psi[Y]_\mu)$, and $\mathbf{FG}(\psi[X]_\nu)$ are formulas of the LTL fragments νLTL, $\mathbf{GF}(\mu LTL)$, and $\mathbf{FG}(\nu LTL)$. For these fragments we can build automata using Lemma 3.9 with *af* as a basis. The constructions are presented in Sections 5.1 and 6.2. This suggests the following approach to constructing automata for LTL formulas:

Let $L^j_{\varphi,X,Y}$ be the language of all words that satisfy condition (j) of the Master Theorem for the formula φ and the sets X and Y. The Master Theorem can then be reformulated as:

$$\mathcal{L}(\varphi) = \bigcup_{\substack{X \subseteq \mu(\varphi) \\ Y \subseteq \nu(\varphi)}} L^1_{\varphi,X,Y} \cap L^2_{\varphi,X,Y} \cap L^3_{\varphi,X,Y}$$

Therefore, given an automata model effectively closed under union and intersection, in order to construct automata for all of LTL it suffices to exhibit automata recognising $L^1_{\varphi,X,Y}, L^2_{\varphi,X,Y}, L^3_{\varphi,X,Y}$. In the following chapter we consider the case of DRAs, and then we proceed to NBAs and LDBAs.

Remark 4.16. *The formulation of the Master Theorem is almost symmetric and we only have one impurity that breaks this symmetry: the advice function $\cdot[X]_\nu$ in condition (1). Can we also use $\cdot[Y]_\mu$ at that location? Yes, the theorem is also true for $\cdot[Y]_\mu$ at that location and the corresponding proof is almost identical to the one we have seen. So why are we choosing $\cdot[X]_\nu$? The application of $\cdot[X]_\nu$ yields a formula of the fragment νLTL. The languages corresponding to this fragment are characterised by (finite)* bad prefixes. *Whenever an ω-word does* not *satisfy $\psi[X]_\nu$, we can learn this from a finite prefix. Let us now consider $\cdot[Y]_\mu$. The languages corresponding to the fragment $\cdot[Y]_\mu$ are characterised by (finite)* good prefixes *and if an ω-word satisfies $\psi[Y]_\mu$, then we know*

this after reading a finite prefix, but in general there is no finite prefix for proving that $\psi[Y]_\mu$ *is* not *satisfied.*

This distinction allows us in the case of $\psi[X]_\nu$ *to sequentially test* i*'s for condition (1), since either we have not yet identified a bad prefix and thus the word can be continued such that (1) holds, or we have found a bad prefix and we can move to another candidate for* i*. Lemma 4.20 shows how to do this efficiently. However, this argument does not work for* $\psi[Y]_\mu$ *and in this case we need to check different* i*'s in parallel. This can be easily done if nondeterminism is available, but it is non-trivial for a deterministic automaton.*

4.6 Variants of the Master Theorem

4.6.1 Restricted Guessing

Let us consider the formula $\varphi = \mathbf{F}(a \wedge \mathbf{G}(b \vee \mathbf{F}c))$ from Example 4.15 and some word w. The Master Theorem (Theorem 4.14) dictates that we need to check *all* possible selections of $X \subseteq \mu(\varphi)$ and $Y \subseteq \nu(\varphi)$ in order to determine $w \models \varphi$. However, some of the guesses seem to be superfluous: consider for example $X = \{\varphi, \mathbf{F}c\}$ and $Y = \nu(\varphi)$. Why is it relevant that φ holds infinitely often? We are only interested in $w \models \varphi$ and not in $w \models \mathbf{GF}\varphi$. Obviously, it is unnecessary work to check a top-level $\mathbf{F}$ operator for repeated satisfaction. Hence we discard all sets X that contain a least-fixed-point operator that is not nested within a greatest-fixed-point operator, because only in the context of a greatest-fixed-point operator we are interested in repeated satisfaction. We also restrict Y analogously. We denote this restriction by $\Downarrow(\varphi)$ and use $\Downarrow_\nu$ to indicate that we passed during the recursive descent on φ a greatest-fixed-point operator.

Definition 4.17. *Let* φ *be a formula. The* restriction sets $\Downarrow(\varphi)$ *and* $\Downarrow_\nu(\varphi)$ *are inductively defined as follows:*

$$
\begin{array}{lll@{\qquad}lll}
\Downarrow(\mathbf{tt}) & = \emptyset & & \Downarrow(\mathbf{X}\varphi) & = \Downarrow(\varphi) \\
\Downarrow(\mathbf{ff}) & = \emptyset & & & \\
\Downarrow(a) & = \emptyset & & \Downarrow(\varphi\mathbf{U}\psi) & = \Downarrow(\varphi) \cup \Downarrow(\psi) \\
\Downarrow(\neg a) & = \emptyset & & \Downarrow(\varphi\mathbf{M}\psi) & = \Downarrow(\varphi) \cup \Downarrow(\psi) \\
\Downarrow(\varphi \wedge \psi) & = \Downarrow(\varphi) \cup \Downarrow(\psi) & & \Downarrow(\varphi\mathbf{R}\psi) & = \Downarrow(\varphi) \cup \Downarrow_\nu(\psi) \\
\Downarrow(\varphi \vee \psi) & = \Downarrow(\varphi) \cup \Downarrow(\psi) & & \Downarrow(\varphi\mathbf{W}\psi) & = \Downarrow_\nu(\varphi) \cup \Downarrow(\psi)
\end{array}
$$

$$
\begin{array}{lll@{\qquad}lll}
\Downarrow_\nu(\mathbf{tt}) & = \emptyset & & \Downarrow_\nu(\mathbf{X}\varphi) & = \Downarrow_\nu(\varphi) \\
\Downarrow_\nu(\mathbf{ff}) & = \emptyset & & & \\
\Downarrow_\nu(a) & = \emptyset & & \Downarrow_\nu(\varphi\mathbf{U}\psi) & = \mu(\varphi\mathbf{U}\psi) \cup \nu(\varphi\mathbf{U}\psi) \\
\Downarrow_\nu(\neg a) & = \emptyset & & \Downarrow_\nu(\varphi\mathbf{M}\psi) & = \mu(\varphi\mathbf{M}\psi) \cup \nu(\varphi\mathbf{M}\psi) \\
\Downarrow_\nu(\varphi \wedge \psi) & = \Downarrow_\nu(\varphi) \cup \Downarrow_\nu(\psi) & & \Downarrow_\nu(\varphi\mathbf{R}\psi) & = \Downarrow_\nu(\varphi) \cup \Downarrow_\nu(\psi) \\
\Downarrow_\nu(\varphi \vee \psi) & = \Downarrow_\nu(\varphi) \cup \Downarrow_\nu(\psi) & & \Downarrow_\nu(\varphi\mathbf{W}\psi) & = \Downarrow_\nu(\varphi) \cup \Downarrow_\nu(\psi)
\end{array}
$$

Proposition 4.18. *Let* φ *be a formula and let* w *be a word. Then* $w \models \varphi$ *if and only if there exist* $X \subseteq \mu(\varphi) \cap \Downarrow(\varphi)$ *and* $Y \subseteq \nu(\varphi) \cap \Downarrow(\varphi)$ *satisfying:*

1. $\exists i.\, w_i \models \mathit{af}(\varphi, w_{0i})[X]_\nu$
2. $\forall \psi \in X.\, w \models \mathbf{GF}(\psi[Y]_\mu)$
3. $\forall \psi \in Y.\, w \models \mathbf{FG}(\psi[X]_\nu)$

This sharper version of the Master Theorem (Theorem 4.14) with its usage of $\Downarrow$ can be intuitively understood as first peeling off a layer of least-fixed-point operators, since repeated satisfaction is irrelevant for these, and then a layer of greatest-fixed-point operators, since $\cdot[\cdot]_\nu$ does affect these and they do not appear within a potential advice X. Thus we only need to consider the empty advice sets $X = \emptyset$ and $Y = \emptyset$ for a formula with just one alternation of least- to greatest-fixed-point operators. Before we dive into the proof, we have a look at two other things: First, we revisit and update Example 4.15. Second, we introduce and discuss two small technical lemmas required for the proof.

Example 4.19. *Let $\varphi = \mathbf{F}(a \wedge \mathbf{G}(b \vee \mathbf{F}c))$ and let $\varphi' = d\mathbf{U}(e \wedge \varphi)$ as in Example 4.15. We then compute:*

$$\Downarrow(\varphi') = \Downarrow(\varphi) = \Downarrow(\mathbf{G}(b \vee \mathbf{F}c)) = \Downarrow_\nu(b \vee \mathbf{F}c) = \{\mathbf{F}c\}$$

Thus we only need to consider the advice $(X, Y) = (\emptyset, \emptyset)$ and the advice $(X', Y') = (\{\mathbf{F}c\}, \emptyset)$. For $i = 0$ condition (1) of Proposition 4.18 is not true for both possible guesses, since we have $w \not\models \varphi'[X]_\nu = \varphi'[X']_\nu = \mathbf{ff}$, and thus we cannot determine $w \models \varphi$. Let now $i = 1$ and assume $w[0] = \{a, b, c, d, e\}$. Then $w \models \varphi$ is implied by one of the two following groups of conditions:

1. – $w \models af((d\mathbf{U}(e \wedge \varphi), w[0])[X]_\nu \sim_l (\mathbf{G}b)$
2. – $w \models af((d\mathbf{U}(e \wedge \varphi), w[0])[X']_\nu \sim_l \mathbf{tt}$
   – $w \models \mathbf{GF}(\mathbf{F}c[Y']_\mu) = \mathbf{GF}(\mathbf{F}c) \sim_l \mathbf{GF}c$

Thus the number of advice pairs (X, Y) we need to consider is reduced considerably at the expense of a longer 'waiting time' – meaning a larger index i – for $w_i \models af(\varphi, w_{0i})[X]_\nu$ to become true.

The first technical lemma, which is also needed for the later presented DRA constructions, basically states that applying $\cdot[\cdot]_\nu$ in the context of property (1) can be arbitrarily delayed. The corresponding proof can be found in Section 4.A.

Lemma 4.20. *Let φ be a formula, let w be a word. If $w \models \varphi[X]_\nu$, then $w_i \models af(\varphi, w_{0i})[X]_\nu$ for every index i.*

The second technical lemma builds upon the first one and formalises the previous intuition of 'peeling off a layer of least-fixed-point operators'. The proof is again moved to Section 4.A.

Lemma 4.21. *Let φ be a formula and let w be a word. If w is μ-stable with respect to φ and $w \models \varphi$, then there exits an index i such that $w_i \models af(\varphi, w_{0i})[\mathcal{GF}^\varphi_w \cap \Downarrow(\varphi)]_\nu$.*

Using this we can now get back to the delayed proof of the sharper version of the Master Theorem:

Proof of Proposition 4.18. ($\Leftarrow$) Immediate consequence of Theorem 4.14.

($\Rightarrow$) Assume $w \models \varphi$. Let $X = \mathcal{GF}^\varphi_w$ and $Y = \mathcal{FG}^\varphi_w$. From Lemma 4.12 we know that $w \models \mathbf{GF}(\psi[Y]_\mu)$ for all $\psi \in X$ and $w \models \mathbf{FG}(\psi[X]_\nu)$ for all $\psi \in Y$. Let us define restricted versions $X_\Downarrow = X \cap \Downarrow(\varphi)$ and $Y_\Downarrow = Y \cap \Downarrow(\varphi)$. We now want to prove:

1. $\exists i.\ w_i \models af(\varphi, w_{0i})[X_\Downarrow]_\nu$
2. $\forall \psi \in X_\Downarrow.\, w \models \mathbf{GF}(\psi[Y_\Downarrow]_\mu)$
3. $\forall \psi \in Y_\Downarrow.\ w \models \mathbf{FG}(\psi[X_\Downarrow]_\nu)$

1. We obtain an index i from Lemma 4.5 such that w_i is μ-stable with respect to φ and thus also with respect to $\varphi' = \mathit{af}(\varphi, w_{0i})$ (Lemma 3.3). We then apply Lemma 4.21 to φ' and obtain and index j such that $w_j \models \mathit{af}(\varphi', w_{ij})[\mathcal{GF}^{\varphi'}_{w_i} \cap \Downarrow(\varphi')]_\nu$. With the normalisation of the sub- and superscripts and unfolding of φ we have $w_j \models \mathit{af}(\varphi, w_{0j})[\mathcal{GF}^{\varphi}_{w} \cap \Downarrow(\varphi')]_\nu$. Observe that $\Downarrow(\varphi) = \Downarrow(\mathit{af}(\varphi, w_{0i})) = \Downarrow(\varphi')$ holds by Lemma 3.3 and we can proceed to $w_j \models \mathit{af}(\varphi, w_{0j})[X \cap \Downarrow(\varphi)]_\nu$, which concludes this part.

2. Let $\psi \in X_\Downarrow$. Since $\psi \in \Downarrow(\varphi)$, we have $\nu(\psi) \subseteq \Downarrow(\varphi)$. We then derive:

$$\begin{aligned}
\psi[Y]_\mu &= \psi[Y \cap \nu(\psi)]_\mu && \text{(elements not in } \nu(\psi) \text{ can be safely ignored)} \\
&= \psi[Y \cap \nu(\psi) \cap \Downarrow(\psi)]_\mu && (\nu(\psi) \subseteq \Downarrow(\varphi)) \\
&= \psi[Y_\Downarrow]_\mu && \text{(elements not in } \nu(\psi) \text{ are ignored)}
\end{aligned}$$

   Thus $\forall \psi \in X_\Downarrow.\, w \models \mathbf{GF}(\psi[Y_\Downarrow]_\mu)$.

3. This is proven analogous to the previous part.

□

4.6.2 Asymmetric Master Theorem

The translation from LTL to LDBAs presented in [SEJK16] also relies on a decomposition into 'simple' LTL fragments, but requires a restricted form of nondeterminism for the automata construction. We compare our previous work with the Master Theorem by reformulating the decomposition of [SEJK16] in our novel notational and theoretical framework.

Proposition 4.22 ([SEJK16])**.** *Let φ be a formula and let w be a word. Then $w \models \varphi$ if and only if there exist $i \geq 0$ and $Y \subseteq \nu(\varphi)$ satisfying:*

1. $w_i \models \mathit{af}(\varphi, w_{0i})[Y]_\mu$
2. $\forall\, (\psi_1 \mathbf{R} \psi_2) \in Y.\ w_i \models \mathbf{G}(\psi_2[Y]_\mu)$
3. $\forall\, (\psi_1 \mathbf{W} \psi_2) \in Y.\, w_i \models \mathbf{G}(\psi_1[Y]_\mu \vee \psi_2[Y]_\mu)$

We consider this LTL characterisation to be 'asymmetric', since it only refers to greatest-fixed-point operators and ignores least-fixed-point operators. Consequently, the Master Theorem (Theorem 4.14) is considered to be 'symmetric', because greatest and least fixed-points are used. The 'asymmetric' LTL characterisation has several drawbacks compared to Theorem 4.14. First, it is 'less' compositional, since (1-3) all depend on the same index i. Second, it constructs formulas belonging to the fragments μLTL and $\mathbf{G}(\mu LTL)$. As one can see in [SEJK16] the construction for formulas of the fragment $\mathbf{G}(\mu LTL)$ is more involved compared to the one for $\mathbf{GF}(\mu LTL)$ presented in the next chapter. Third, we could not find a DRA construction for (1) with asymptotic optimal size that also synchronises with the automata for (2) and (3). Thus we could not come up with a simple and compositional DRA construction based on Proposition 4.22, but it can be used to build (NBAs and) LDBAs as shown in [SEJK16].

4.A Omitted Proofs

Proof of Lemma 4.10

Lemma 4.10. *Let φ be a formula and let w be a word.*

- *Let $X \subseteq \mu(\varphi)$ be a set of formulas.*
  1. *If $\mathcal{F}_w^\varphi \subseteq X$ and $w \models \varphi$, then $w \models \varphi[X]_\nu$.*
  2. *If $X \subseteq \mathcal{GF}_w^\varphi$ and $w \models \varphi[X]_\nu$, then $w \models \varphi$.*

  In particular:
  3. *If $\mathcal{F}_w^\varphi = X = \mathcal{GF}_w^\varphi$, then $w \models \varphi \iff w \models \varphi[X]_\nu$.*
- *Let $Y \subseteq \nu(\varphi)$ be a set of formulas.*
  4. *If $\mathcal{FG}_w^\varphi \subseteq Y$ and $w \models \varphi$, then $w \models \varphi[Y]_\mu$.*
  5. *If $Y \subseteq \mathcal{G}_w^\varphi$ and $w \models \varphi[Y]_\mu$, then $w \models \varphi$.*

  In particular:
  6. *If $\mathcal{FG}_w^\varphi = Y = \mathcal{G}_w^\varphi$, then $w \models \varphi \iff w \models \varphi[Y]_\mu$.*

Proof. We will now continue with the proof of the remaining parts (2), (4), and (5).

(2) Assume $X \subseteq \mathcal{GF}_w^\varphi$. Then $X \subseteq \mathcal{GF}_{w_i}^\varphi$ for all $i \geq 0$. We prove the following stronger statement via structural induction on φ:

$$\forall i.\, (\, (w_i \models \varphi[X]_\nu) \implies (w_i \models \varphi)\,)$$

- Case $\varphi = \psi_1 \mathbf{U} \psi_2$: If $\varphi \notin X$, then by definition $\varphi[X]_\nu = \mathbf{ff}$. So $w_i \not\models \varphi[X]_\nu = \mathbf{ff}$ for all i and thus the implication $(w_i \models \varphi[X]_\nu) \implies (w_i \models \varphi)$ holds for every $i \geq 0$. Assume now $\varphi \in X$. Since $X \subseteq \mathcal{GF}_w^\varphi$ we have $w_i \models \mathbf{GF}\varphi$ and so in particular $w_i \models \mathbf{F}\psi_2$. To prove the implication assume $w_i \models (\psi_1 \mathbf{U} \psi_2)[X]_\nu$ for an arbitrary fixed i. We show $w_i \models \psi_1 \mathbf{U} \psi_2$:

$$\begin{aligned}
& w_i \models (\psi_1 \mathbf{U} \psi_2)[X]_\nu \\
\iff & w_i \models (\psi_1[X]_\nu) \mathbf{W} (\psi_2[X]_\nu) && (\varphi \in X, \text{ Definition 4.7}) \\
\iff & \forall j.\, w_{i+j} \models \psi_1[X]_\nu \vee \exists k \leq j.\, w_{i+k} \models \psi_2[X]_\nu \\
\implies & \forall j.\, w_{i+j} \models \psi_1 \vee \exists k \leq j.\, w_{i+k} \models \psi_2 && (\text{induction hypothesis}) \\
\iff & w_i \models \psi_1 \mathbf{W} \psi_2 \\
\iff & w_i \models \psi_1 \mathbf{U} \psi_2 && (w_i \models \mathbf{F}\psi_2)
\end{aligned}$$

- Case $\varphi = \psi_1 \vee \psi_2$: Let $i \geq 0$ arbitrary and assume $w_i \models \psi_1 \vee \psi_2$. We have:

$$\begin{aligned}
& w_i \models (\psi_1 \vee \psi_2)[X]_\nu \\
\iff & w_i \models \psi_1[X]_\nu \vee (w_i \models \psi_2[X]_\nu && (\text{Definition 4.7}) \\
\implies & w_i \models \psi_1 \vee w_i \models \psi_2 && (\text{induction hypothesis}) \\
\iff & w_i \models \psi_1 \vee \psi_2
\end{aligned}$$

(4) Assume $\mathcal{FG}_w^\varphi \subseteq Y$. Then $\mathcal{FG}_{w_i}^\varphi \subseteq Y$ for all i. We prove the following stronger statement via structural induction on φ:

$$\forall i.\, (\, (w_i \models \varphi) \implies (w_i \models \varphi[Y]_\mu)\,)$$

– Case $\varphi = \psi_1 \mathbf{W} \psi_2$: Let $i \geq 0$ arbitrary and assume $w_i \models \varphi$. If $\varphi \in Y$ then $\varphi[Y]_\mu = \mathbf{tt}$ and so $w_i \models \varphi[Y]_\mu$ trivially holds. Assume now $\varphi \notin Y$. Since $\mathcal{FG}^\varphi_{w_i} \subseteq Y$ we have $w_i \not\models \mathbf{FG}\varphi$ and so in particular $w_i \not\models \mathbf{G}\psi_1$. We prove $w_i \models (\psi_1 \mathbf{W} \psi_2)[Y]_\mu$:

$$
\begin{aligned}
& w_i \models \psi_1 \mathbf{W} \psi_2 \\
\iff & w_i \models \psi_1 \mathbf{U} \psi_2 && (w_i \not\models \mathbf{G}\psi_1) \\
\iff & \exists j.\, w_{i+j} \models \psi_2 \wedge \forall k < j.\, w_{i+k} \models \psi_1 \\
\implies & \exists j.\, w_{i+j} \models \psi_2[Y]_\mu \wedge \forall k < j.\, w_{i+k} \models \psi_1[Y]_\mu && \text{(induction hypothesis)} \\
\iff & w_i \models (\psi_1[Y]_\mu) \mathbf{U} (\psi_2[Y]_\mu) \\
\iff & w_i \models (\psi_1 \mathbf{W} \psi_2)[Y]_\mu && (\varphi \notin Y, \text{ Definition 4.8})
\end{aligned}
$$

– Case $\varphi = \psi_1 \vee \psi_2$: Let $i \geq 0$ arbitrary and assume $w_i \models \psi_1 \vee \psi_2$. We have:

$$
\begin{aligned}
& w_i \models \psi_1 \vee \psi_2 \\
\iff & w_i \models \psi_1 \vee w_i \models \psi_2 \\
\implies & w_i \models \psi_1[Y]_\mu \vee w_i \models \psi_2[Y]_\mu && \text{(induction hypothesis)} \\
\iff & w_i \models (\psi_1 \vee \psi_2)[Y]_\mu && \text{(Definition 4.8)}
\end{aligned}
$$

(5) Assume $Y \subseteq \mathcal{G}^\varphi_w$. Then $Y \subseteq \mathcal{G}^\varphi_{w_i}$ for all i. We prove the following stronger statement via structural induction on φ:

$$\forall i.\, (\, (w_i \models \varphi[Y]_\mu) \implies (w_i \models \varphi)\,)$$

– Case $\varphi = \psi_1 \mathbf{W} \psi_2$: If $\varphi \in Y$, then since $Y \subseteq \mathcal{G}^\varphi_w$ we have $w_i \models \mathbf{G}\varphi$ and so $w_i \models \varphi$. Assume now that $\varphi \notin Y$ and $w_i \models (\psi_1 \mathbf{W} \psi_2)[Y]_\mu$ for an arbitrary fixed i. We prove $w_i \models \psi_1 \mathbf{W} \psi_2$:

$$
\begin{aligned}
& w_i \models (\psi_1 \mathbf{W} \psi_2)[Y]_\mu \\
\iff & w_i \models (\psi_1[Y]_\mu) \mathbf{U} (\psi_2[Y]_\mu) && (\varphi \notin Y, \text{ Definition 4.8}) \\
\iff & \exists j.\, w_{i+j} \models \psi_2[Y]_\mu \wedge \forall k < j.\, w_{i+k} \models \psi_1[Y]_\mu \\
\implies & \exists j.\, w_{i+j} \models \psi_2 \wedge \forall k < j.\, w_{i+k} \models \psi_1 && \text{(induction hypothesis)} \\
\iff & w_i \models \psi_1 \mathbf{U} \psi_2 \\
\implies & w_i \models \psi_1 \mathbf{W} \psi_2
\end{aligned}
$$

– Case $\varphi = \psi_1 \vee \psi_2$: We derive in a straightforward manner for an arbitrary and fixed i:

$$
\begin{aligned}
& w_i \models (\psi_1 \vee \psi_2)[Y]_\mu \\
\iff & w_i \models \psi_1[Y]_\mu \vee w_i \models \psi_2[Y]_\mu && \text{(Definition 4.8)} \\
\implies & w_i \models \psi_1 \vee w_i \models \psi_2 && \text{(induction hypothesis)} \\
\iff & w_i \models \psi_1 \vee \psi_2
\end{aligned}
$$

□

Proof of Lemma 4.20

Lemma 4.20. *Let φ be a formula, let w be a word. If $w \models \varphi[X]_\nu$, then $w_i \models af(\varphi, w_{0i})[X]_\nu$ for every index i.*

Proof. Assume $w \models \varphi[X]_\nu$. It suffices to prove $w_1 \models af(\varphi, w_{01})[X]_\nu$ for a single letter w_{01}, since the general case immediately follows by an induction on i. For the single letter case we proceed by structural induction on φ, and consider only some representative cases:

- Case $\varphi = a$. Since $w \models a[X]_\nu = a$, we have $a \in w_{01}$. So $af(a, w_{01})[X]_\nu = \mathbf{tt}[X]_\nu = \mathbf{tt}$, and thus $w_1 \models af(\varphi, w_{01})[X]_\nu$.
- Case $\varphi = \psi \mathbf{U} \chi$. Since $w \models \varphi[X]_\nu$, we have $\varphi[X]_\nu \neq \mathbf{ff}$, and so $\varphi \in X$. We derive:

$$
\begin{aligned}
& w \models \varphi[X]_\nu \\
\iff & w \models (\psi[X]_\nu) \mathbf{W} (\chi[X]_\nu) && (\varphi \in X, \text{ Definition 4.7}) \\
\iff & w \models \chi[X]_\nu \vee (\psi[X]_\nu \wedge \mathbf{X}((\psi[X]_\nu) \mathbf{W} (\chi[X]_\nu))) && (\text{LTL semantics}) \\
\iff & w \models \chi[X]_\nu \vee (\psi[X]_\nu \wedge \mathbf{X}((\psi \mathbf{U} \chi)[X]_\nu)) && (\varphi \in X, \text{ Definition 4.7}) \\
\implies & w_1 \models af(\chi, w_{01})[X]_\nu \vee (af(\psi, w_{01})[X]_\nu \wedge \varphi[X]_\nu) && (\text{induction hypothesis}) \\
\iff & w_1 \models (af(\chi, w_{01}) \vee (af(\psi, w_{01}) \wedge \varphi))[X]_\nu && (\text{Definition 4.7}) \\
\iff & w_1 \models af(\varphi, w_{01})[X]_\nu && (\text{Definition 3.1})
\end{aligned}
$$

□

Proof of Lemma 4.21

Lemma 4.21. *Let φ be a formula and let w be a word. If w is μ-stable with respect to φ and $w \models \varphi$, then there exits an index i such that $w_i \models af(\varphi, w_{0i})[\mathcal{GF}_w^\varphi \cap \Downarrow(\varphi)]_\nu$.*

Proof. We proceed by structural induction on φ. We consider three representative cases: first, $\psi_1 \vee \psi_2$ for the straight-forward cases ($\mathbf{tt}$, $\mathbf{ff}$, a, $\neg a$, $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $\mathbf{X}\psi$); second, $\psi_1 \mathbf{U} \psi_2$ for the least-fixed-point operators; third, $\psi_1 \mathbf{W} \psi_2$ for the greatest-fixed-point operators.

- Case $\varphi = \psi_1 \vee \psi_2$. Assume $w \models \psi_1 \vee \psi_2$ and μ-stability. We proceed by case distinction and assume $w \models \psi_1$. Since ψ_1 is a subformula of φ, w is also μ-stable with respect to ψ_1. By applying the induction hypothesis we obtain an index i such that $w_i \models af(\psi_1, w_{0i})[\mathcal{GF}_w^{\psi_1} \cap \Downarrow(\psi_1)]_\nu$. Thus $w_i \models af(\psi_1, w_{0i})[\mathcal{GF}_w^{\psi_1} \cap \Downarrow(\psi_1)]_\nu \vee af(\psi_2, w_{0i})[\mathcal{GF}_w^{\psi_1} \cap \Downarrow(\psi_1)]_\nu$, which is equivalent to $w_i \models af(\varphi, w_{0i})[\mathcal{GF}_w^{\psi_1} \cap \Downarrow(\psi_1)]_\nu$. Hence $w_i \models af(\varphi, w_{0i})[\mathcal{GF}_w^\varphi \cap \Downarrow(\varphi)]_\nu$. The other case is analogous.
- Case $\varphi = \psi_1 \mathbf{U} \psi_2$. Assume $w \models \psi_1 \mathbf{U} \psi_2$ and μ-stability. Thus there exists an index i such that $w_j \models \psi_1$ for all $j < i$ and $w_i \models \psi_2$. Further, w_j is μ-stable with respect to ψ_1 and w_i is μ-stable with respect to ψ_2. We apply the induction hypothesis for each j and i and get indices j' and i' such that the property holds. Let k be the maximum of all these (finitely many) indices. By Lemma 4.20 we then have $w_k \models af(\psi_1, w_{jk})[\mathcal{GF}_{w_j}^{\psi_1} \cap \Downarrow(\psi_1)]_\nu$ for all $j < i$ and $w_k \models af(\psi_2, w_{ik})[\mathcal{GF}_{w_i}^{\psi_2} \cap \Downarrow(\psi_2)]_\nu$. Normalising the sub- and superscripts yields: $w_k \models af(\psi_1, w_{jk})[\mathcal{GF}_w^\varphi \cap \Downarrow(\varphi)]_\nu$ for all $j < i$ and $w_k \models af(\psi_2, w_{ik})[\mathcal{GF}_w^\varphi \cap \Downarrow(\varphi)]_\nu$. Recombining this to $\varphi = \psi_1 \mathbf{U} \psi_2$ yields $w_{k+1} \models af(\psi_1 \mathbf{U} \psi_2, w_{0(k+1)})[\mathcal{GF}_w^\varphi \cap \Downarrow(\varphi)]_\nu$.

– Case $\varphi = \psi_1 \mathbf{W} \psi_2$. Assume $w \models \psi_1 \mathbf{W} \psi_2$ and μ-stability. In the case $w \models \psi_1 \mathbf{U} \psi_2$ the proof is analogous to the previous case. Hence assume $w \not\models \mathbf{F}\psi_2$. Thus we have $w \models \mathbf{G}\psi_1$. Let now j be an arbitrary index with $w_j \models \psi_1$. Applying Lemma 4.10 we get $w_j \models \psi_1[\mathcal{F}_{w_j}^{\psi_1}]_\nu$ and due to μ-stability we also have $w_j \models \psi_1[\mathcal{GF}_{w_j}^{\psi_1}]_\nu$, which we normalise to $w_j \models \psi_1[\mathcal{GF}_w^\varphi \cap \mu(\psi_1)]_\nu$. Since $\mu(\psi_1) \subseteq \Downarrow(\varphi)$, we rewrite this to $w_j \models \psi_1[\mathcal{GF}_w^\varphi \cap \Downarrow(\varphi)]_\nu$. Since we can do this to all indices j, we get $w \models (\psi_1 \mathbf{W} \psi_2)[\mathcal{GF}_w^\varphi \cap \Downarrow(\varphi)]_\nu$ and we are done.

□

5 DRA Constructions

We now turn our attention towards constructing deterministic Rabin automata (DRA) using the Master Theorem (Theorem 4.14). We proceed in the two steps outlined before: First, we show in Section 5.1 that the function *af* can be used to construct deterministic Büchi (DBA) and co-Büchi automata (DCA) for easy LTL fragments, namely μLTL, νLTL, $\mathbf{GF}(\mu LTL)$, and $\mathbf{FG}(\nu LTL)$. Second, using these constructions we build in Section 5.2 automata checking (1), (2), and (3) of Theorem 4.14 for fixed sets X and Y and then assemble these parts with Boolean operations to obtain the final DRA. An extended discussion of optimisations to the presented construction can be found in Chapter 7.

5.1 DRAs for μLTL, νLTL, $\mathbf{GF}(\mu LTL)$, and $\mathbf{FG}(\nu LTL)$

We construct DRAs for μLTL and νLTL formulas by using *af* to track the 'effect' of the word w on the formula φ and we obtain from Lemma 3.9 two simple criteria – $\exists i.\, af(\varphi, w_{0i}) \sim \mathbf{tt}$ and $\forall i.\, af(\varphi, w_{0i}) \nsim \mathbf{ff}$ – for deciding if the read word w is satisfying or not, thus should be accepted or not. For formulas from $\mathbf{GF}(\mu LTL)$ and $\mathbf{FG}(\nu LTL)$ we simply repeatedly check these criteria. Formally we have:

Proposition 5.1. *Let $\varphi \in \mu LTL$, and let $\sim$ be an af-congruence.*

- *The following DBA over the alphabet 2^{Ap} recognises $\mathcal{L}(\varphi)$:*

$$\mathcal{B}^{\varphi}_{\mu} = (Reach(\varphi)_{/\sim}, af_{\sim}, [\varphi]_{\sim}, inf([\mathbf{tt}]_{\sim}))$$

- *The following DCA over the alphabet 2^{Ap} recognises $\mathcal{L}(\varphi)$:*

$$\mathcal{C}^{\varphi}_{\mu} = (Reach(\varphi)_{/\sim}, af_{\sim}, [\varphi]_{\sim}, fin(\overline{\{[\mathbf{tt}]_{\sim}\}}))$$

- *The following DBA over the alphabet 2^{Ap} recognises $\mathcal{L}(\mathbf{GF}\varphi)$:*

$$\mathcal{B}^{\varphi}_{\mathbf{GF}\mu} = (Reach(\mathbf{F}\varphi)_{/\sim}, af^{\mathbf{F}\varphi}_{\sim}, [\mathbf{F}\varphi]_{\sim}, inf([\mathbf{tt}]_{\sim}))$$

$$af^{\mathbf{F}\varphi}_{\sim}([\psi]_{\sim}, \sigma) = \begin{cases} [\mathbf{F}\varphi]_{\sim} & \text{if } \psi \sim \mathbf{tt} \\ [af(\psi, \sigma)]_{\sim} & \text{otherwise.} \end{cases}$$

Let $\varphi \in \nu LTL$.

- *The following DCA over the alphabet 2^{Ap} recognises $\mathcal{L}(\varphi)$:*

$$\mathcal{C}^{\varphi}_{\nu} = (Reach(\varphi)_{/\sim}, af_{\sim}, [\varphi]_{\sim}, fin([\mathbf{ff}]_{\sim}))$$

- *The following DBA over the alphabet 2^{Ap} recognises $\mathcal{L}(\varphi)$:*

$$\mathcal{B}^{\varphi}_{\nu} = (Reach(\varphi)_{/\sim}, af_{\sim}, [\varphi]_{\sim}, inf(\overline{\{[\mathbf{ff}]_{\sim}\}}))$$

– *The following DCA over the alphabet 2^{Ap} recognises $\mathcal{L}(\mathbf{FG}\varphi)$:*

$$\mathcal{C}^{\varphi}_{\mathbf{FG}\nu} = (\mathit{Reach}(\mathbf{G}\varphi)_{/\sim}, \mathit{af}^{\mathbf{G}\varphi}_{\sim}, [\mathbf{G}\varphi]_{\sim}, \mathit{fin}([\mathbf{ff}]_{\sim}))$$

$$\mathit{af}^{\mathbf{G}\varphi}_{\sim}([\psi]_{\sim}, \sigma) = \begin{cases} [\mathbf{G}\varphi]_{\sim} & \text{if } \psi \sim \mathbf{ff} \\ [\mathit{af}(\psi, \sigma)]_{\sim} & \text{otherwise.} \end{cases}$$

First, note that instead of $\mathit{inf}(\{q\})$ we write just $\mathit{inf}(q)$ to make the notation more succinct. Second, observe that $\mathcal{B}^{\varphi}_{\mu}$ and $\mathcal{C}^{\varphi}_{\mu}$, as well as $\mathcal{B}^{\varphi}_{\nu}$ and $\mathcal{C}^{\varphi}_{\nu}$ share the same structure and only differ in the definition of the acceptance condition. $\mathcal{C}^{\varphi}_{\mu}$ can be understood as first applying the classic complementation procedure from deterministic finite automata, which simply swaps final and non-final states, on $\mathcal{B}^{\varphi}_{\mu}$ to obtain a Büchi automaton recognising $\neg\varphi$ and then flipping the acceptance condition to finally arrive at the co-Büchi condition recognising $\neg\neg\varphi \sim_l \varphi$. This works, because in fact the automaton $\mathcal{B}^{\varphi}_{\mu}$ is *weak*, meaning that the states of every strongly connected component (SCC) are either all accepting or all rejecting. The same observation also holds for $\mathcal{B}^{\varphi}_{\nu}$ and $\mathcal{C}^{\varphi}_{\nu}$. Before proving correctness of these constructions we look at several examples:

Example 5.2. *Let $\varphi = a \wedge \mathbf{X}(b \vee \mathbf{F}c) \in \mu LTL$. The DBA $\mathcal{B}^{\varphi}_{\mathbf{GF}\mu}$ constructed from Proposition 5.1 using $\sim_p$ as the underlying equivalence relation recognising $\mathcal{L}(\mathbf{GF}\varphi)$ is depicted in Figure 5.1. Notice that from now on we label the states, which are in-fact equivalence classes, by a representative of the respective class.*

Example 5.3. *Let $\varphi = a\mathbf{W}b \vee c \in \nu LTL$. The DCA $\mathcal{C}^{\varphi}_{\mathbf{FG}\nu}$ constructed from Proposition 5.1 using $\sim_l$ as the underlying equivalence relation recognising $\mathcal{L}(\mathbf{FG}\varphi)$ is depicted in Figure 5.2.*

Proof of Proposition 5.1. Let φ be a formula of μLTL, and let $\sim$ be an *af*-congruence. We want to prove (1) $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{B}^{\varphi}_{\mu})$, (2) $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{C}^{\varphi}_{\mu})$ and (3) $\mathcal{L}(\mathbf{GF}\varphi) = \mathcal{L}(\mathcal{B}^{\varphi}_{\mathbf{GF}\mu})$. Let w now be an arbitrary word.

($\Rightarrow_1$) Assume $w \models \varphi$. Using Lemma 3.9 we can find an index i such that $\mathit{af}(\varphi, w_{0i}) \sim \mathbf{tt}$. Hence the run r with $r[i] = [\mathit{af}(\varphi, w_{0i})]_{\sim}$ starting in the initial state $[\varphi]_{\sim}$ will reach $[\mathbf{tt}]_{\sim}$ after reading w_{0i}. After reaching $[\mathbf{tt}]_{\sim}$ the run r loops in that state and thus r is accepting and the word w is in $\mathcal{L}(\mathcal{B}^{\varphi}_{\mu})$.

($\Leftarrow_1$) Assume w is accepted by $\mathcal{B}^{\varphi}_{\mu}$. Then there exists an accepting run r for w and this run visits the state $[\mathbf{tt}]_{\sim}$ infinitely often. Let i be the index such that $[\mathbf{tt}]_{\sim} = \mathit{af}_{\sim}([\varphi]_{\sim}, w_{0i}) = [\mathit{af}(\varphi, w_{0i})]_{\sim}$. Thus applying Lemma 3.9 to $\mathit{af}(\varphi, w_{0i}) \sim \mathbf{tt}$ yields $w \models \varphi$ and we are done.

(2) Observe that $\mathcal{B}^{\varphi}_{\mu}$ and $\mathcal{C}^{\varphi}_{\mu}$ have exactly the same structure apart from the acceptance condition. They are both weak and deterministic and thus all words accepted by $\mathcal{B}^{\varphi}_{\mu}$ are also accepted by $\mathcal{C}^{\varphi}_{\mu}$ and vice-versa. Thus $\mathcal{L}(\mathcal{B}^{\varphi}_{\mu}) = \mathcal{L}(\mathcal{C}^{\varphi}_{\mu})$ and (2) is an intermediate consequence of (1).

($\Rightarrow_3$) Assume $w \models \mathbf{GF}\varphi$. This is equivalent to $\forall i.\, w_i \models \mathbf{F}\varphi$ and with Lemma 3.9 we get $\forall i.\, \exists j.\, \mathit{af}(\mathbf{F}\varphi, w_{ij}) \sim \mathbf{tt}$. Since $\mathcal{B}^{\varphi}_{\mathbf{GF}\mu}$ is deterministic, there exists exactly one run r for the word w. We now show that this run r is accepting. The run starts in $r[0] = [\mathbf{F}\varphi]_{\sim}$ and let $j > i$ be the smallest index for $i = 0$, such that $\mathit{af}(\mathbf{F}\varphi, w_{0j}) \sim \mathbf{tt}$. Observe that this is the first point where $\mathit{af}_{\sim}$ and $\mathit{af}^{\mathbf{F}\varphi}_{\sim}$ diverge and the later one resets to $[\mathbf{F}\varphi]_{\sim}$. Thus after visiting the accepting state $[\mathbf{tt}]_{\sim}$ the run r returns to the initial state $[\mathbf{F}\varphi]_{\sim}$ and

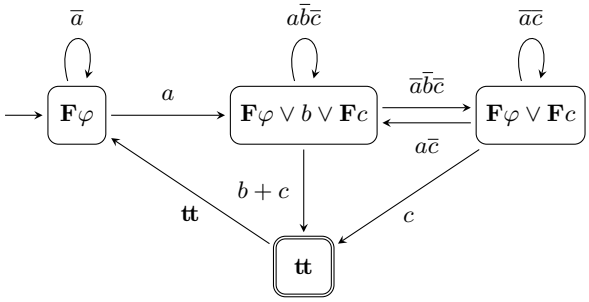


Figure 5.1: DBA $\mathcal{B}^{\varphi}_{\mathbf{GF}\mu}$ for $\varphi = a \wedge \mathbf{X}(b \vee \mathbf{F}c)$ using $\sim_p$.

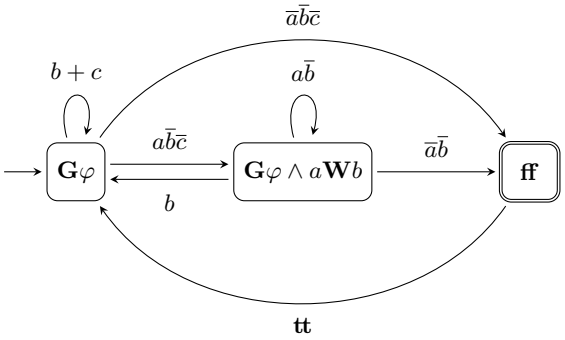


Figure 5.2: DCA $\mathcal{C}^{\varphi}_{\mathbf{FG}\nu}$ for $\varphi = a\mathbf{W}b \vee c$ using $\sim_l$.

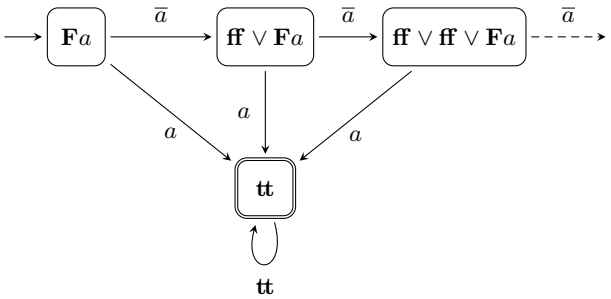


Figure 5.3: Infinite DBA $\mathcal{B}^{\varphi}_{\mu}$ for $\varphi = \mathbf{F}a$ using $\sim_c$.

we can apply the same argument starting at $r[j+1] = [\mathbf{F}\varphi]_\sim$. Repeating this argument again and again we conclude that the run r is accepting and thus the word w is contained in $\mathcal{L}(\mathcal{B}^\varphi_{\mathbf{GF}\mu})$.

($\Leftarrow_3$) Assume w is accepted by $\mathcal{B}^\varphi_{\mathbf{GF}\mu}$. Then there exists an accepting run r for w and this run visits the state $[\mathbf{tt}]_\sim$ infinitely often. Provided we have shown $\forall i.\, \exists j \geq i.\, \exists k.\, \mathit{af}(\mathbf{F}\varphi, w_{jk}) \sim \mathbf{tt}$, we apply Lemma 3.9 to arrive at $w \models \mathbf{GF}\varphi$ which we want to show. We now focus on the remaining goal $\forall i.\, \exists j \geq i.\, \exists k.\, \mathit{af}(\mathbf{F}\varphi, w_{jk}) \sim \mathbf{tt}$: Let i be an arbitrary index. Since r is an accepting run, there exist infinitely many j's, such that $r[j-1] = [\mathbf{tt}]_\sim$ and $r[j] = [\mathbf{F}\varphi]_\sim$. Let j be such an index with $j > i$. Since r is an accepting run, there also exists some $k > j$ such that $r[k] = [\mathbf{tt}]_\sim$ and the run r has not visited $[\mathbf{tt}]_\sim$ in-between. Thus we follow $\mathit{af}(\varphi, w_{jk}) \sim \mathbf{tt}$ and fill the remaining gap.

We now move on to the second part concerned with νLTL. Let φ be a formula of νLTL and let $\sim$ still be an *af*-congruence. We want to prove (4) $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{C}^\varphi_\nu)$, (5) $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{B}^\varphi_\nu)$ and (6) $\mathcal{L}(\mathbf{FG}\varphi) = \mathcal{L}(\mathcal{C}^\varphi_{\mathbf{FG}\nu})$. The proof of (4) and (5) is analogous to the proof of (1) and (2) and thus we are skipping it. Observe that $\mathcal{C}^\varphi_{\mathbf{FG}\nu}$ mirrors the structure of $\mathcal{B}^\varphi_{\mathbf{GF}\mu}$ and the proof of (6) is also analogous to the one of (3). □

Complexity Analysis. The size is of the automata from Proposition 5.1 crucially depends on the choice of $\sim$ because it determines how $\mathit{Reach}(\varphi)$ is partitioned into equivalence classes. The spectrum ranges here from an optimal choice (doubly exponential) up-to infinity: On the one hand with $\sim_c$ we obtain a possibly infinite structures for which an example can be seen in Figure 5.3. On the other hand the equivalence relation $\sim_l$ will produce the smallest automata that are still doubly exponential in the worst-case [KV98; AL04]. But this does not come for free, since deciding $\sim_l$ is relatively expensive operation. In our analysis we focus on the equivalence relation $\sim_p$ that strikes a good balance between the cost of deciding $\sim_p$ and the worst-case bounds.

In Lemma 3.8 we already determined the upper bound $2^{2^{|\mathit{sf}(\varphi)|}}$ for the cardinality of $\mathit{Reach}(\varphi)_{/\sim_p}$. Assume φ has length n. Then the DBAs and DCAs $\mathcal{B}^\varphi_\mu$, $\mathcal{B}^\varphi_\nu$, $\mathcal{C}^\varphi_\mu$, and $\mathcal{C}^\varphi_\nu$ have at most 2^{2^n} states. Further $\mathcal{B}^\varphi_{\mathbf{GF}\mu}$ and $\mathcal{C}^\varphi_{\mathbf{FG}\nu}$ have at most $2^{2^{n+1}}$ by the same argument.

5.2 DRAs for Arbitrary LTL Formulas

Let φ be a formula of length n. We use Theorem 4.14 (restated below for reference) to construct a DRA for $\mathcal{L}(\varphi)$ with $2^{2^{O(n)}}$ states and at most 2^n Rabin pairs. Since in this chapter our goal is only to show that we can easily obtain automata of asymptotically optimal size, we give priority to a simpler construction over one with fewer states. We comment in Chapter 7 on optimisations that reduce the size of the automata.

Theorem 4.14 (Master Theorem)**.** *Let φ be a formula and let w be a word. Then $w \models \varphi$ if and only if there exist $X \subseteq \mu(\varphi)$ and $Y \subseteq \nu(\varphi)$ satisfying:*

1. $\exists i.\, w_i \models \mathit{af}(\varphi, w_{0i})[X]_\nu$
2. $\forall \psi \in X.\, w \models \mathbf{GF}(\psi[Y]_\mu)$
3. $\forall \psi \in Y.\, w \models \mathbf{FG}(\psi[X]_\nu)$

For a fixed guess X and Y we first construct DBAs and DCAs checking (1-3), to which we will refer by $\mathcal{C}^1_{\varphi,X}$, $\mathcal{B}^2_{X,Y}$, and $\mathcal{C}^3_{X,Y}$. These automata are then intersected resulting in

a DRA $\mathcal{R}_{\varphi,X,Y}$ with $2^{2^{O(n)}}$ states and one single Rabin pair. We then construct a DRA $\mathcal{A}_{\mathsf{DRA}}(\varphi)$ for $\mathcal{L}(\varphi)$ by taking the union of all $\mathcal{R}_{\varphi,X,Y}$. Proposition 5.1 provides us with the tools to build $\mathcal{B}^2_{X,Y}$ and $\mathcal{C}^3_{X,Y}$, but cannot be immediately applied to $\mathcal{C}^1_{\varphi,X}$. Thus we need to delay the definition of $\mathcal{A}_{\mathsf{DRA}}(\varphi)$ in order to figure out how to construct an automaton for condition (1).

Interlude: An Automaton for Condition (1)

We need to define an automaton that accepts a word w if and only if $w_i \models \mathit{af}(\varphi, w_{0i})[X]_\nu$ for some suffix w_i. It is intuitive that if we have an automaton identifying such an index i we immediately obtain that (1) of Theorem 4.14 holds. However, the other direction is not immediately clear, since there is no nondeterminism for guessing a suitable i. We address this by resorting to the previously seen Lemma 4.20, which allows us to postpone checking $w_i \models \mathit{af}(\varphi, w_{0i})[X]_\nu$ by an arbitrary number of steps and thus the automaton cannot miss the right moment.

Lemma 4.20. *Let φ be a formula, let w be a word. If $w \models \varphi[X]_\nu$, then $w_i \models \mathit{af}(\varphi, w_{0i})[X]_\nu$ for every index i.*

Loosely speaking, $\mathcal{C}^1_{\varphi,X}$ – the automaton responsible for checking (1) – starts by checking $w \models \varphi[X]_\nu$. For this it uses the construction of Proposition 5.1 on the νLTL formula $\varphi[X]_\nu$. Intuitively, if that automaton rejects, then it rejects 'after finite time'. If the formula becomes equivalent to $\mathbf{ff}$ after, say, i steps, then $w \not\models \varphi[X]_\nu$, and $\mathcal{C}^1_{\varphi,X}$ proceeds to check $w \models \mathit{af}(\varphi, w_{0i})[X]_\nu$. In order to perform this reset, $\mathcal{C}^1_{\varphi,X}$ needs to know $\mathit{af}(\varphi, w_{0i})$, and so it also maintains $\mathit{af}(\varphi, w_{0i})$ in its state. In other words, after j steps $\mathcal{C}^1_{\varphi,X}$ is in state:

$$\langle \mathit{af}(\varphi, w_{0j})\,, \mathit{af}((\mathit{af}(\varphi, w_{0i})\,[X]_\nu), w_{ij})\rangle$$

where $i \leq j$ is the number of steps after which $\mathcal{C}^1_{\varphi,X}$ had to reset for the last time. If the second component of the state becomes $\mathbf{ff}$, then the automaton uses the first component to determine which formula to check next. The accepting condition then states that the transitions leading to a state of the form $\langle \psi, \mathbf{ff}\rangle$ must occur finitely often, which implies that eventually one of the checks $w \models \varphi_j[X]_\nu$ succeeds.

We formally define this idea in two steps: first, we integrate $\cdot[\cdot]_\nu$ into our use of *af*-congruences and coin the term $\cdot[\cdot]_\nu$-congruence; second, we base the formal definition and the correctness proof of $\mathcal{C}^1_{\varphi,X}$ on this definition.

Definition 5.4. *Let $\sim$ be an equivalence relation. We call $\sim$ an* $\cdot[\cdot]_\nu$-congruence *if the following holds for all formulas φ and ψ:*

$$\varphi \sim \psi \implies \varphi[X]_\nu \sim \psi[X]_\nu$$

Looking at our three equivalence relations $\sim_c$, $\sim_p$, and $\sim_l$ we realise that only two of them are actually $\cdot[\cdot]_\nu$-congruences and $\sim_l$ is unfortunately ruled out for the construction of $\mathcal{C}^1_{\varphi,X}$. Observe that $\sim_l$ is still suitable for the construction of the other components.

Lemma 5.5. *$\sim_c$ and $\sim_p$ are $\cdot[\cdot]_\nu$-congruences. $\sim_l$ is not an $\cdot[\cdot]_\nu$-congruence.*[1]

[1] In fact the definition of $\cdot[\cdot]_\nu$-congruence can be relaxed a bit as seen in Section 7.4.

Proof sketch. ($\sim_c$, $\sim_p$) Notice that $\cdot[\cdot]_\nu$ is in-fact a substitution only replacing proper subformulas. Thus constants ($\mathbf{tt}$, $\mathbf{ff}$) and the Boolean connectives ($\wedge$, $\vee$) are not replaced in the syntax tree. Observe that $\sim_c$ and $\sim_p$ are congruent on such substitutions and, since $\cdot[\cdot]_\nu$ is a special case, we are done.

($\sim_l$) The following counterexample shows that $\sim_l$ cannot be an $\cdot[\cdot]_\nu$-congruence. Let $\varphi = \mathbf{FG}a$ and $\psi = \mathbf{FG}a \vee \mathbf{G}a$ be two formulas. Then we have $\varphi \sim_l \psi$, but:

$$\varphi[\emptyset]_\nu = \mathbf{ff} \not\sim_l \mathbf{G}a \sim_l \mathbf{ff} \vee \mathbf{G}a = \psi[\emptyset]_\nu$$

□

Proposition 5.6. *Let $\sim$ be an af- and $\cdot[\cdot]_\nu$-congruence, let φ be a formula, and let X be a set of formulas. Then the following DCA over the alphabet 2^{Ap} recognises exactly the words w such that $\exists i.\, w_i \models af(\varphi, w_{0i})[X]_\nu$ holds:*

$$\mathcal{C}^1_{\varphi,X} = (Q, \delta, \langle [\varphi]_\sim, [\varphi[X]_\nu]_\sim \rangle, fin(\alpha))$$

where we define the states Q, the transition function δ, and the rejecting states α in the following way:

- *$Q = Reach(\varphi)_{/\sim} \times \bigcup_{\psi \in Reach(\varphi)} Reach(\psi[X]_\nu)_{/\sim}$. That is, a state is a tuple of equivalence classes $[\cdot]_\sim$, where the second tracks the formula after applying $\cdot[\cdot]_\nu$.*
- $$\delta(\langle [\xi]_\sim, [\zeta]_\sim \rangle, \sigma) = \begin{cases} \langle [af(\xi,\sigma)]_\sim, [af(\xi,\sigma)[X]_\nu]_\sim \rangle & \text{if } \zeta \sim \mathbf{ff} \\ \langle [af(\xi,\sigma)]_\sim, [af(\zeta,\sigma)]_\sim \rangle & \text{otherwise.} \end{cases}$$

  That is, a transition either resets a failed attempt to prove $w_i \models af(\varphi, w_{0i})[X]_\nu$ for some i or continues unfolding both equivalence classes using $af_\sim$.
- *$\alpha = Reach(\varphi)_{/\sim} \times \{[\mathbf{ff}]_\sim\}$.*

The reader might have noticed that the $\cdot[\cdot]_\nu$-congruence property is actually only necessary for the first component, since we apply it only to states from that component and the second component could use a different *af*-congruence. We will see in Section 7.5.1 how to adapt this insight into a more involved, but potentially more succinct construction.

Example 5.7. *Let $\varphi = \mathbf{G}(a\mathbf{U}b \vee \mathbf{F}c)$ be a formula and let $X = \{a\mathbf{U}b\}$ be a guess. Hence $\varphi[X]_\nu = \mathbf{G}(a\mathbf{W}b \vee \mathbf{ff})$. The DCA $\mathcal{C}^1_{\varphi,X}$ from Proposition 5.6 for φ is shown in Figure 5.4 and for this particular example we use the special af- and $\cdot[\cdot]_\nu$-congruence $\sim_q$ based on $\sim_p$ that is be introduced in Section 7.4. For the time being the reader can think of $\sim_q$ as $\sim_l$, but with the important difference that $\sim_q$ is also an $\cdot[\cdot]_\nu$-congruence. In the context of our example we thus can assume that $\varphi[X]_\nu \sim_q \varphi[X]_\nu \wedge a\mathbf{W}b$ and $\varphi \sim_q \varphi \wedge (a\mathbf{U}b \vee \mathbf{F}c)$ hold.*

Let us now consider two words w and w'. For $w = (\overline{a}\overline{b}c)((a\overline{b}\overline{c})(\overline{a}b\overline{c}))^\omega$ we have $X = \mathcal{GF}^\varphi_w$. On the one hand the word w is accepted, since the run eventually loops in the initial state. On the other hand, we have $w_1 \models af(\varphi, w_{01})[X]_\nu \sim_l \mathbf{G}(a\mathbf{W}b)$. For $w' = (\overline{a}\overline{b}c)^\omega$ we have $X \neq \mathcal{GF}^\varphi_{w'}$. The word w' is rejected, since the run alternates between the initial state and the rejecting state $\langle [\varphi]_\sim, [\mathbf{ff}]_\sim \rangle$, and we also have $w'_i \not\models af(\varphi, w_{0i})[X]_\nu \sim_l \mathbf{G}(a\mathbf{W}b)$ for all suffixes w'_i.

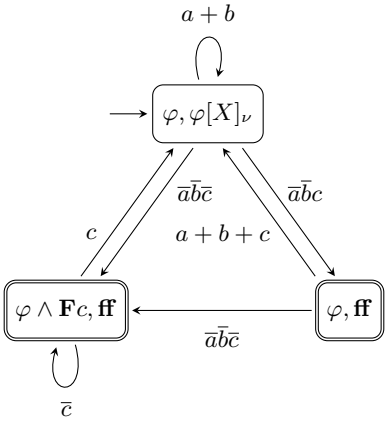


Figure 5.4: DCA $\mathcal{C}^1_{\varphi,X}$ for $\varphi = \mathbf{G}(a\mathbf{U}b \vee \mathbf{F}c)$ and $X = \{a\mathbf{U}b\}$ using the yet-to-be-defined $\sim_q$.

Proof of Proposition 5.6. ($\Rightarrow$) Let w be a word such that $w_i \models \mathit{af}(\varphi, w_{0i})[X]_\nu$ for some i. We need to show that the run r induced by w is an accepting run on $\mathcal{C}^1_{\varphi,X}$. Then by Lemma 4.20 we now that all following $j > i$ have the same property ($w_j \models \mathit{af}(\varphi, w_{0j})[X]_\nu$). Let r be the run on $\mathcal{C}^1_{\varphi,X}$ corresponding to w. We prove that there is at most one $k > i$, such that $r[k] \in \alpha$ and thus $w \in \mathcal{L}(\mathcal{C}^1_{\varphi,X})$. Assume k_1 is the first index larger then i where $r[k_1] \in \alpha$. If there is no such element we are immediately done. Thus $r[k_1 + 1] = \langle \mathit{af}(\varphi, w_{0(k_1+2)}), (\mathit{af}(\varphi, w_{0(k_1+2)}))[X]_\nu \rangle$. Since $k_1 + 2 > i$ we have $w_{k_1+2} \models \mathit{af}(\varphi, w_{0(k_1+2)})[X]_\nu$, thus due Lemma 3.9 we have that the second component never reaches $\mathbf{ff}$ again.

($\Leftarrow$) Assume w is accepted by $\mathcal{C}^1_{\varphi,X}$. Then the corresponding run r visits the set of state α only finitely often. Let i be the time last time were the run r encounters $[\mathbf{ff}]_\sim$ in the second component or -1 if there is never one. Then the second component is $\mathit{af}(\varphi, w_{0(i+1)})[X]_\nu$ at $r[i+1]$. Since this component never sees $[\mathbf{ff}]_\sim$ again after i, we can apply Lemma 3.9 to obtain $w_{i+1} \models \mathit{af}(\varphi, w_{0(i+1)})[X]_\nu$, which concludes the proof. □

Complexity Analysis. For the analysis we focus only on $\sim_p$ as the underlying congruence. As seen before $\mathit{Reach}(\varphi)_{/\sim_p}$ has at most size 2^{2^n} where n is the length of the formula. Interestingly $\bigcup_{\psi \in \mathit{Reach}(\varphi)} \mathit{Reach}(\psi[X]_\nu)_{/\sim}$ can also be bounded by 2^{2^n}. This set is a subset of all (positive) Boolean functions over at most n variables, since $\cdot[\cdot]_\nu$ only replaces existing proper subformulas by other subformulas of the same (or smaller) size or the constant $\mathbf{ff}$. Thus the total number of states of $\mathcal{C}_{\varphi,X}$ is bounded by $\left(2^{2^n}\right)^2 = 2^{2^{n+1}}$.

The Complete DRA

Finding a way to check (1) was the last missing piece and we can now assemble DRAs for arbitrary formulas by means of union and intersection applied to automata from Proposition 5.1 and Proposition 5.6:

Theorem 5.8. *Let φ be a formula and let $\sim$ be an af- and $\cdot[\cdot]_\nu$-congruence. We define for each $X \subseteq \mu(\varphi)$ and each $Y \subseteq \nu(\varphi)$ the following, from the equivalence relation $\sim$ constructed DBAs and DCAs:*

– $\mathcal{C}^1_{\varphi,X}$ *is the DCA from Proposition 5.6.*

– $\mathcal{B}^2_{X,Y} = \bigcap_{\psi \in X} \mathcal{B}^{\psi[Y]_\mu}_{\mathbf{GF}_\mu}$ *is the intersection of DBAs from Proposition 5.1.*

– $\mathcal{C}^3_{X,Y} = \bigcap_{\psi \in Y} \mathcal{C}^{\psi[X]_\nu}_{\mathbf{FG}_\nu}$ *is the intersection of DCAs from Proposition 5.1.*

Let $\mathcal{R}_{\varphi,X,Y}$ be the intersection DRA of the DBAs and DCAs $\mathcal{C}^1_{\varphi,X}$, $\mathcal{B}^2_{X,Y}$, and $\mathcal{C}^3_{X,Y}$ with exactly one Rabin pair:

$$\mathcal{R}_{\varphi,X,Y} = \mathcal{C}^1_{\varphi,X} \cap \mathcal{B}^2_{X,Y} \cap \mathcal{C}^3_{X,Y}$$

Then the following DRA over the alphabet 2^{Ap} recognises $\mathcal{L}(\varphi)$:

$$\mathcal{A}_{\mathsf{DRA}}(\varphi) = \bigcup_{\substack{X \subseteq \mu(\varphi) \\ Y \subseteq \nu(\varphi)}} \mathcal{R}_{\varphi,X,Y}$$

Example 5.9. *Let $\varphi = \mathbf{G}(a\mathbf{U}b \vee \mathbf{F}c)$ be the formula from Example 5.7. We now build the DRA $\mathcal{R}_{\varphi,X,Y}$ (Theorem 5.8) for the guess $X = \{a\mathbf{U}b\}$ and $Y = \emptyset$. The DRA $\mathcal{R}_{\varphi,X,Y}$ is the intersection of the DCA $\mathcal{C}^1_{\varphi,X}$ (Figure 5.4) and the DBA $\mathcal{B}^{(a\mathbf{U}b)}_{\mathbf{GF}_\mu}$ (Proposition 5.1). For this particular example we use again the special af- and $\cdot[\cdot]_\nu$-congruence $\sim_q$ mentioned in Example 5.7.*

Let us again consider the two words w and w'. For $w = (\overline{a}\overline{b}c)((a\overline{b}\overline{c})(\overline{a}b\overline{c}))^\omega$ we have $X = \mathcal{GF}^\varphi_w$. On the one hand the word w is accepted, since the run eventually alternates between the initial state and the accepting state (blue-shaded †, on the right-hand side of the initial state). On the other hand, we have $w \models \mathbf{G}(a\mathbf{U}b \vee \mathbf{F}c)$. For $w' = (\overline{a}\overline{b}c)^\omega$ we have $X \neq \mathcal{GF}^\varphi_{w'}$. The word w' is rejected, since the run alternates between the initial state and the red-shaded ⋆, rejecting states at the bottom of the figure. Further, the run never visits the accepting state on the right of the initial state showing $X \neq \mathcal{GF}^\varphi_{w'}$. Observe that in this case we still have $w' \models \mathbf{G}(a\mathbf{U}b \vee \mathbf{F}c)$ and the word is accepted by another DRA component. More precisely by the DRA $\mathcal{R}_{\varphi,X',Y'}$ with $X' = \{\mathbf{F}c\}$ and $Y' = \emptyset$.

Proof of Theorem 5.8. Let φ be a formula, let $\mathcal{A}_{\mathsf{DRA}}(\varphi)$ be the corresponding automaton constructed from a suitable equivalence relation $\sim$, and let w be a word.

($\Rightarrow$) Assume w is in $\mathcal{L}(\varphi)$. By Theorem 4.14 there exist X and Y such that (1-3) hold. We show that $w \in \mathcal{L}(\mathcal{A}_{\mathsf{DRA}}(\varphi))$ by proving that the run r for w on $\mathcal{R}_{\varphi,X,Y}$ is accepting. This run is accepting if there exist accepting runs for w on $\mathcal{C}^1_{\varphi,X}$, $\mathcal{B}^2_{X,Y}$, and $\mathcal{C}^3_{X,Y}$. In order to show this we simply need to apply Proposition 5.6 to (1) and Proposition 5.1 to (2-3) and we are done.

($\Leftarrow$) Assume w is accepted by $\mathcal{A}_{\mathsf{DRA}}(\varphi)$. Thus there exists an accepting run r for w on $\mathcal{R}_{\varphi,X,Y}$ for some $X \subseteq \mu(\varphi)$ and $Y \subseteq \nu(\varphi)$. Thus $\mathcal{C}^1_{\varphi,X}$, $\mathcal{B}^2_{X,Y}$, and $\mathcal{C}^3_{X,Y}$ accept w and applying Proposition 5.6 and Proposition 5.1 yields (1-3) for X and Y. Hence the right-hand side of Theorem 4.14 is fulfilled and so w is in $\mathcal{L}(\varphi)$. □

Complexity Analysis. For the analysis we focus again on $\sim_p$ as the underlying congruence. Let φ be a formula of length n. We have seen before that $\mathcal{C}^1_{\varphi,X}$ has at most $2^{2^{n+1}}$ states. By Proposition 5.1, $\mathcal{L}(\mathbf{GF}(\psi[Y]_\mu)$ is recognised by a DBA with at most $2^{2^{n+1}}$ states. Recall that the intersection of the languages of k DBAs with $s_1, \ldots, s_k$ states is

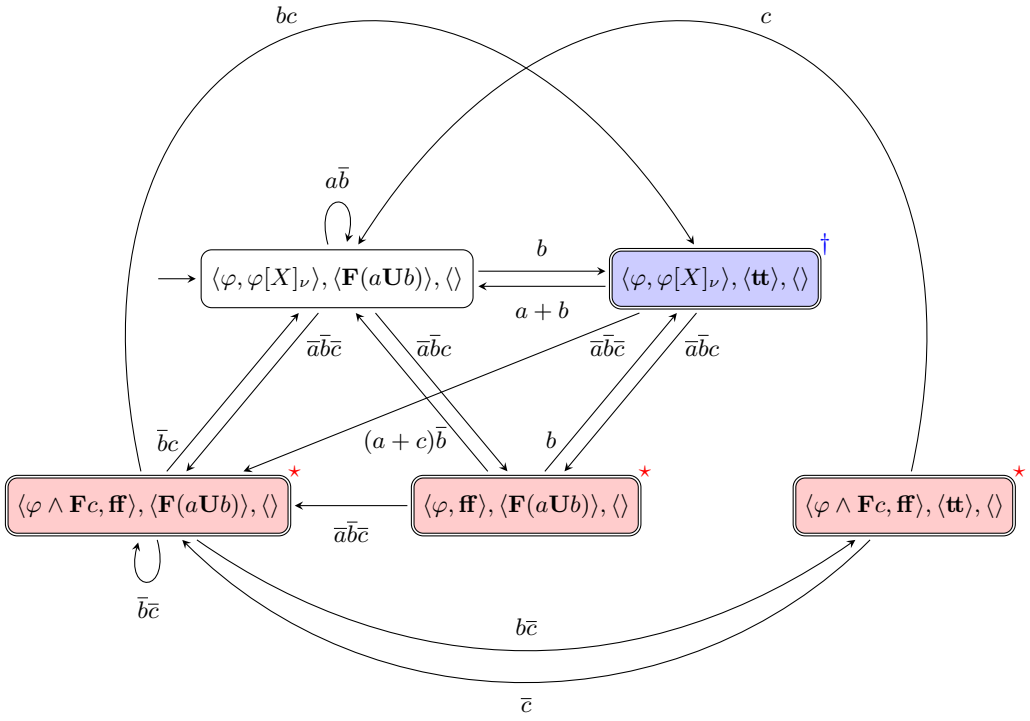


Figure 5.5: DRA $\mathcal{R}_{\varphi,X,Y}$ for $\varphi = \mathbf{G}(a\mathbf{U}b \vee \mathbf{F}c)$, $X = \{a\mathbf{U}b\}$, and $Y = \emptyset$ using the yet-to-be-defined $\sim_q$. The DRA $\mathcal{R}_{\varphi,X,Y}$ has only one Rabin pair: $\mathit{fin}(\star) \wedge \mathit{inf}(\dagger)$.

recognised by a DBA with $k \cdot \prod_{j=1}^{k} s_j$ states. Since $|X| \leq n$, the intersection of the DBAs $\mathcal{B}^2_{X,Y}$ for the formulas $\mathbf{GF}(\psi[Y]_\mu)$ yields a DBA with at most

$$n \cdot \left(2^{2^{n+1}}\right)^n = 2^{n2^{n+1}+(\log_2 n)} \leq 2^{2^{n+(\log_2 n)+2}}$$

states. Dually by Proposition 5.1, $\mathcal{L}(\mathbf{FG}(\psi[X]_\mu)$ is recognised by a DCA with at most $2^{2^{n+1}}$ states. Recall that the intersection of the languages of k DCAs with $s_1, \ldots, s_k$ states is recognised by a DBA with $\prod_{j=1}^{k} s_j$ states. Since $|Y| \leq n$, the intersection of the DCAs $\mathcal{C}^3_{X,Y}$ for the formulas $\mathbf{FG}(\psi[X]_\nu)$ yields a DCA with at most

$$\left(2^{2^{n+1}}\right)^n = 2^{n2^{n+1}} = 2^{2^{n+(\log_2 n)+1}}$$

states. Thus the intersection DRA $\mathcal{R}_{\varphi,X,Y}$ has at most

$$2^{2^{n+1}} \cdot 2^{2^{n+(\log_2 n)+2}} \cdot 2^{2^{n+(\log_2 n)+1}} \leq 2^{2^{n+(\log_2 n)+4}} \in 2^{2^{O(n)}}$$

states and one Rabin pair. Taking the union of all these yields a DRA $\mathcal{A}_{\mathsf{DRA}}(\varphi)$ with at most

$$\left(2^{2^{n+(\log_2 n)+4}}\right)^{2^n} = 2^{2^n \cdot 2^{n+(\log_2 n)+4}} = 2^{2^{2n+(\log_2 n)+4}} \in 2^{2^{O(n)}}$$

states and at most 2^n Rabin pairs. Observe that this matches asymptotically the known double exponential lower bound [KV98; AL04].

6 NBA and LDBA Constructions

In this chapter we proceed in the same manner as before: we first define constructions for LTL fragments and then assemble a translation for arbitrary LTL formulas from these components. However, we postpone this familiar program and begin with a construction for a special subclass of nondeterministic Büchi automata (NBA), the limit-deterministic Büchi automaton (LDBA). LDBAs can be partitioned into an initial component, which might contain states with a nondeterministic transition relation, and an accepting component that contains all accepting states (or transitions) and is deterministic. If a run enters the accepting component via a so-called 'jump', it cannot leave this component anymore. We begin with the LDBA construction, since it has the same structure of the final NBA construction[1], but is built from the known constructions from Chapter 5. Then we adapt the function *af* to the nondeterministic setting and return to the roadmap of the previous chapter: the translation to NBAs with (asymptotically) optimal size for fragments and arbitrary LTL formulas based on the Master Theorem.

6.1 LDBAs for Arbitrary LTL Formulas

The primary challenge for building LDBAs with the Master Theorem is that in the accepting component it is impossible to construct DBAs for some of the formulas required by Theorem 4.14, such as $\mathbf{FG}a$. However, we can make use of the available nondeterminism to 'guess' when $\mathbf{G}a$ holds and use the DBA construction for νLTL. Using this initial idea we rephrase Theorem 4.14 and break the symmetry of the Master Theorem in order to reduce 'guessing' to exactly one point i in the run. Specifically, we 'synchronise' checking (1) and (3), which results in the following corollary:

Corollary 6.1. *(Variant of the Master Theorem) Let φ be a formula and let w be a word. Then $w \models \varphi$ if and only if there exist $X \subseteq \mu(\varphi)$, $Y \subseteq \nu(\varphi)$, and $i \geq 0$ satisfying:*

$$\begin{aligned} &1'.\ w_i \models \mathit{af}(\varphi, w_{0i})[X]_\nu \\ &2'.\ \forall \psi \in X.\ w_i \models \mathbf{GF}(\psi[Y]_\mu) \\ &3'.\ \forall \psi \in Y.\ w_i \models \mathbf{G}(\psi[X]_\nu) \end{aligned}$$

Proof. Clearly, the existence of an index i satisfying (1'-3') implies that conditions (1-3) of Theorem 4.14 hold. For the other direction, assume conditions (1-3) hold. By Lemma 4.20 the index i stemming from condition (1) can be chosen arbitrarily large. Furthermore, since $w \models \bigwedge_{\psi \in X} \mathbf{FG}(\psi[X]_\nu)$, we can choose i so that it also satisfies $w_i \models \bigwedge_{\psi \in X} \mathbf{G}(\psi[X]_\nu)$. □

The LDBA construction tracks in the initial component $\mathit{af}(\varphi, w_{0i})$, i.e. after reading a finite word w_{0i} the initial component is in state $[\mathit{af}(\varphi, w_{0i})]_\sim$. The states $[\mathit{af}(\varphi, w_{0i})]_\sim$ are then connected to the accepting component by jumps that guess sets X and Y and the

[1] The translation to NBAs presented in [EKS18] is different from the construction we present here.

stabilisation point i. The jump leads to the corresponding initial state of the intersection automaton ($\mathcal{B}_{X,Y}$) of three DBAs, which are in charge of checking ($1'$), ($2'$), and ($3'$), and we refer to these automata by $\mathcal{B}^1_X$, $\mathcal{B}^2_{X,Y}$, and $\mathcal{B}^3_{X,Y}$, respectively.

To be more precise: recall that $af([\varphi]_\sim, w_{0i})_\sim \in \mathit{Reach}(\varphi)_{/\sim}$ for every word w and every $i \geq 0$. For every $[\psi]_\sim \in \mathit{Reach}(\varphi)_{/\sim}$ and for each pair of sets X, Y we construct a DBA $\mathcal{B}_{\psi,X,Y}$ recognising the intersection of the languages of the formulas:

$$\psi[X]_\nu \qquad \bigwedge_{\psi \in X} \mathbf{GF}(\psi[Y]_\mu) \qquad \bigwedge_{\psi \in Y} \mathbf{G}(\psi[X]_\nu)$$

These formulas belong to νLTL, $\mathbf{GF}(\mu LTL)$, and νLTL, respectively, and so we can obtain DBAs for them following the recipes of Proposition 5.1. Note that DBAs $\mathcal{B}_{\psi,X,Y}$ and $\mathcal{B}_{\chi,X,Y}$ have on common states the same transition relation and accepting states. We take advantage of this and combine all $\mathcal{B}_{\psi,X,Y}$ into a single structure that we call $\mathcal{B}_{X,Y}$. Formally, we construct a *semi-deterministic*[2] Büchi automaton – a Büchi automaton with a deterministic transition relation, but with potentially multiple initial states – recognising the union of all $\mathcal{B}_{\psi,X,Y}$ by taking the (potentially non-disjoint) union of states, transitions, initial states, and accepting states. Consequently, $\mathcal{B}_{X,Y}$ has a set of initial states $Q_{0,X,Y}$ containing for each $[\psi]_\sim \in \mathit{Reach}(\varphi)_{/\sim}$ an initial state $([\psi[X]_\nu]_\sim, q_0', q_0'') \in Q_{0,X,Y}$. Summarising, we obtain:

Theorem 6.2. *Let φ be a formula and let $\sim$ be an af- and $\cdot[\cdot]_\nu$-congruence. We define for each $X \subseteq \mu(\varphi)$ and each $Y \subseteq \nu(\varphi)$ the following, from the equivalence relation $\sim$ constructed, DBAs:*

- $\mathcal{B}^1_X = \bigcup_{[\psi]_\sim \in \mathit{Reach}(\varphi)_{/\sim}} \mathcal{B}_\nu^{\psi[X]_\nu}$ *is the (semi-deterministic) union of DBAs from Proposition 5.1.*[3]
- $\mathcal{B}^2_{X,Y} = \bigcap_{\psi \in X} \mathcal{B}^{\psi[Y]_\mu}_{\mathbf{GF}\mu}$ *is the intersection of DBAs from Proposition 5.1.*
- $\mathcal{B}^3_{X,Y}$ *is the DBA for $\bigwedge_{\psi \in Y} \mathbf{G}(\psi[X]_\nu)$ from Proposition 5.1.*

Let $\mathcal{B}_{X,Y}$ be the (semi-deterministic) intersection of the DBAs $\mathcal{B}^1_X$, $\mathcal{B}^2_{X,Y}$, and $\mathcal{B}^3_{X,Y}$:

$$\mathcal{B}_{X,Y} = (Q_{X,Y}, \delta_{X,Y}, Q_{0,X,Y}, \mathit{inf}(\alpha_{X,Y}))$$

Then the following LDBA over the alphabet 2^{Ap} recognises $\mathcal{L}(\varphi)$:

$$\mathcal{A}_{\mathsf{LDBA}}(\varphi) = (Q, \delta, [\varphi]_\sim, \mathit{inf}(\alpha))$$

where we define the states Q, the transition relation δ, and the accepting states α in the following way:

[2] The term appears in [VW86b] and coincides with the above mentioned definition. However, later publications might call an automaton semi-deterministic, when it is limit-deterministic or deterministic-in-the-limit. In our case semi-determinism is stricter than limit-determinism and these terms are not interchangeable.

[3] Similar to Proposition 7.13 $\sim$ needs to be an $\cdot[\cdot]_\nu$-congruence to ensure that the application of $\cdot[\cdot]_\nu$ is well-defined for the transition to the accepting component. However, after the transition $\cdot[\cdot]_\nu$ is not used anymore, thus any *af*-congruence $\approx$ with $\sim \preccurlyeq \approx$ can be used for constructing $\mathcal{B}^1_X$, $\mathcal{B}^2_{X,Y}$, and $\mathcal{B}^3_{X,Y}$.

- $Q = \mathit{Reach}(\varphi)_{/\sim} \cup \bigcup\{Q_{X,Y} : X \subseteq \mu(\varphi), Y \subseteq \nu(\varphi)\}$. *That is, a state is either an equivalence class* $[\psi]_\sim$ *or a product state* $([\psi]_\sim, q', q'') \in \mathcal{B}_{X,Y}$ *from one of the accepting components.*

- $\delta = af_\sim \cup \delta_\frown \cup \delta_{X,Y}$. *That is, a transition is either within the initial component* ($af_\sim$)*, within an accepting component* ($\delta_{X,Y}$)*, or it is a jump* ($\delta_\frown$) *from initial to accepting component. Let* $\delta_\frown$ *be defined for each state* $[\psi]_\sim \in \mathit{Reach}(\varphi)_{/\sim}$ *and each letter* $\sigma \in 2^{Ap}$ *as:*

$$\delta_\frown([\psi]_\sim, \sigma) = \bigcup\{\delta_{X,Y}(q_0, \sigma) : q_0 = ([\psi[X]_\nu]_\sim, q_0', q_0'') \in Q_{0,X,Y}, X \subseteq \mu(\varphi), Y \subseteq \nu(\varphi)\}$$

- $\alpha = \bigcup\{\alpha_{X,Y} : X \subseteq \mu(\varphi), Y \subseteq \nu(\varphi)\}$.

Note that the automata $\mathcal{B}^1_X$ and $\mathcal{B}^3_{X,Y}$ are in-fact *weak*, meaning that the states of every strongly connected component (SCC) are either all accepting or all rejecting. Intersecting a DBA with a weak DBA is simpler compared to the general case and it suffices to use the classical product construction for automata on finite words. This means it is not necessary to add additional information to the product states to track the progress of the Büchi acceptance conditions, e.g. multiple copies of the state space or a round-robin counter. Before we move on to the proof of the theorem, we illustrate the construction using an example:

Example 6.3. *Let* $\varphi = \mathbf{F}a \vee \mathbf{FG}b$. *The LDBA* $\mathcal{A}_{\mathsf{LDBA}}(\varphi)$ *constructed from Theorem 6.2 using* $\sim_p$ *as the underlying equivalence relation is depicted in Figure 6.1. Further, we use the following simplified versions of* $\cdot[\cdot]_\nu$ *and* $\cdot[\cdot]_\mu$ *to enhance readability:*

$$(\mathbf{F}\psi)[Y]_\nu = \begin{cases} \mathbf{tt} & \text{if } \mathbf{F}\psi \in Y \\ \mathbf{ff} & \text{otherwise.} \end{cases} \qquad (\mathbf{G}\psi)[X]_\mu = \begin{cases} \mathbf{tt} & \text{if } \mathbf{G}\psi \in X \\ \mathbf{ff} & \text{otherwise.} \end{cases}$$

Observe that transitions within the upper half (initial component) and the lower half (accepting component) are deterministic and only transitions from the upper to the lower part add nondeterminism. In this drawing we omitted several accepting components, e.g. $X = \{\mathbf{F}a\}, Y = \{\mathbf{G}b\}$ *or* $X' = \{\mathbf{FG}b\}, Y' = \{\}$*, and some non-accepting states. We mark these omissions by dashed outgoing transitions.*

This example illustrates already some potential optimisations for an implementation: First, $\mathcal{B}^1_X$ *and* $\mathcal{B}^3_{X,Y}$ *can be combined into a single component, since both are in charge of accepting formulas from* νLTL*. Second, the optimised version of the Master Theorem (Proposition 4.18) allows the removal of redundant components. In this example the component* $\mathcal{B}_{\emptyset,\emptyset}$ *would be enough, since it is sufficient for recognising all words of the language* $\mathcal{L}(\varphi)$*.*

Proof of Theorem 6.2. Let φ be a formula, let $\mathcal{A}_{\mathsf{LDBA}}(\varphi)$ be the corresponding automaton constructed from a suitable equivalence relation $\sim$, and let w be a word. Further, we denote by $q_{\chi,X,Y} \in Q_{0,X,Y}$ the initial state of $\mathcal{B}_{X,Y}$ with $[\chi]_\sim = [\psi[X]_\nu]_\sim$ in the first component for every reachable state $[\psi]_\sim \in \mathit{Reach}(\varphi)_{/\sim}$ and for all sets of formulas $X \subseteq \mu(\varphi)$ and $Y \subseteq \nu(\varphi)$. We denote by $\mathcal{L}(q_{\chi,X,Y})$ the language accepted from $q_{\chi,X,Y}$

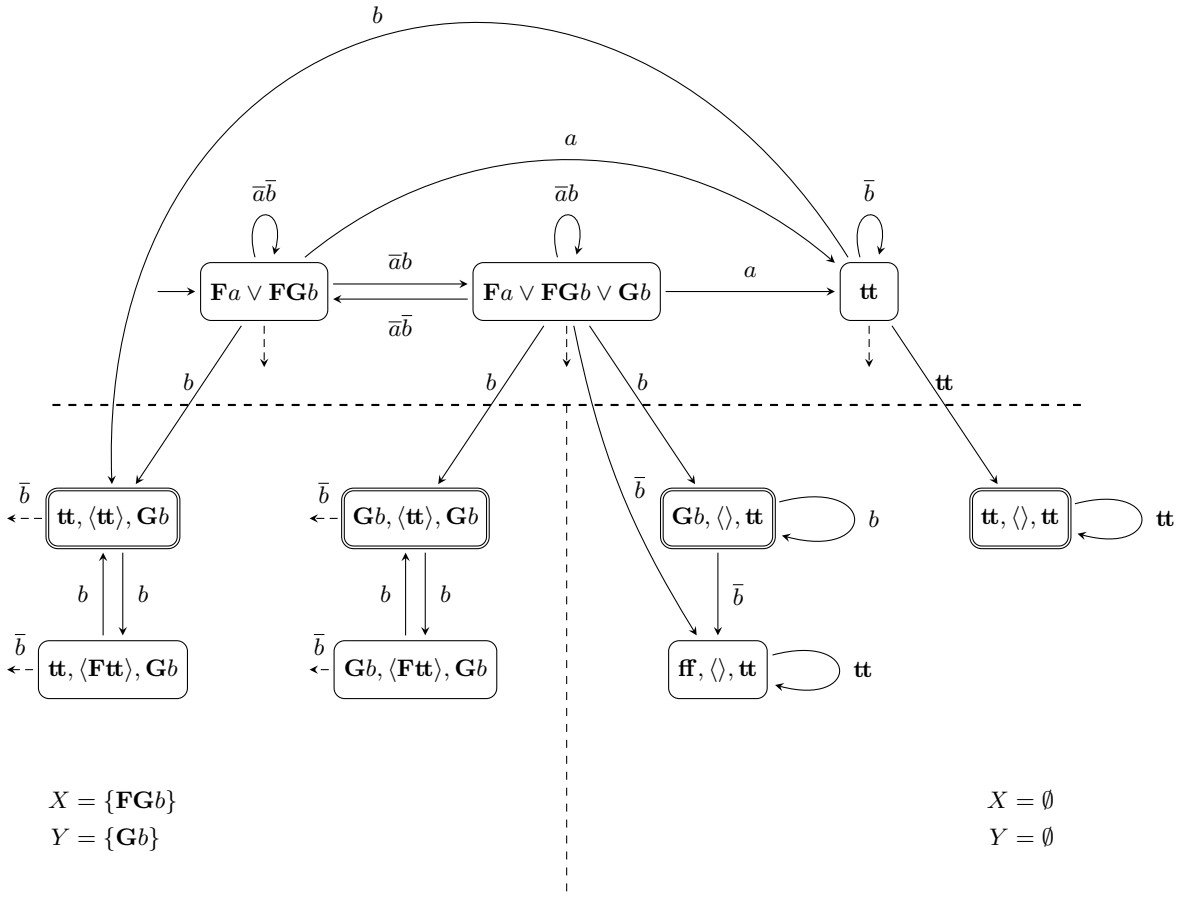


Figure 6.1: $\mathcal{A}_{\mathsf{LDBA}}(\varphi)$ for $\varphi = \mathbf{F}a \vee \mathbf{FG}b$. Omitted states and transitions are indicated by dashed arrows.

on the automaton $\mathcal{B}_{X,Y}$ and using Proposition 5.1 we can characterise the language $\mathcal{L}(q_{\chi,X,Y})$ in terms of LTL:

$$\begin{aligned}\forall v.\ v \in \mathcal{L}(q_{\psi[X]_\nu,X,Y}) \iff\ & v \models \psi[X]_\nu \\ & \wedge \forall \psi \in X.\ v \models \mathbf{GF}(\psi[Y]_\mu) \\ & \wedge \forall \psi \in Y.\ v \models \mathbf{G}(\psi[X]_\nu)\end{aligned}$$

($\Rightarrow$) Assume w is in $\mathcal{L}(\varphi)$. By Corollary 6.1 there exist i, X, and Y such that (1'-3') hold. We now construct an accepting run r on $\mathcal{A}_{\mathsf{LDBA}}(\varphi)$. The run r follows for the first $i-1$ steps the initial component and reaches $af_\sim([\varphi]_\sim, w_{0i}) = [af(\varphi, w_{0i})]_\sim = [\psi]_\sim$ for some ψ. The run then branches off to the accepting component $\mathcal{B}_{X,Y}$. Notice that (1'-3') exactly matches the right-hand side of our characterisation of $\mathcal{L}(q_{\psi[X]_\nu,X,Y})$ and thus we have $w_i \in \mathcal{L}(q_{\psi[X]_\nu,X,Y})$. Thus there exists an accepting run r' for w_i on $\mathcal{B}_{X,Y}$ starting in $q_{\psi[X]_\nu,X,Y}$. Observe that the we can reach from $[\psi]_\sim$ the same states as $q_{\psi[X]_\nu,X,Y}$, since $\delta_\frown([\psi]_\sim, w[i]) \cap Q_{X,Y} = \delta_{X,Y}(q_{\psi[X]_\nu,X,Y}, w[i])$. Thus r simply follows the accepting run r' from this point on and we are done.

($\Leftarrow$) Assume w is accepted by $\mathcal{A}_{\mathsf{LDBA}}(\varphi)$. Then there exists an accepting run r and since every accepting state is located in some $\mathcal{B}_{X,Y}$ the run r eventually transitions to some $\mathcal{B}_{X,Y}$. Let now i be the time at which the run moves to $\mathcal{B}_{X,Y}$ for some $[\psi]_\sim = [af(\varphi, w_{0i})]_\sim = af_\sim([\varphi]_\sim, w_{0i})$, some $X \subseteq \mu(\varphi)$, and some $Y \subseteq \nu(\varphi)$. Since the run r is accepting and $\delta_\frown([\psi]_\sim, w[i]) \cap Q_{X,Y} = \delta_{X,Y}(q_{\psi[X]_\nu,X,Y}, w[i])$, we can construct an accepting run r' on $\mathcal{B}_{X,Y}$ for w_i:

$$r'[j] = \begin{cases} q_{\psi[X]_\nu,X,Y} & \text{if } j = 0 \\ r[i+j] & \text{otherwise.} \end{cases}$$

Thus $w \in \mathcal{L}(q_{\psi[X]_\nu,X,Y})$ and using our characterisation of this language we end up with (1'-3') of Corollary 6.1. Consequently $w \models \varphi$ and we are done. □

Complexity Analysis. We base our analysis on the equivalence relation $\sim_p$. Let φ be a formula of length n. Since $\mathcal{A}_{\mathsf{LDBA}}(\varphi)$ is an union and intersection of several components, the number of states can be bounded by the size of the components:

$$|Reach(\varphi)_{/\sim_p}| + \sum_{\substack{X \subseteq \mu(\varphi) \\ Y \subseteq \nu(\varphi)}} (|Q^1_X| \cdot |Q^2_{X,Y}| \cdot |Q^3_{X,Y}|)$$

We bound the size of each of these components by a doubly-exponential function. In the case of $\mathcal{B}^1_X$ it is not immediately clear from the previous results how to do this, because $\mathcal{B}^1_X$ contains for each element of $Reach(\varphi)_{/\sim_p}$ a potentially 2^{2^n}-sized automaton. However, observe that all equivalence classes in Q^1_X correspond to a Boolean function over the same set of proper subformulas of size at most $|sf(\varphi)| \leq n$ and so Q^1_X has at most 2^{2^n} elements. $\mathcal{B}^2_{X,Y}$ is an intersection of $|X|$-many automata with at most $2^{2^{n+1}}$ states. We bound $|X|$ by $|\mu(\varphi)| \leq n$ and thus the number of states is at most $n \cdot (2^{2^{n+1}})^n = 2^{2^{n+(\log_2 n)+(\log_2 \log_2 n)+1}}$. $\mathcal{B}^3_{X,Y}$ is constructed from the formula $\bigwedge_{\psi \in Y} \mathbf{G}(\psi[X]_\nu)$. This formula has at most $|sf(\varphi)| + |\nu(\varphi)|$ proper subformulas and thus the automaton has at most $2^{2^{|sf(\varphi)|+|\nu(\varphi)|}} \leq 2^{2^{2n}}$ states. Going back to our initial bound and inserting the upper

bounds for the components we obtain:

$$2^{2^n} + 2^{|\mu(\varphi)|+|\nu(\varphi)|}(2^{2^n} \cdot 2^{2^{n+(\log_2 n)+(\log_2 \log_2 n)+1}} \cdot 2^{2^{2n}}) \leq 2^{2^n} + 2^{2^{4n+2}} \leq 2^{2^{4n+3}} \in 2^{2^{O(n)}}$$

This construction can be further refined by replacing components, especially $\mathcal{B}^2_{X,Y}$, by smaller components as described in Chapter 7. Recall that the lower bound for the blowup of a translation of LTL to LDBA is double exponential [SEJK16].

The definition of an LDBA allows the initial component to be nondeterministic. However, in our construction we make it deterministic and thus every accepting run has exactly one nondeterministic step. This (and some other technical details) make this LDBA usable for quantitative (and not only qualitative) probabilistic model checking, as described in [SEJK16] and Section 9.1.

6.2 NBAs for μLTL, νLTL, $\mathbf{GF}(\mu LTL)$, and $\mathbf{FG}(\nu LTL)$

Before we start building NBAs for general LTL formulas (Section 6.3), we define constructions for fragments of LTL. For this we adapt in this section *af* to the nondeterministic setting by putting the result of applying *af* into disjunctive normal form. This step reduces the size of the automata for 'simple' languages from being double exponential to just exponential by using nondeterminism to guess the clause of the formula that is going to be satisfied. From this newly defined $af_\vee$ we then obtain the automaton constructions. Thus we have three subsections: first, we define what we mean by disjunctive normal form; second, we introduce $af_\vee$ and prove fundamental properties about this function; third, we give the automata constructions.

But before jumping into technical definitions and lemmas, let us start with some informal intuition: Consider the formula $\varphi = \mathbf{GX}(a \vee b)$. In the automaton obtained from Proposition 5.1 we find states for the formulas φ and $af(\varphi, \{a\})$ and the transition:

$$[\varphi]_\sim \xrightarrow{a\bar{b}} [af(\varphi, \{a\})]_\sim = [\varphi \wedge (a \vee b)]_\sim$$

Putting *af* into DNF means that we rewrite $\varphi \wedge (a \vee b)$ to $(\varphi \wedge a) \vee (\varphi \wedge b)$ and instead of a single transition, we now have two transitions:

$$\varphi \xrightarrow{a\bar{b}} \varphi \wedge a \quad \text{and} \quad \varphi \xrightarrow{a\bar{b}} \varphi \wedge b$$

In other words, the nondeterminism is used to guess which of the two disjuncts of the DNF is going to hold. To be more precise, we replace the equivalence classes by clauses, sets of modal operators and literals, that represent conjunctions. Thus we will find in the NBA for φ the following two transitions:

$$\{\varphi\} \xrightarrow{a\bar{b}} \{\varphi, a\} \quad \text{and} \quad \{\varphi\} \xrightarrow{a\bar{b}} \{\varphi, b\}$$

As previously mentioned *af* deterministically tracks the current run-DAG level of the for an LTL formula canonical VWAA [MSS88; Var94]. Similar $af_\vee$ *nondeterministically* tracks the current level of the run-DAG which resembles the idea used in tableau constructions [GPVW95]. However, the NBAs that we obtain for arbitrary LTL formulas are

based on a product construction and are structurally different to the automata obtained through previously known tableau constructions.

6.2.1 Disjunctive Normal Form

We now introduce the standard definition of a disjunctive normal form and the related notation. By $X \otimes Y \coloneqq \{A \cup B : A \in X, B \in Y\}$ we denote the pair-wise union of two sets of sets. This intuitively corresponds to computing the set of satisfying assignments for a conjunction given satisfying assignments for both conjuncts. We denote by $\bigotimes\{X_1, X_2, \dots X_n\} \coloneqq \{A_1 \cup A_2 \cup \dots \cup A_n : A_1 \in X_1, A_2 \in X_2, \dots A_n \in X_n\}$ the lifting of $\otimes$ to finite sets. Further, we also lift $\cup$ to finite sets, denoted by $\bigcup\{X_1, X_2, \dots X_n\} \coloneqq X_1 \cup X_2 \cup \dots \cup X_n$.

In general Boolean functions do not possess a unique, minimal[4] disjunctive normal form. However, in our context we interpret LTL formulas as monotone Boolean functions, i.e., a and $\neg a$ are mapped to distinct variables x_a and $x_{\neg a}$, and in this specific context there exists a unique minimal disjunctive normal form for each formula. We compute the minimal set of satisfying assignments using $min(X) \coloneqq \{A \in X : \forall B \in X.\ B \not\subset A\}$ where X is a finite set. We then use the following shorthands: $X \cup_{min} Y \coloneqq min(X \cup Y)$, $X \otimes_{min} Y \coloneqq min(X \otimes Y)$, and $\bigotimes_{min} \mathcal{X} \coloneqq min(\bigotimes \mathcal{X})$.

When working with these definitions the following identities are useful and allow easier compositional reasoning: $\bigotimes_{min} \emptyset = \{\emptyset\}$, $\bigotimes_{min}\{X\} = min(X)$, and $\bigotimes_{min}(\mathcal{X} \cup \mathcal{Y}) = (\bigotimes_{min} \mathcal{X}) \otimes_{min} (\bigotimes_{min} \mathcal{Y})$ where X, $\mathcal{X}$, $\mathcal{Y}$, and the elements of $\mathcal{X}$ and $\mathcal{Y}$ are finite sets. Lastly, for each set A one can 'trace-back' the contribution a particular X in $\mathcal{X}$ made to A: $A \in \bigotimes_{min} \mathcal{X} \wedge X \in \mathcal{X} \implies \exists B \in X.\ B \subseteq A$ where X, $\mathcal{X}$, and the elements of $\mathcal{X}$ are finite sets. We are now equipped with necessary tools to define the minimal disjunctive normal form and state the well-known relation to satisfying assignments:

Proposition 6.4 (Minimal Disjunctive Normal Form)**.** *Let φ be a formula. We recursively define the minimal disjunctive normal form $dnf(\varphi)$ as follows:*

$$
\begin{array}{lll}
dnf(\mathbf{tt}) & = \{\emptyset\} & dnf(\mathbf{X}\varphi) = \{\{\mathbf{X}\varphi\}\} \\
dnf(\mathbf{ff}) & = \emptyset & \\
dnf(a) & = \{\{a\}\} & dnf(\varphi \mathbf{U} \psi) = \{\{\varphi \mathbf{U} \psi\}\} \\
dnf(\neg a) & = \{\{\neg a\}\} & dnf(\varphi \mathbf{M} \psi) = \{\{\varphi \mathbf{M} \psi\}\} \\
dnf(\varphi \wedge \psi) & = dnf(\psi_1) \otimes_{min} dnf(\psi_2) & dnf(\varphi \mathbf{R} \psi) = \{\{\varphi \mathbf{R} \psi\}\} \\
dnf(\varphi \vee \psi) & = dnf(\psi_1) \cup_{min} dnf(\psi_2) & dnf(\varphi \mathbf{W} \psi) = \{\{\varphi \mathbf{W} \psi\}\}
\end{array}
$$

Further, let $\mathcal{I}$ be a propositional assignment. Then:

$$\mathcal{I} \models_p \varphi \iff \exists \Psi \subseteq \mathcal{I}.\ \Psi \in dnf(\varphi)$$

Proof sketch. This is proven by a straight-forward structural induction on φ. □

Example 6.5. *Let* $\varphi = a \vee \overbrace{((a \wedge \mathbf{X}b) \vee \mathbf{F}(b \vee c))}^{\psi}$. *We then compute* $dnf(a) = \{\{a\}\}$, $dnf(a \wedge \mathbf{X}b) = \{\{a, \mathbf{X}b\}\}$, *and* $dnf(\mathbf{F}(b \vee c)) = \{\{\mathbf{F}(b \vee c)\}\}$. *Then*

$$dnf(\psi) = \{\{a, \mathbf{X}b\}, \{\mathbf{F}(b \vee c)\}\}$$

[4] Minimal in the sense that there exists no other clause representation of the Boolean function with fewer clauses.

and finally:

$$\begin{aligned} \mathit{dnf}(\varphi) &= \mathit{min}(\{\{a\}\} \cup \{\{a, \mathbf{X}b\}, \{\mathbf{F}(b \vee c)\}\}) \\ &= \mathit{min}(\{\{a\}, \{a, \mathbf{X}b\}, \{\mathbf{F}(b \vee c)\}\}) \\ &= \{\{a\}, \{\mathbf{F}(b \vee c)\}\} \end{aligned}$$

6.2.2 Disjunctive *af*

We now define based on *dnf* from the preceding subsection the disjunctive version of *af*.

Definition 6.6 (Disjunctive *af*)**.** *Let Ψ be a set of formulas and let σ be a letter. We then define $af_\vee$ as:*

$$af_\vee(\Psi, \sigma) \coloneqq \bigotimes_{min} \{\mathit{dnf}(af(\psi, \sigma)) : \psi \in \Psi\}$$

Furthermore, we generalise the definition of $af_\vee$ and define the set of reachable clauses. Let φ be a formula and let w be a finite word. We then define:

$$\begin{aligned} af_\vee(\Psi, \epsilon) &\coloneqq \{\Psi\} \\ af_\vee(\Psi, w\sigma) &\coloneqq \bigcup \{af_\vee(\Psi', \sigma) : \Psi' \in af_\vee(\Psi, w)\} \\ af_\vee(\varphi, w) &\coloneqq \bigcup \{af_\vee(\Psi, w) : \Psi \in \mathit{dnf}(\varphi)\} \\ \mathit{Reach}_\vee(\varphi) &\coloneqq \bigcup \{af_\vee(\varphi, w) : w \in (2^{Ap})^*\} \end{aligned}$$

Example 6.7. *Let $\varphi = \mathbf{X}a \vee \mathbf{X}(a \wedge b)$. Let us now compute $af_\vee(\varphi, \{\})$ and $af_\vee(\varphi, \{\}\{b\})$:*

$$\begin{aligned} af_\vee(\varphi, \{\}) &= \bigcup \Big\{ af_\vee(\Psi, \{\}) : \Psi \in \mathit{dnf}(\varphi) \Big\} \\ &= \bigcup \Big\{ af_\vee(\{\mathbf{X}a\}, \{\}), af_\vee(\{\mathbf{X}(a \wedge b)\}, \{\}) \Big\} \\ &= \bigcup \Big\{ \mathit{dnf}(a), \mathit{dnf}(a \wedge b) \Big\} \\ &= \bigcup \Big\{ \{\{a\}\}, \{\{a, b\}\} \Big\} = \{\{a\}, \{a, b\}\} \end{aligned}$$

$$\begin{aligned} af_\vee(\varphi, \{\}\{b\}) &= \ldots \text{ — in the same manner as before} \\ &= \bigcup \Big\{ af_\vee(\{a\}, \{b\}), af_\vee(\{a, b\}, \{b\}) \Big\} \\ &= \bigcup \Big\{ \mathit{dnf}(\mathbf{ff}), \mathit{dnf}(\mathbf{ff}) \otimes_{min} \mathit{dnf}(\mathbf{ff}) \Big\} \\ &= \bigcup \Big\{ \{\}, \{\} \Big\} = \emptyset \end{aligned}$$

In fact with the words $\{\}$ and $\{\}\{b\}$ we discovered most of the clauses. The last, unexplored clause $\emptyset$ can be reached with the word $\{\}\{a\}$. Thus the set of all from φ reachable clauses is: $\mathit{Reach}_\vee(\varphi) = \{\{\mathbf{X}a\}, \{\mathbf{X}(a \wedge b)\}, \{a, b\}, \{a\}, \{\}\}$.

We now transfer the knowledge we have about *af* to $af_\vee$, which will be useful when we prove correctness of the derived constructions. Ideally we want to prove:

$$\mathit{dnf}(af(\varphi, w)) = af_\vee(\varphi, w)$$

Such a statement would put (minimal) satisfying assignments of $af(\varphi, w)$ and the clauses of $af_\vee(\varphi, w)$ in a direct relation. Unfortunately, this statement is *not* true: Let $\varphi = \mathbf{X}a \vee \mathbf{X}(a \wedge b)$ from Example 6.7 and let σ be an arbitrary letter. On the one hand, we have $af(\varphi, \sigma) = a \vee (a \wedge b)$ and thus $dnf(af(\varphi, \sigma)) = \{\{a\}\}$. On the other hand, we have $af_\vee(\varphi, \sigma) = \bigcup\{af_\vee(\{\mathbf{X}a\}, \sigma), af_\vee(\{\mathbf{X}(a \vee b)\})\} = \{\{a\}, \{a, b\}\}$, which is not equal.

While our hypothesis turned out to be wrong, this counterexample gives us the insight that $af_\vee(\varphi, w)$ might over-approximate the set of (minimal) satisfying assignments, but $dnf(af(\varphi, w))$ is always a subset. Thus we arrive at the following lemma:

Lemma 6.8. *Let φ be a formula, let w be a finite word, and let $\mathcal{I}$ be a propositional assignment. Then:*

$$\mathcal{I} \models_p af(\varphi, w) \iff \exists \Psi \subseteq \mathcal{I}.\ \Psi \in af_\vee(\varphi, w)$$

In order to keep the section short the proofs of Lemmas 6.8 and 6.9 are moved to Section 6.A. Applying Lemma 6.8 to the counterexample we see that $af_\vee(\varphi, \sigma)$ only contains satisfying assignments ($\mathcal{I} = \{a\}$, $\mathcal{J} = \{a, b\}$) for $af(\varphi, \sigma)$. Further, propositional assignments that do not satisfy $af(\varphi, \sigma)$ such as $\mathcal{K} = \{b\}$ are not in $af_\vee(\varphi, \sigma)$ and all subsets ($\emptyset$) of $\mathcal{K}$ are also not an element of $af_\vee(\varphi, \sigma)$.

Based on Lemma 6.8 we establish more results, most notably: (a) the empty set $\emptyset$ precisely reflects when af is propositionally equivalent to $\mathbf{tt}$ or $\mathbf{ff}$, (b) $af_\vee$ decomposes the language into a union of intersections, and (c) that $\cdot[\cdot]_\nu$ commutes with $af_\vee$ in the expected way. Let us now denote with $\Psi[X]_\nu \coloneqq \{\psi[X]_\nu : \psi \in \Psi\}$ the application of $\cdot[\cdot]_\nu$ on each element of Ψ.

Lemma 6.9. *Let φ be a formula, let w be a finite word, let $X \subseteq \mu(\varphi)$ be a set of subformulas, and let $\mathcal{I} \subseteq sf(\varphi)$ be a propositional assignment that only contains proper subformulas. Then:*

1. $af_\vee(\varphi, w) \subseteq 2^{sf(\varphi)}$
2. $\emptyset \in af_\vee(\varphi, w) \iff af(\varphi, w) \sim_p \mathbf{tt}$
3. $af_\vee(\varphi, w) = \emptyset \iff af(\varphi, w) \sim_p \mathbf{ff}$
4. $\mathcal{L}(af(\varphi, w)) = \bigcup \left\{ \bigcap_{\psi \in \Psi} \mathcal{L}(\psi) : \Psi \in af_\vee(\varphi, w) \right\}$
5. $\mathcal{I} \models_p af(\varphi, w)[X]_\nu \iff \exists \Psi.\ \Psi[X]_\nu \subseteq \mathcal{I} \wedge \Psi \in af_\vee(\varphi, w)$

6.2.3 Automata Constructions

We first describe how to construct NBAs for the LTL fragments of Section 2.2.2. For the sake of completeness we also give a construction for $\mathbf{FG}(\nu LTL)$, but we do not use it later.

Proposition 6.10. *Let $\varphi \in \mu LTL$.*

– *The following NBA over the alphabet 2^{Ap} recognizes $\mathcal{L}(\varphi)$:*

$$\mathcal{B}_\mu^\varphi = (\ Reach_\vee(\varphi), af_\vee, dnf(\varphi), inf(\{\emptyset\})\ )$$

– *The following NBA over the alphabet* 2^{Ap} *recognizes* $\mathcal{L}(\mathbf{GF}\varphi)$*:*

$$\mathcal{B}^{\varphi}_{\mathbf{GF}\mu} = (\, Reach_{\vee}(\mathbf{F}\varphi) \cup \{\{\mathbf{F}\varphi\}\}, af_{\vee}^{\mathbf{F}\varphi}, \{\{\mathbf{F}\varphi\}\}, inf(\{\emptyset\})\,)$$

$$af_{\vee}^{\mathbf{F}\varphi}(\Psi, \sigma) = \begin{cases} \{\{\mathbf{F}\varphi\}\} & \text{if } \Psi = \emptyset \\ af_{\vee}(\Psi, \sigma) & \text{otherwise.} \end{cases}$$

Let $\varphi \in \nu LTL$*.*

– *The following NBA over the alphabet* 2^{Ap} *recognizes* $\mathcal{L}(\varphi)$*:*

$$\mathcal{B}^{\varphi}_{\nu} = (\, Reach_{\vee}(\varphi), af_{\vee}, dnf(\varphi), inf(Reach_{\vee}(\varphi))\,)$$

– *The following NBA over the alphabet* 2^{Ap} *recognizes* $\mathcal{L}(\mathbf{FG}\varphi)$*:*

$$\mathcal{B}^{\varphi}_{\mathbf{FG}\nu} = (\, Reach_{\vee}(\mathbf{G}\varphi) \cup \{\{\mathbf{FG}\varphi\}, \{\mathbf{G}\varphi\}\}, af_{\vee}^{\mathbf{G}\varphi}, \{\{\mathbf{FG}\varphi\}\}, inf(Reach_{\vee}(\mathbf{G}\varphi))\,)$$

$$af_{\vee}^{\mathbf{G}\varphi}(\Psi, \sigma) = \begin{cases} \{\{\mathbf{FG}\varphi\}, \{\mathbf{G}\varphi\}\} & \text{if } \Psi = \{\mathbf{FG}\varphi\} \\ af_{\vee}(\psi, \sigma) & \text{otherwise.} \end{cases}$$

Example 6.11. *Let* $\varphi = a \wedge \mathbf{X}(b \vee \mathbf{F}c)$*, the formula for which a DBA was given in Example 5.2. The NBA* $\mathcal{B}^{\varphi}_{\mathbf{GF}\mu}$ *for* φ *is shown in Figure 6.2. Compared to the DBA of Example 5.2, the NBA has a simpler structure, although in this case the same number of states.*

Example 6.12. *Let* $\varphi = \bigvee_{i=1}^{3} \mathbf{F}(a_i \wedge \mathbf{XF}b_i)$*. In Example 6.11 we showed that the NBA has a simpler structure, but the number of states stayed the same. In this example we look at the formula* φ*, which yields a polynomial-sized nondeterministic automaton for* φ *(Figure 6.3), while we obtain an exponential-sized deterministic automaton (Figure 6.4). A closer look at this is taken in the complexity analysis paragraph of Section 5.1 and Section 6.2.*

Proof of Proposition 6.10. Let φ be a formula of μLTL. We want to prove (1) $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{B}^{\varphi}_{\mu})$ and (2) $\mathcal{L}(\mathbf{GF}\varphi) = \mathcal{L}(\mathcal{B}^{\varphi}_{\mathbf{GF}\mu})$. Let w now be an arbitrary word.

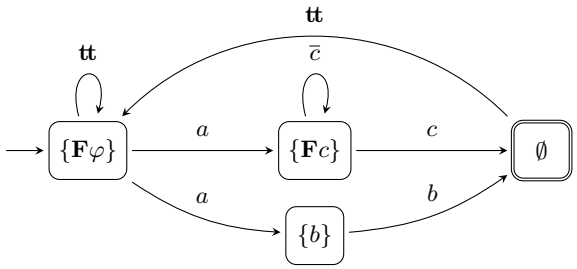


Figure 6.2: NBA $\mathcal{B}^{\varphi}_{\mathbf{GF}\mu}$ for $\varphi = a \wedge \mathbf{X}(b \vee \mathbf{F}c)$. Notice the different self-loops on $\{\mathbf{F}\varphi\}$ and $\{\mathbf{F}c\}$. In the first state the automaton guesses nondeterministically when to track φ and in the second state the construction already resolves the nondeterminism.

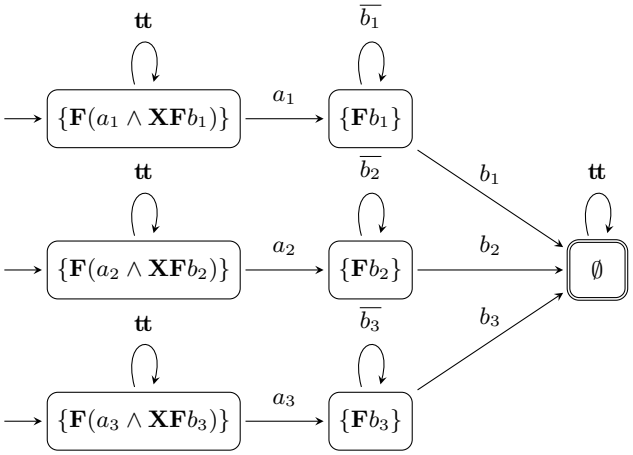


Figure 6.3: NBA $\mathcal{B}_\mu^\varphi$ for $\varphi = \bigvee_{i=1}^{3} \psi_i$ with $\psi_i = a_i \wedge \mathbf{XF}b_i$. Notice that this NBA repeats for each ψ_i the same structure and has in total $2n + 1 = 7$ states.

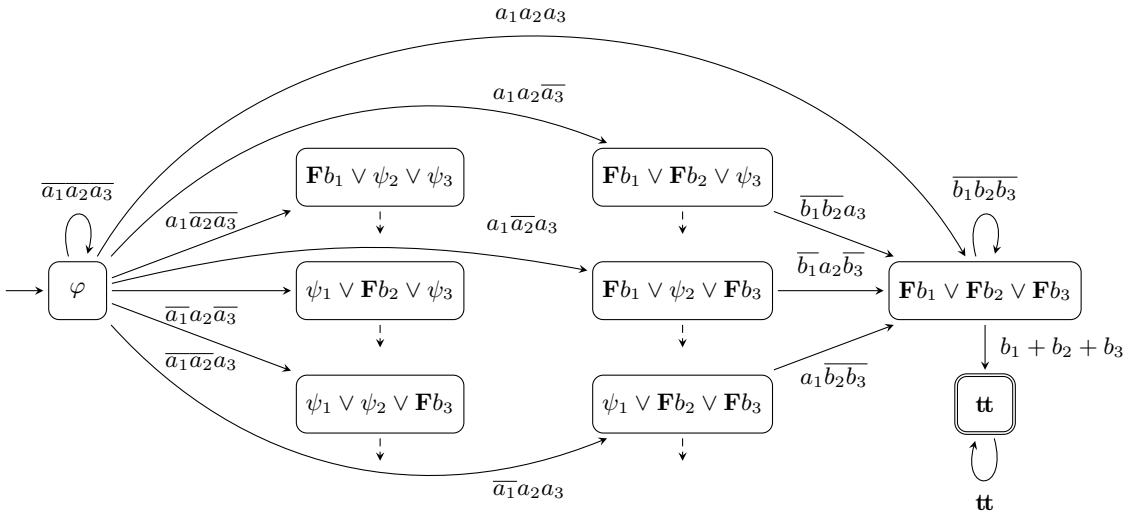


Figure 6.4: DBA $\mathcal{B}_\mu^\varphi$ for $\varphi = \bigvee_{i=1}^{3} \psi_i$ with $\psi_i = a_i \wedge \mathbf{XF}b_i$ from Proposition 5.1 using $\sim_l$ as the underlying equivalence relation. Omitted transitions are indicated by dashed arrows. This automaton has $2^n + 1 = 9$ states.

($\Rightarrow_1$) Assume $w \models \varphi$. Using Lemma 3.9 with $\sim_p$ we have $af(\varphi, w_{0i}) \sim_p \mathbf{tt}$ for some i and also $\emptyset \in af_\vee(\varphi, w_{0i})$ using Lemma 6.9. Hence there exists some a path from the initial states to $\emptyset$ labeled w_{0i}. Let r be the run that follows this path and loops in state $\emptyset$ afterwards. Thus r is accepting and the word w is accepted by $\mathcal{B}^\varphi_\mu$.

($\Leftarrow_1$) Assume w is accepted by $\mathcal{B}^\varphi_\mu$. Then there exists an accepting run r for w and this run visits the state $\emptyset$ infinitely often. Let be i be the index such that $\emptyset \in af_\vee(\varphi, w_{0i})$. Applying Lemma 6.9 we have $af(\varphi, w_{0i}) \sim_p \mathbf{tt}$ and with Lemma 3.9 we also have $w \models \varphi$.

($\Rightarrow_2$) Assume $w \models \mathbf{GF}\varphi$. This is equivalent to $\forall i.\, w_i \models \mathbf{F}\varphi$ and $\forall i.\exists j.\, af(\mathbf{F}\varphi, w_{ij}) \sim_p \mathbf{tt}$ (Lemma 3.9). By applying Lemma 6.8 we obtain $\forall i.\exists j.\, \emptyset \in af_\vee(\mathbf{F}\varphi, w_{ij})$. From this we now construct an accepting run r piecewise. We start by setting $r[0] = \{\mathbf{F}\varphi\}$. Let $j > i$ be the smallest index for $i = 0$, such that $\emptyset \in af_\vee(\mathbf{F}\varphi, w_{0j})$. Then the run follows this path to reach $\emptyset$ for the segment w_{0j} and thus $r[j] = \emptyset$. After visiting the accepting state the run returns to the initial state and we repeat this from $r[j+1] = \{\mathbf{F}\varphi\}$. Applying this construction again and again we obtain an accepting run and thus the word w is accepted by $\mathcal{B}^\varphi_{\mathbf{GF}\mu}$.

($\Leftarrow_2$) Assume w is accepted by $\mathcal{B}^\varphi_{\mathbf{GF}\mu}$. Then there exists an accepting run r for w and this run visits the state $\emptyset$ infinitely often. Provided we have shown $\forall i.\exists j \geq i.\exists k.\, af(\mathbf{F}\varphi, w_{jk}) \sim_p \mathbf{tt}$, we apply Lemma 3.9 again to $w \models \mathbf{GF}\varphi$. We now focus on the remaining goal $\forall i.\exists j \geq i.\exists k.\, af(\mathbf{F}\varphi, w_{jk}) \sim_p \mathbf{tt}$: Let i be an arbitrary index. Since r is an accepting run, there exist infinitely many j's, such that $r[j-1] = \emptyset$ and $r[j] = \{\mathbf{F}\varphi\}$. Let j be such an index with $j > i$. Since r is an accepting run, there also exists some k, such that $r[k] = \emptyset$ and the run r has not visited $\emptyset$ in-between of j and k. Thus $\emptyset \in af_\vee(\varphi, w_{jk})$ and with Lemma 6.9 we fill the remaining gap.

We now move on to the second part concerned with νLTL. Let φ be a formula of νLTL. We want to prove (3) $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{B}^\varphi_\nu)$ and (4) $\mathcal{L}(\mathbf{FG}\varphi) = \mathcal{L}(\mathcal{B}^\varphi_{\mathbf{FG}\nu})$. Let w now be an arbitrary word. The proof of (3) is analogous to the proof of (1) and we are skipping it. Observe that $\mathcal{B}^\varphi_{\mathbf{FG}\nu}$ adds to the structure of $\mathcal{B}^{\mathbf{G}\varphi}_\nu$ just one additional, non-accepting state with a self-loop and a transition to $\{\mathbf{G}\varphi\}$.

($\Rightarrow_4$) Assume $w \models \mathbf{FG}\varphi$. Then there exits some i such that $w_i \models \mathbf{G}\varphi$. An accepting run just stays in $\mathbf{FG}\varphi$ up to i and then moves to $\mathbf{G}\varphi$. From this point on there exists a continuation according to (3) that is accepting.

($\Leftarrow_4$) Assume w is accepted by $\mathcal{B}^\varphi_{\mathbf{FG}\nu}$. Let r be the corresponding accepting run. r eventually leaves at some point i the state $\mathbf{FG}\varphi$ and moves to $\{\mathbf{G}\varphi\}$. By (3) we know that then $w_i \models \mathbf{G}\varphi$ and hence $w \models \mathbf{FG}\varphi$. □

Complexity Analysis. The elements of $Reach_\vee(\varphi)$ are *sets* of proper subformulas of φ, i.e. $Reach_\vee(\varphi) \subseteq 2^{sf(\varphi)}$. Since the number of proper subformulas is bounded by the length of the formula, we immediately obtain $|Reach_\vee(\varphi)| \leq 2^{|sf(\varphi)|} \leq 2^n$ where n is the length of φ. Thus all NBAs have at most $O(2^n)$ states. More precisely, the NBAs for μLTL and νLTL have at most $2^{|sf(\varphi)|}$, the NBAs for $\mathbf{GF}(\mu LTL)$ have at most $2^{|sf(\varphi)|+1}$, and the NBAs for $\mathbf{FG}(\nu LTL)$ have at most $2^{|sf(\varphi)|+1}+1$ states. Observe that the blow-up in translating LTL to NBAs for formulas constructed just using literals (a, $\neg a$), Boolean connectives ($\wedge$, $\vee$), and the modal operator $\mathbf{X}$ is already exponential, see e.g. [BK08].

6.3 NBAs for Arbitrary LTL Formulas

Equipped with NBA constructions for LTL fragments we go back to the construction in Theorem 6.2 and revise it such that we obtain NBAs of exponential size for arbitrary LTL formulas. We achieve this by simply replacing the deterministic building blocks (DBAs) with nondeterministic (NBAs) ones.

Theorem 6.13. *Let φ be a formula. We define for each $X \subseteq \mu(\varphi)$ and each $Y \subseteq \nu(\varphi)$ the following NBAs:*

- $\mathcal{B}^1_X = \bigcup_{\Psi \in Reach_\vee(\varphi)} \mathcal{B}_\nu^{\wedge \Psi[X]_\nu}$ *is the union of NBAs from Proposition 6.10.*
- $\mathcal{B}^2_{X,Y} = \bigcap_{\psi \in X} \mathcal{B}^{\psi[Y]_\nu}_{\mathbf{GF}\mu}$ *is the intersection of NBAs from Proposition 6.10.*
- $\mathcal{B}^3_{X,Y}$ *is the NBA for* $\bigwedge_{\psi \in Y} \mathbf{G}(\psi[X]_\nu)$ *from Proposition 6.10.*

Let $\mathcal{B}_{X,Y}$ be the intersection of the NBAs $\mathcal{B}^1_X$, $\mathcal{B}^2_{X,Y}$, and $\mathcal{B}^3_{X,Y}$:

$$\mathcal{B}_{X,Y} = (Q_{X,Y}, \delta_{X,Y}, Q_{0,X,Y}, inf(\alpha_{X,Y}))$$

Then the following NBA over the alphabet 2^{Ap} recognises $\mathcal{L}(\varphi)$:

$$\mathcal{A}_{\mathsf{NBA}}(\varphi) = (Q, \delta, dnf(\varphi), inf(\alpha))$$

where we define the states Q, the transition relation δ, and the accepting states α in the following way:

- $Q = Reach_\vee(\varphi) \cup \bigcup\{Q_{X,Y} : X \subseteq \mu(\varphi), Y \subseteq \nu(\varphi)\}$. *That is, a state is either a clause Ψ or a tuple of clauses $(\Psi, \Psi', \Psi'') \in \mathcal{B}_{X,Y}$ from one of the accepting components.*
- $\delta = af_\vee \cup \delta_\frown \cup \delta_{X,Y}$. *That is, a transition is either within the initial component ($af_\vee$), within an accepting component ($\delta_{X,Y}$), or it is a jump ($\delta_\frown$) from initial to accepting component. Let $\delta_\frown$ be defined for each state $\Psi \in Reach_\vee(\varphi)$ and each letter $\sigma \in 2^{Ap}$ as:*

$$\delta_\frown(\Psi, \sigma) = \bigcup\{\delta_{X,Y}(q_0, \sigma) : q_0 = (\Psi[X]_\nu, \Psi', \Psi'') \in Q_{0,X,Y}, X \subseteq \mu(\varphi), Y \subseteq \nu(\varphi)\}$$

- $\alpha = \bigcup\{\alpha_{X,Y} : X \subseteq \mu(\varphi), Y \subseteq \nu(\varphi)\}$.

Note that the automata $\mathcal{B}^1_X$ and $\mathcal{B}^3_{X,Y}$ are in-fact *weak*, meaning that the states of every strongly connected component (SCC) are either all accepting or all rejecting. Thus as with the DBAs in the LDBA construction, intersecting a NBA with a weak NBA is simpler compared to the general case and it suffices to use the classical product construction for automata on finite words. Again, this means it is not necessary to add additional information to the product states to track the progress of the Büchi acceptance conditions, e.g. multiple copies of the state space or a round-robin counter.

Furthermore, in the same manner as before we integrate several automata into the same structure $\mathcal{B}^1_X$ without making the different state sets distinct. Last, note how this definition deals with the issue arising from $\psi[X]_\nu = \mathbf{ff}$, meaning we can have a Ψ and a X such that $\mathbf{ff} \in \Psi[X]_\nu$. In that case we have no matching initial state in $\mathcal{B}_{X,Y}$, since $dnf(\psi_1 \wedge \cdots \wedge \mathbf{ff} \wedge \cdots \wedge \psi_n) = \emptyset$. Before we move on to the proof of the theorem, we illustrate the construction in the following example:

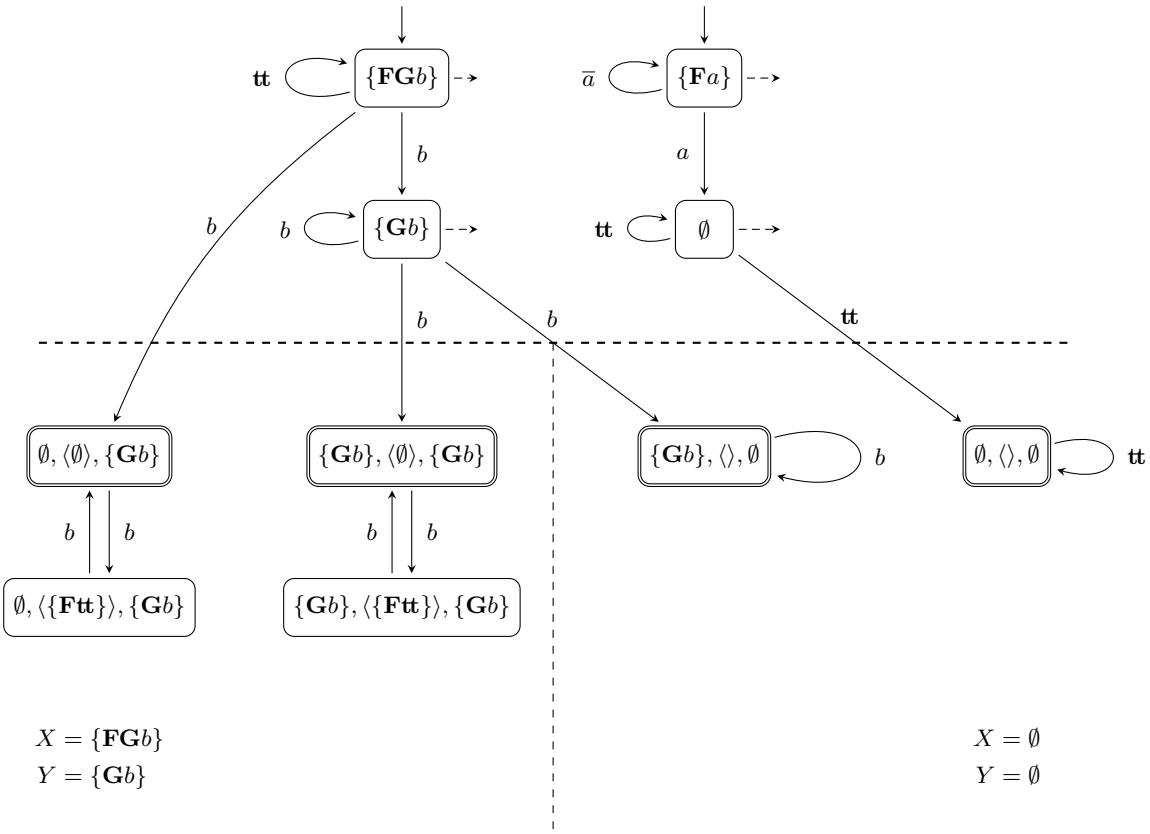


Figure 6.5: $\mathcal{A}_{\mathsf{NBA}}(\varphi)$ for $\varphi = \mathbf{F}a \vee \mathbf{FG}b$. Omitted states and transitions are indicated by dashed arrows.

Example 6.14. *Let us revisit the formula $\varphi = \mathbf{F}a \vee \mathbf{FG}b$ previously used in Example 6.3. The NBA $\mathcal{A}_{\mathsf{NBA}}(\varphi)$ constructed from Theorem 6.13 is depicted in Figure 6.5. Further, we use the following simplified versions of $\cdot[\cdot]_\nu$ and $\cdot[\cdot]_\mu$ to enhance readability:*

$$(\mathbf{F}\psi)[Y]_\nu = \begin{cases} \mathbf{tt} & \text{if } \mathbf{F}\psi \in Y \\ \mathbf{ff} & \text{otherwise.} \end{cases} \qquad (\mathbf{G}\psi)[X]_\mu = \begin{cases} \mathbf{tt} & \text{if } \mathbf{G}\psi \in X \\ \mathbf{ff} & \text{otherwise.} \end{cases}$$

Compared to the LDBA in Figure 6.1 the transition structure is simpler. We can apply similar optimisations as before, such as the removal of redundant components by replacing Theorem 4.14 with the restricted version of it: Proposition 4.18.

Proof of Theorem 6.13. Let φ be a formula, let $\mathcal{A}_{\mathsf{NBA}}(\varphi)$ be the corresponding automaton, and let w be a word. Further, we denote by $Q_{\Psi',X,Y} \subseteq Q_{0,X,Y}$ the initial states of $\mathcal{B}_{X,Y}$ with $\Psi' = \Psi[X]_\nu$ in the first component for every reachable state $\Psi \in \mathit{Reach}_\vee(\varphi)$ and for all sets of formulas $X \subseteq \mu(\varphi)$ and $Y \subseteq \nu(\varphi)$. We denote by $\mathcal{L}(Q_{\Psi',X,Y})$ the language accepted from the states $Q_{\Psi',X,Y}$ on the automaton $\mathcal{B}_{X,Y}$ and using Proposition 6.10 we can characterise the language $\mathcal{L}(Q_{\Psi',X,Y})$ in terms of LTL:

$$\begin{aligned} \forall v.\ v \in \mathcal{L}(Q_{\Psi',X,Y}) \iff & \forall \psi \in \Psi'.\ v \models \psi \\ & \wedge \forall \psi \in X.\ v \models \mathbf{GF}(\psi[Y]_\mu) \\ & \wedge \forall \psi \in Y.\ v \models \mathbf{G}(\psi[X]_\nu) \end{aligned}$$

($\Rightarrow$) Assume $w \models \varphi$. By Corollary 6.1 there exists i, X, and Y such that (1'-3') hold. We now construct an accepting run r on $\mathcal{A}_{\mathsf{NBA}}(\varphi)$ and thus show $w \in \mathcal{L}(\mathcal{A}_{\mathsf{NBA}}(\varphi))$. Let now $\mathcal{I} \coloneqq \{\psi \in sf(af(\varphi, w_{0i})[X]_\nu) : w_i \models \psi\}$ be a propositional assignment. From (1') and Lemma 2.7 we follow $\mathcal{I} \models_p af(\varphi, w_{0i})[X]_\nu$ and thus we can apply Lemma 6.9 to obtain a clause $\Psi \in af_\vee(\varphi, w_{0i})$ such that $\Psi[X]_\nu \subseteq \mathcal{I}$. Thus the run follows for the first i steps a matching path in the initial component to arrive at Ψ. Assume we have $w_i \in \mathcal{L}(Q_{\Psi',X,Y})$, then the run r branches off from the clause Ψ via $\delta_\frown$ to the accepting component $\mathcal{B}_{X,Y}$ and follows the accepting run belonging to w_i.

Thus it remains to show that right-hand side of the characterisation of $\mathcal{L}(Q_{\Psi',X,Y})$ holds for w_i. (2') and (3') match the second and third conjunct and we only need to prove $\forall \psi \in \Psi'.\, w_i \models \psi$. However, this follows immediately from the definition of $\mathcal{I}$ and the subset relation $\Psi' = \Psi[X]_\nu \subseteq \mathcal{I}$ and we are done.

($\Leftarrow$) Assume w is accepted by $\mathcal{A}_{\mathsf{NBA}}(\varphi)$. Then there exists an accepting run r. Let now i be the index at that the run transitions via $\delta_\frown$ to $\mathcal{B}_{X,Y}$ for some $\Psi \in af_\vee(\varphi, w_{0i})$, $X \subseteq \mu(\varphi)$, and $Y \subseteq \nu(\varphi)$. This has to happen, since every accepting state is in some accepting component $\mathcal{B}_{X,Y}$. Because the run r is accepting, w_i is in $\mathcal{L}(Q_{\Psi',X,Y})$ for $\Psi' = \Psi[X]_\nu$ and we obtain the following from the LTL characterisation:

$$\forall \psi \in \Psi[X]_\nu.\, w_i \models \psi \qquad w_i \models \bigwedge_{\psi \in X} \mathbf{GF}(\psi[Y]_\mu) \qquad w_i \models \bigwedge_{\psi \in Y} \mathbf{G}(\psi[X]_\nu)$$

From the second and third fact, we can immediately follow (2') and (3') of Corollary 6.1 and for showing $w \models \varphi$ we only need to prove the remaining (1'): $w_i \models af(\varphi, w_{0i})[X]_\nu$. Let now $\mathcal{I} \coloneqq \{\psi \in sf(af(\varphi, w_{0i})[X]_\nu) : w_i \models \psi\}$ be a propositional assignment. Observe that from the first fact we can follow $\Psi[X]_\nu \subseteq \mathcal{I}$. Then we apply Lemma 6.9 and get $\mathcal{I} \models_p af(\varphi, w_{0i})[X]_\nu$. Using Lemma 2.7 we derive (1') and we are done. □

Complexity Analysis. Let φ be a formula of length n. Since $\mathcal{A}_{\mathsf{NBA}}(\varphi)$ is constructed from unions and intersections of several components, the number of states can be bounded by the size of the components:

$$|Reach_\vee(\varphi)| + \sum_{\substack{X \subseteq \mu(\varphi) \\ Y \subseteq \nu(\varphi)}} (|Q^1_X| \cdot |Q^2_{X,Y}| \cdot |Q^3_{X,Y}|)$$

We bound the size of each of these components by an exponential function. $\mathcal{B}^1_X$ is a combination of exponentially many automata with size $O(2^n)$ and thus blindly applying the upper bounds of the preceding section gives a $O(2^n \cdot 2^n) = O(2^{2n})$ upper bound. However, observe that the number of proper subformulas of that formula is at most $|sf(\varphi)| \leq n$ and so Q^1_X has at most 2^n elements. $\mathcal{B}^2_{X,Y}$ is an intersection of $|X|$-many automata with at most 2^{n+1} states. We bound $|X|$ by $|\mu(\varphi)| \leq n$ and thus the number of states is at most $n \cdot (2^{n+1})^n = 2^{n^2+n+\log_2(n)}$ including the necessary round-robin counter. $\mathcal{B}^3_{X,Y}$ is constructed from the formula $\bigwedge_{\psi \in Y} \mathbf{G}(\psi[X]_\nu)$. This formula has at most $|sf(\varphi)| + |\nu(\varphi)|$ proper subformulas and thus the automaton has at most $2^{|sf(\varphi)|+|\nu(\varphi)|} \leq 2^{2n}$ states. Going back to our initial bound and inserting the upper bounds for the components we obtain:

$$2^n + 2^{|\mu(\varphi)|+|\nu(\varphi)|}(2^n \cdot 2^{n^2+n+\log_2(n)} \cdot 2^{2n}) \leq 2^n + 2^{n^2+5n+\log_2(n)} \leq 2^{n^2+6n+\log_2(n)} \in 2^{O(n^2)}$$

We will later refine the construction in Section 7.3.4 by replacing $\mathcal{B}^2_{X,Y}$ by a component of size $n \cdot 2^{n+1}$ and thus reduce the upper bound on the size to $O(2^{5n}) \subseteq 2^{O(n)}$.

Alternative Construction. Note that in [EKS18] the NBA construction is derived in the same way as the DRA construction. Here, we followed the LDBA construction, i.e. we did not replace components in the DRA construction by their nondeterministic counterparts, but we replaces the components of the LDBA construction. The advantage of basing the NBA construction on the LDBA construction is that nondeterministic guessing is centralised to a single point, which seems to be saving states in several scenarios.

6.A Omitted Proofs

Proof of Lemma 6.8

We first show how to obtain a satisfying propositional assignment for φ from a satisfying assignment for $af(\varphi, \sigma)$. Second, we establish Lemma 6.8 for a single letter σ in Lemma 6.16 and then generalise it to arbitrary words.

Lemma 6.15. *Let φ be a formula, let σ be a letter, and let $\mathcal{I}$ be a propositional assignment. Then:*

$$\mathcal{I} \models_p af(\varphi, \sigma) \iff \{\psi \in sf(\varphi) : \mathcal{I} \models_p af(\psi, \sigma)\} \models_p \varphi$$

Proof. The result follows from a straight-forward induction on φ. □

Lemma 6.16. *Let Φ be a set of formulas, let σ be a letter, and let $\mathcal{I}$ be a propositional assignment. Then:*

$$\forall \psi \in \Phi.\, \mathcal{I} \models_p af(\psi, \sigma) \iff \exists \Psi \subseteq \mathcal{I}.\, \Psi \in af_\vee(\Phi, \sigma)$$

Proof. ($\Leftarrow$) Assume there exists a subset $\Psi \subseteq \mathcal{I}$ of the propositional assignment $\mathcal{I}$ with $\Psi \in af_\vee(\Phi, \sigma)$. Further, let $\psi \in \Phi$ be an arbitrary element of the set Φ. We unfold the definition of $af_\vee$ and follow from the properties of $\otimes_{min}$ that there exists some $\Psi' \subseteq \Psi$ with $\Psi' \in dnf(af(\psi, \sigma))$ that contributed to Ψ for $\psi \in \Phi$. Since Ψ' is an element of the DNF we have $\Psi' \models_p af(\psi, \sigma)$ (Proposition 6.4) and due to the monotonicity of $\models_p$ we also have $\mathcal{I} \models_p af(\psi, \sigma)$.

($\Rightarrow$) We prove this direction by an induction on the size of Φ.

- Base $|\Phi| = 0$. Then the right-hand side evaluates to $af_\vee(\Phi, \sigma) = \{\emptyset\}$. Because the empty set is always a subset, this side also holds for all $\mathcal{I}$.

- Step $|\Phi| = n + 1$. Let Φ' be a set of size n with $\Phi = \Phi' \uplus \{\psi\}$. Assume now that $\mathcal{I} \models_p af(\psi, \sigma)$ for all formulas $\psi \in \Phi$. First, this also holds for the subset Φ' and applying the induction hypothesis to Φ' yields $\exists \Psi' \subseteq \mathcal{I}.\, \Psi' \in af_\vee(\Phi', \sigma)$. Let now Ψ' be such a set of formulas. Second, with Proposition 6.4 applied to ψ we obtain a set $\Psi'' \subseteq \mathcal{I}$ with $\Psi'' \in dnf(af(\psi, \sigma))$. Taking these two steps together we have $\Psi' \cup \Psi'' \in (af_\vee(\Phi', \sigma)) \otimes (af_\vee(\{\psi\}, \sigma))$. Hence there exits some subset $\Psi \subseteq \Psi' \cup \Psi''$ that is contained in $(af_\vee(\Phi', \sigma)) \otimes_{min} (af_\vee(\{\psi\}, \sigma)) = af_\vee(\Phi', \sigma)$ and we are done.

□

Lemma 6.8. *Let φ be a formula, let w be a finite word, and let $\mathcal{I}$ be a propositional assignment. Then:*

$$\mathcal{I} \models_p af(\varphi, w) \iff \exists \Psi \subseteq \mathcal{I}.\, \Psi \in af_\vee(\varphi, w)$$

Proof. We prove the statement by induction on the length of w.

- Base $w = \epsilon$. Simplifying the statement we see that we need to prove: $\mathcal{I} \models_p \varphi \iff \exists \Psi \subseteq \mathcal{I}.\, \Psi \in dnf(\varphi)$. This immediately follows from Proposition 6.4.

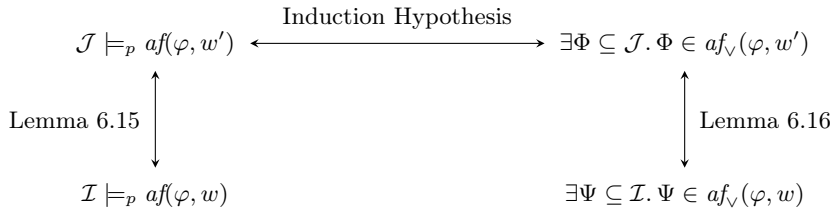


Figure 6.6: Induction step proof structure of Lemma 6.8

– Step $w = w'\sigma$. We prove both directions separately and the proofs follow the structure sketched in Figure 6.6. For both directions we define the propositional assignment $\mathcal{J} = \{\psi : \mathcal{I} \models_p af(\psi, \sigma)\}$.

($\Rightarrow$) Assume $\mathcal{I} \models_p af(\varphi, w)$ holds. Applying Lemma 6.15 gives us $\mathcal{J} \models_p af(\varphi, w')$ and with the induction hypothesis we obtain a set Φ such that $\Phi \subseteq \mathcal{J}$ and $\Phi \in af_\vee(\varphi, w')$. By definition of $\mathcal{J}$ we have $\mathcal{I} \models_p af(\psi, \sigma)$ for all $\psi \in \mathcal{J}$ and thus also all $\psi \in \Phi$. Hence we can apply Lemma 6.16 and have $\exists \Psi \subseteq \mathcal{I}.\, \Psi \in af_\vee(\Phi, \sigma)$. Since $\Phi \in af_\vee(\varphi, w')$, we have $af_\vee(\Phi, \sigma) \subseteq af_\vee(\varphi, w'\sigma)$ and are done.

($\Leftarrow$) Let Ψ and $\mathcal{I}$ be sets such that $\Psi \subseteq \mathcal{I}$ and $\mathcal{I} \in af_\vee(\varphi, w)$. Further let Φ be a set such that $\Phi \in af_\vee(\varphi, w')$ and $\Psi \in af_\vee(\Phi, \sigma)$. Assuming we proved $\Phi \subseteq \mathcal{J}$, we can simply apply the induction hypothesis and Lemma 6.15 to obtain at $\mathcal{I} \models_p af(\varphi, w)$. Thus it remains to show: $\Phi \subseteq \mathcal{J}$. Let ψ an arbitrary formula from the set Φ. In order to show the inclusion, we need to establish $\mathcal{I} \models_p af(\psi, \sigma)$, but this exactly follows from Lemma 6.16 and we are done.

□

Proof of Lemma 6.9

Lemma 6.9. *Let φ be a formula, let w be a finite word, let $X \subseteq \mu(\varphi)$ be a set of subformulas, and let $\mathcal{I} \subseteq sf(\varphi)$ be a propositional assignment that only contains proper subformulas. Then:*

1. $af_\vee(\varphi, w) \subseteq 2^{sf(\varphi)}$
2. $\emptyset \in af_\vee(\varphi, w) \iff af(\varphi, w) \sim_p \mathbf{tt}$
3. $af_\vee(\varphi, w) = \emptyset \iff af(\varphi, w) \sim_p \mathbf{ff}$
4. $\mathcal{L}(af(\varphi, w)) = \bigcup \left\{ \bigcap_{\psi \in \Psi} \mathcal{L}(\psi) : \Psi \in af_\vee(\varphi, w) \right\}$
5. $\mathcal{I} \models_p af(\varphi, w)[X]_\nu \iff \exists \Psi.\, \Psi[X]_\nu \subseteq \mathcal{I} \wedge \Psi \in af_\vee(\varphi, w)$

Proof. (1) This holds intuitively, because all involved steps either construct singleton sets from subformulas or combine existing sets without adding new literals or modal operators. Formally this is shown by an induction on the length of w and a structural induction on φ.

(2) We obtain this by two simple steps:

$$\begin{aligned} af(\varphi, w) \sim_p \mathbf{tt} &\iff \emptyset \models_p af(\varphi, w) \\ &\iff \emptyset \in af_\vee(\varphi, w) \quad \text{(Lemma 6.8)} \end{aligned}$$

(3) Let $\mathcal{U}$ be the universe, meaning it contains all formulas, and thus $\mathcal{U}$ sets every propositional variable to $\mathbf{tt}$. Then:

$$\begin{aligned}
af(\varphi, w) \sim_p \mathbf{ff} &\iff \mathcal{U} \not\models_p af(\varphi, w) && (\models_p \text{ is monotone}) \\
&\iff \forall \Psi \subseteq \mathcal{U}.\, \Psi \notin af_\vee(\varphi, w) && (\text{Lemma 6.8}) \\
&\iff \forall \Psi.\, \Psi \notin af_\vee(\varphi, w) && (\text{every set of formulas is a subset of } \mathcal{U}) \\
&\iff af_\vee(\varphi, w) = \emptyset
\end{aligned}$$

(4) Let w' be an arbitrary word and let $\mathcal{I} = \{\psi : w' \models \psi\}$ be a propositional assignment. We then show (4) by proving the following equivalence:

$$\begin{aligned}
w' \in \mathcal{L}(af(\varphi, w)) &\iff w' \models af(\varphi, w) && (\text{Lemma 3.3}) \\
&\iff \mathcal{I} \models_p af(\varphi, w) && (\text{Lemma 2.7}) \\
&\iff \exists \Psi.\, \Psi \in af_\vee(\varphi, w) \wedge \Psi \subseteq \mathcal{I} && (\text{Lemma 6.8}) \\
&\iff \exists \Psi.\, \Psi \in af_\vee(\varphi, w) \wedge \forall \psi \in \Psi.\, w' \models \psi \\
&\iff w' \in \bigcup \left\{ \bigcap\nolimits_{\psi \in \Psi} \mathcal{L}(\psi) : \Psi \in af_\vee(\varphi, w) \right\}
\end{aligned}$$

(5) We base our proof on following claim:
Let χ be a formula. Then:

$$\mathcal{I} \models_p \chi[X]_\nu \iff \{\psi : \psi[X]_\nu \in \mathcal{I}\} \models_p \chi$$

Proof of the claim. Intuitively this holds, because in the propositional view on LTL we can apply on both sides of $\models_p$ – meaning on the formula and point-wise on the propositional assignment – a propositional substitution preserving the truth of $\models_p$ as long as we do not substitute with $\mathbf{ff}$. Further $\cdot[\cdot]_\nu$ is a substitution replacing proper subformulas by either $\mathbf{ff}$ or other proper subformulas, but since $\mathbf{ff} \notin \mathcal{I}$ we never replace by $\mathbf{ff}$. Formally, this is proven by a straight-forward, structural induction on χ.

($\Rightarrow_5$) Assume $\mathcal{I} \models_p af(\varphi, w)[X]_\nu$. Hence the assignment $\mathcal{J} = \{\psi : \psi[X]_\nu \in \mathcal{I}\}$ is a satisfying assignment of $af(\varphi, w)$, i.e. $\mathcal{J} \models_p af(\varphi, w)$, which follows from the previous claim. By Lemma 6.8 the exists some Ψ such that $\Psi \subseteq \mathcal{J}$ and $\Psi \in af_\vee(\varphi, w)$. Thus $\Psi[X]_\nu \subseteq \mathcal{I}$ and we are done.

($\Leftarrow_5$) Assume $\Psi[X]_\nu \subseteq \mathcal{I}$ and $\Psi \in af_\vee(\varphi, w)$. Further, let $\mathcal{J} = \{\psi : \psi[X]_\nu \in \mathcal{I}\}$. Then $\Psi \subseteq \mathcal{J}$ and we can apply the other direction of Lemma 6.8 to get $\mathcal{J} \models_p af(\varphi, w)$. Applying the claim we obtain $\mathcal{I} \models_p af(\varphi, w)[X]_\nu$ and we are done. □

7 Optimisations of the Constructions

In Chapters 5 and 6 we focussed on the essential ideas and favoured simplicity over optimality. Consequently, we missed several opportunities to obtain smaller automata by using technically-involved constructions. We now discuss techniques that apply to all three constructions, namely: the removal of redundant components using Proposition 4.18, transition-based acceptance that removes 'trivial' states, i.e. states that represent $\mathbf{tt}$ and $\mathbf{ff}$, and specialised intersection constructions for $\mathcal{B}^2_{X,Y}$ and $\mathcal{C}^3_{X,Y}$. We follow this then by improvements specific to a subset of constructions: an augmented version of propositional equivalence and various minor state space optimisations. Keep in mind that, while we do not explicitly mention it in the following section, all of the presented optimisations can be combined.

7.1 Restricted Guessing

A crucial weakness of the presented constructions lies in the need to check exponentially many guesses for X and Y. The reduction of these guesses is paramount to obtaining small automata. We reduce them by using the restricted version of the Master Theorem (Proposition 4.18) as a basis, i.e. X and Y are now chosen from a smaller domain $X \subseteq \mu(\varphi) \cap \Downarrow(\varphi)$ and $Y \subseteq \nu(\varphi) \cap \Downarrow(\varphi)$, respectively. Especially, for formulas with a flat structure, i.e. that only have at most one alternation between greatest- or least-fixed-point operators, this reduces the possible guesses for X and Y considerably. An NBA with such a reduction is depicted in Figure 7.5.

7.2 Transition-Based Acceptance

7.2.1 Deterministic Automata

The constructions for $\mathbf{GF}(\mu LTL)$ and $\mathbf{FG}(\nu LTL)$ defined in Proposition 5.1 use the states $[\mathbf{tt}]_\sim$ and $[\mathbf{ff}]_\sim$ to signal success and failure while checking a LTL formula. The acceptance condition then uses these states for the Büchi and co-Büchi condition as accepting and rejecting state, respectively. From these states runs always return to the initial state effectively dropping one letter of the word. When we use a state-based acceptance condition, these states in general cannot be avoided, but with a transition-based acceptance condition we are able to skip and remove these states. Similarly, we can apply this to automata obtained from Proposition 5.6, where states of the shape $\langle [\psi]_\sim, [\mathbf{ff}]_\sim \rangle$ are used to signal failure before resetting. After the augmentation of all components of the DRA with transition-based acceptance conditions we consequently obtain a DRA construction for all LTL formulas with transition-based acceptance condition. While these savings might seem minor at first sight – we just remove a single state after all – the final DRA is a product automaton and these savings accumulate, e.g. shrinking

a two-state automaton to a single-state component can effectively halve the size of the product. Formally[1]:

Proposition 7.1. *Let $\sim$ be an af-congruence.*

- *Let $\varphi \in \mu LTL$. Then the following DBA over the alphabet 2^{Ap} recognises $\mathcal{L}(\mathbf{GF}\varphi)$:*

$$\mathcal{B}^{\varphi}_{\mathbf{GF}\mu} = (\mathit{Reach}(\mathbf{F}\varphi)_{/\sim}, \mathit{af}^{\mathbf{F}\varphi}_{\sim}, \mathit{af}^{\mathbf{F}\varphi}_{\sim}([\mathbf{F}\varphi]_{\sim}, \emptyset), \mathit{inf}(\alpha))$$

  where the transition function $\mathit{af}^{\mathbf{F}\varphi}_{\sim}$ and the accepting transitions α are defined as:

$$\mathit{af}^{\mathbf{F}\varphi}_{\sim}([\psi]_{\sim}, \sigma) = \begin{cases} [\mathbf{F}\varphi]_{\sim} & \text{if } \mathit{af}(\psi, \sigma) \sim \mathbf{tt} \text{ and } \mathit{af}(\mathbf{F}\varphi, \sigma) \sim \mathbf{tt} \\ [\mathit{af}(\mathbf{F}\varphi, \sigma)]_{\sim} & \text{otherwise if } \mathit{af}(\psi, \sigma) \sim \mathbf{tt} \\ [\mathit{af}(\psi, \sigma)]_{\sim} & \text{otherwise.} \end{cases}$$

$$\alpha = \{([\psi]_{\sim}, \sigma, [\chi]_{\sim}) \in \mathit{af}^{\mathbf{F}\varphi}_{\sim} : \mathit{af}(\psi, \sigma) \sim \mathbf{tt}\}$$

- *Let $\varphi \in \nu LTL$. Then the following DCA over the alphabet 2^{Ap} recognises $\mathcal{L}(\mathbf{FG}\varphi)$:*

$$\mathcal{C}^{\varphi}_{\mathbf{FG}\nu} = (\mathit{Reach}(\mathbf{G}\varphi)_{/\sim}, \mathit{af}^{\mathbf{G}\varphi}_{\sim}, \mathit{af}^{\mathbf{G}\varphi}_{\sim}([\mathbf{G}\varphi]_{\sim}, \emptyset), \mathit{fin}([\mathbf{ff}]_{\sim}))$$

  where the transition function $\mathit{af}^{\mathbf{G}\varphi}_{\sim}$ and the rejecting transitions β are defined as:

$$\mathit{af}^{\mathbf{G}\varphi}_{\sim}(\psi, \sigma) = \begin{cases} [\mathbf{G}\varphi]_{\sim} & \text{if } \mathit{af}(\psi, \sigma) \sim \mathbf{ff} \text{ and } \mathit{af}(\mathbf{G}\varphi, \sigma) \sim \mathbf{ff} \\ [\mathit{af}(\mathbf{G}\varphi, \sigma)]_{\sim} & \text{otherwise if } \mathit{af}(\psi, \sigma) \sim \mathbf{ff} \\ [\mathit{af}(\psi, \sigma)]_{\sim} & \text{otherwise.} \end{cases}$$

$$\beta = \{([\psi]_{\sim}, \sigma, [\chi]_{\sim}) \in \mathit{af}^{\mathbf{G}\varphi}_{\sim} : \mathit{af}(\psi, \sigma) \sim \mathbf{ff}\}$$

Proof sketch. The proof is analogous to the proof of Proposition 5.1 with index tweaks and a minor change: the automata defined in Proposition 7.1 implicitly read an $\emptyset$ to compute the initial state. However, this does not change the recognised language, since $w \models \mathbf{GF}\varphi \longleftrightarrow \sigma w \models \mathbf{GF}\varphi$ and $w \models \mathbf{FG}\varphi \longleftrightarrow \sigma w \models \mathbf{FG}\varphi$. □

Besides removing $[\mathbf{tt}]_{\sim}$ and $[\mathbf{ff}]_{\sim}$ we avoid the construction of a transient initial state by implicitly reading the letter $\emptyset$ in the beginning. In the following example we effectively remove[2] with this technique two states from the component.

Example 7.2. *Let us consider the two short formulas $\varphi_1 = a$ and $\varphi_2 = (a \leftrightarrow \mathbf{X}a)$ where $\psi \leftrightarrow \chi$ is an abbreviation for $(\psi \wedge \chi) \vee (\neg\psi \wedge \neg\chi)$. The result of applying Proposition 5.1 and Proposition 7.1 with $\sim_p$ as af-congruence to φ_1 and φ_2 is depicted in Figure 7.1. Notice that not only $[\mathbf{tt}]_{\sim}$ is removed in Figure 7.1d, but also the state $[\mathbf{F}\varphi_2]_{\sim}$.*

Let us now proceed to Proposition 5.6 and revise the construction in the same way:

Proposition 7.3. *Let $\sim$ be an af- and $\cdot[\cdot]_{\nu}$-congruence, let φ be a formula, and let X be a set of formulas. Then the following DCA over the alphabet 2^{Ap} recognises exactly the words w such that $\exists i.\, w_i \models \mathit{af}(\varphi, w_{0i})[X]_{\nu}$ holds:*

$$\mathcal{C}^{1}_{\varphi,X} = (Q, \delta, \langle [\varphi]_{\sim}, [\varphi[X]_{\nu}]_{\sim} \rangle, \mathit{fin}(\alpha))$$

where we define the states Q, the transition function δ, and the rejecting transitions α in the following way:

[1] Acknowledgment: Proposition 7.1 has been revised after a discussion with Alexandre Duret-Lutz.
[2] Technically we do not remove states, but make them unreachable from the initial state.

– $Q = \mathit{Reach}(\varphi)_{/\sim} \times \bigcup_{\psi \in \mathit{Reach}(\varphi)} \mathit{Reach}(\psi[X]_\nu)_{/\sim}$. *That is, a state is a tuple of equivalence classes $[\cdot]_\sim$, where the second tracks the formula after applying $\cdot[\cdot]_\nu$.*

–
$$\delta(\langle [\xi]_\sim, [\zeta]_\sim \rangle, \sigma) = \begin{cases} \langle [\mathit{af}(\xi,\sigma)]_\sim, [\mathit{af}(\xi,\sigma)[X]_\nu]_\sim \rangle & \text{if } \mathit{af}(\zeta,\sigma) \sim \mathbf{ff} \\ \langle [\mathit{af}(\xi,\sigma)]_\sim, [\mathit{af}(\zeta,\sigma)]_\sim \rangle & \text{otherwise.} \end{cases}$$
That is, a transition either resets a failed attempt to prove $w_i \models \mathit{af}(\varphi, w_{0i})[X]_\nu$ for some i or continues unfolding both equivalence classes using $\mathit{af}_\sim$.

– $\alpha = \{(\langle [\xi]_\sim, [\zeta]_\sim \rangle, \sigma, \langle [\xi']_\sim, [\zeta']_\sim \rangle) \in \delta : \mathit{af}(\zeta,\sigma) \sim \mathbf{ff}\}$.

Proof sketch. The proof is analogous to the proof of Proposition 5.6 with minor index tweaks. □

Example 7.4. *Let us now revisit Example 5.7 where we picked $\varphi = \mathbf{G}(a\mathbf{U}b \vee \mathbf{F}c)$, $X = \{a\mathbf{U}b\}$ and let us apply Proposition 7.3 to it. In comparison to the automaton depicted in Figure 5.4 this construction can save one state as one can see in Figure 7.2.*

We finally can bring now the pieces together and reexamine Example 5.9. The DRA $\mathcal{R}_{\varphi,X,\emptyset}$ with transition-acceptance is shown in Figure 7.3. Here one can see how these little savings accumulate: the automaton depicted in Figure 7.3 using transition-acceptance has two states, while the automaton depicted in Figure 5.5 has five states.

7.2.2 Nondeterministic Automata

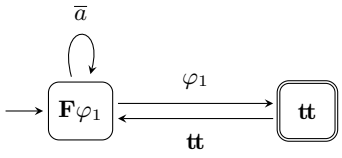
As in the case for DRAs we are able to remove $\emptyset$ and transient initial states for automata built for $\mathbf{GF}\varphi$ with $\varphi \in \mu LTL$ using a transition-based acceptance condition. To be more precise we use the following revised construction, illustrated in Figure 7.4 and Figure 7.5:

Proposition 7.5. *Let $\varphi \in \mu LTL$. The following NBA over the alphabet 2^{Ap} recognises $\mathcal{L}(\mathbf{GF}\varphi)$:*
$$\mathcal{B}^{\varphi}_{\mathbf{GF}\mu} = (\ \mathit{Reach}_\vee(\mathbf{F}\varphi), \mathit{af}_\vee^{\mathbf{F}\varphi}, \mathit{af}_\vee^{\mathbf{F}\varphi}(\{\mathbf{F}\varphi\}, \emptyset), \mathit{inf}(\alpha)\ )$$
where the transition relation $\mathit{af}_\vee^{\mathbf{F}\varphi}$ and the set of accepting transitions α are defined as:

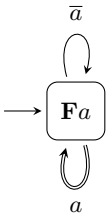
– $\mathit{af}_\vee^{\mathbf{F}\varphi}(\Psi,\sigma) = \begin{cases} \{\{\mathbf{F}\varphi\}\} & \text{if } \emptyset \in \mathit{af}_\vee(\Psi,\sigma) \text{ and } \emptyset \in \mathit{af}_\vee(\{\mathbf{F}\varphi\},\sigma) \\ \mathit{af}_\vee(\{\mathbf{F}\varphi\},\sigma) & \text{otherwise if } \emptyset \in \mathit{af}_\vee(\Psi,\sigma) \\ \mathit{af}_\vee(\Psi,\sigma) & \text{otherwise.} \end{cases}$

– $\alpha = \{(\Psi,\sigma,\Psi') \in \mathit{af}_\vee^{\mathbf{F}\varphi} : \emptyset \in \mathit{af}_\vee(\Psi,\sigma)\}$

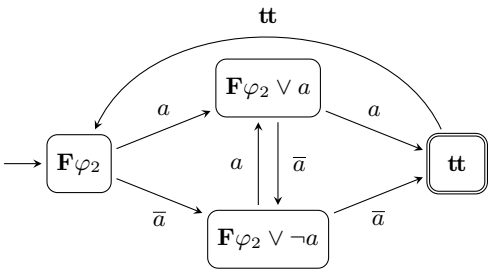
Proof sketch. The proof is analogous to the proof of Proposition 6.10 with minor index tweaks and a small change: the automaton defined in Proposition 7.5 implicitly reads an $\emptyset$ to compute the initial state. However, this does not change the recognised language, since $w \models \mathbf{GF}\varphi \longleftrightarrow \sigma w \models \mathbf{GF}\varphi$. □



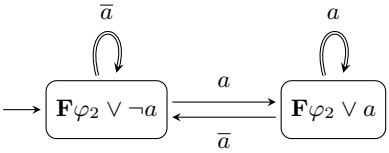
(a) DBA $\mathcal{B}^{\varphi_1}_{\mathbf{GF}\mu}$ (state-based).



(b) DBA $\mathcal{B}^{\varphi_1}_{\mathbf{GF}\mu}$ (transition-based).



(c) DBA $\mathcal{B}^{\varphi_2}_{\mathbf{GF}\mu}$ (state-based).



(d) DBA $\mathcal{B}^{\varphi_2}_{\mathbf{GF}\mu}$ (transition-based).

Figure 7.1: Comparison of automata from Proposition 5.1 (state-based) and from Proposition 7.1 (transition-based) using $\sim_p$ as *af*-congruence.

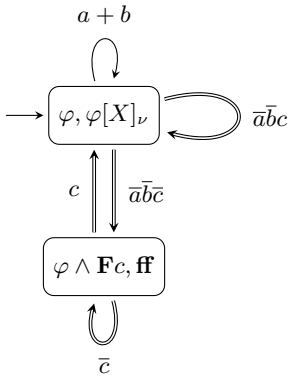


Figure 7.2: DCA $\mathcal{C}^1_{\varphi,X}$ from Proposition 7.3 for $\varphi = \mathbf{G}(a\mathbf{U}b \vee \mathbf{F}c)$ and $X = \{a\mathbf{U}b\}$ using the yet-to-be-defined $\sim_q$.

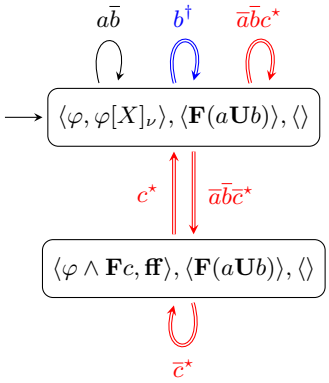


Figure 7.3: DRA $\mathcal{R}_{\varphi,X,Y}$ for $\varphi = \mathbf{G}(a\mathbf{U}b \vee \mathbf{F}c)$, $X = \{a\mathbf{U}b\}$, and $Y = \emptyset$ using the yet-to-be-defined $\sim_q$ based on constructions with transition-acceptance. The DRA $\mathcal{R}_{\varphi,X,Y}$ has only one Rabin pair: $\mathit{fin}(\star) \wedge \mathit{inf}(\dagger)$.

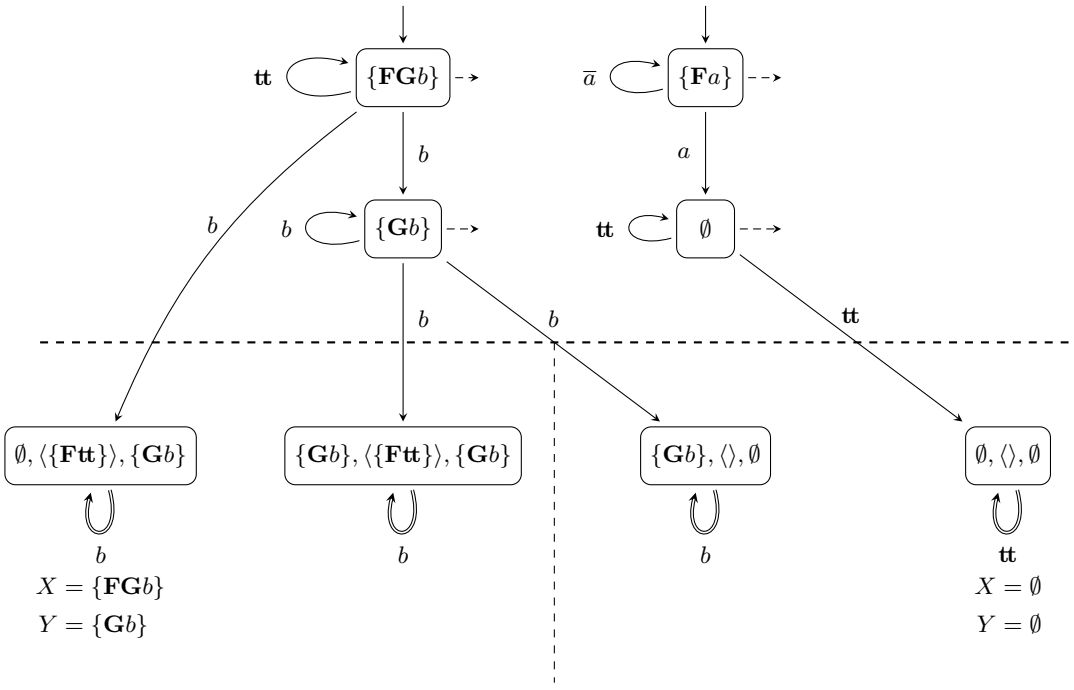


Figure 7.4: $\mathcal{A}_{\mathsf{NBA}}(\varphi)$ for $\varphi = \mathbf{F}a \vee \mathbf{FG}b$ built from components with transition acceptance (Proposition 7.5). Omitted states and transitions are indicated by dashed arrows.

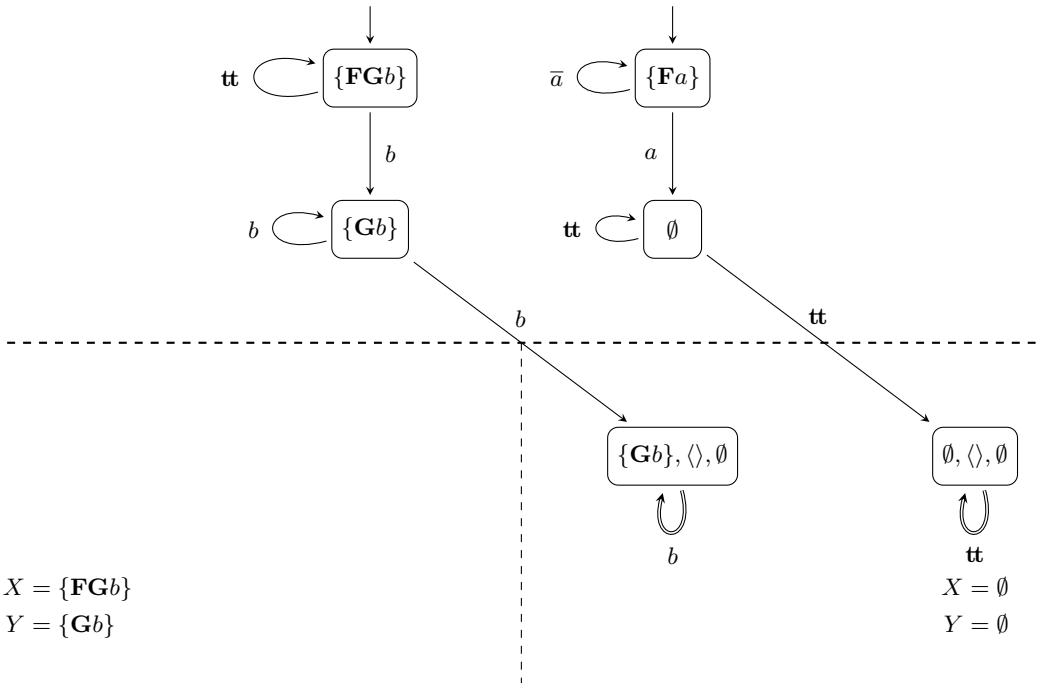


Figure 7.5: $\mathcal{A}_{\mathsf{NBA}}(\varphi)$ for $\varphi = \mathbf{F}a \vee \mathbf{FG}b$ built from components with transition acceptance (Proposition 7.5). Further the number of accepting components is reduced by applying the restrictions of Proposition 4.18.

7.3 Specialised Intersection Constructions

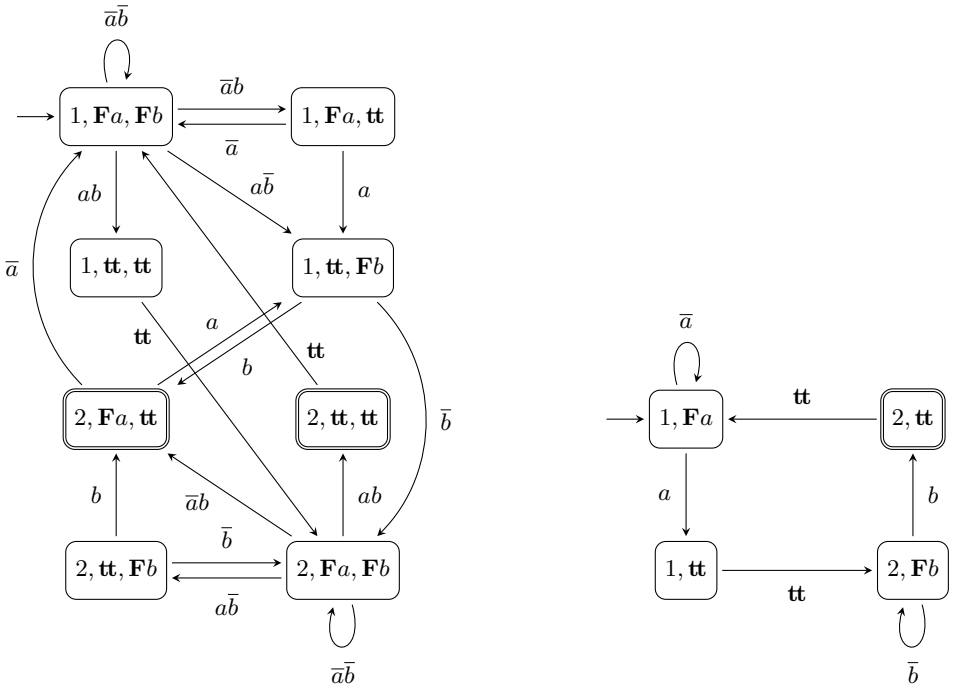
In the presented constructions (Theorems 5.8, 6.2 and 6.13) we rely several times on intersections to obtain automata $\mathcal{B}^2_{X,Y}$ for checking (2) and $\mathcal{C}^3_{X,Y}$ for checking (3) of Theorem 4.14:

$$\mathcal{B}^2_{X,Y} = \bigcap_{\psi \in X} \mathcal{B}^{\psi[Y]_\mu}_{\mathbf{GF}\mu} \qquad \mathcal{C}^3_{X,Y} = \bigcap_{\psi \in Y} \mathcal{C}^{\psi[X]_\nu}_{\mathbf{FG}\nu}$$

In these cases we simply resorted to the general-purpose intersection constructions for Büchi and co-Büchi automata without taking the special structure of the involved components into account. In fact there are several, partly folklore, ways to specialise this intersection to reduce the size of the resulting automata. We will now consider three different approaches to reduce the number of states and conclude this section with updated upper bounds on the constructed NBAs and DRAs.

7.3.1 Generalised Büchi Acceptance

When we intersect Büchi automata, usually one needs to have bookkeeping in the states, e.g. a round-robin counter, additionally to the product construction in order to track which Büchi automaton is the next that owes a visit to the accepting states. This added bookkeeping can be elided by moving this requirement to the acceptance condition, called generalised Büchi acceptance, where a run is accepting if at least one state (or transition) of each Büchi set is visited infinitely often.



(a) General-purpose Büchi intersection. **(b)** Interleaved Büchi intersection.

Figure 7.6: Intersection constructions for DBA $\mathcal{B}^2_{X,\emptyset}$ with the set of formulas $X = \{\mathbf{F}a, \mathbf{F}b\}$ and using Proposition 5.1 with $\sim_l$ as *af*-congruence.

In the deterministic setting we obtain an intersection DBA for a set of formulas such as $X = \{\mathbf{F}a_1, \mathbf{F}a_2, \ldots, \mathbf{F}a_i\}$ that has only a single state, assuming we use a transition-based acceptance construction. This propagates up the construction hierarchy and the final product automaton has a generalised Rabin acceptance condition. Analogously, we get for the other two constructions (NBA, LDBA) generalised Büchi automata.

7.3.2 Interleaving

Observe that $\mathcal{B}^2_{X,Y}$ recognises an intersection of languages $\mathcal{L}(\mathbf{GF}\psi)$ and the corresponding formulas are satisfied by a word w if and only if there are infinitely many suffixes w_i that satisfy ψ. However, it is not necessary to identify all suffixes for proving satisfaction, it is enough to identify an infinite subset. Thus we can serialise checking the formulas $\mathbf{GF}\psi_1, \ldots \mathbf{GF}\psi_n$ and whenever the current automaton visits an accepting state, we suspend it and move to the next one in a round-robin fashion. This effectively reduces the maximal size of $\mathcal{B}^2_{X,Y}$ from $k \cdot \prod_{i=1}^{k} |Q_i|$ to $\sum_{i=1}^{k} |Q_i|$ in the deterministic and nondeterministic case. Formally, we define:

Proposition 7.6. *Let $X = \{\varphi_1, \varphi_2, \ldots, \varphi_k\}$ be a finite set of μLTL formulas and let $\sim$ be an af-congruence. The following DBA over the alphabet 2^{Ap} recognises $\bigcap_{i=1}^{k} \mathcal{L}(\mathbf{GF}\varphi_i)$:*

$$\mathcal{B}^X_{\mathbf{GF}\mu} = (Q, af_{\sim}^{\mathbf{F}X}, \langle 1, [\mathbf{F}\varphi_1]_{\sim} \rangle, \mathit{inf}(\langle k, [\mathbf{tt}]_{\sim} \rangle))$$

where the states Q and the transition function $af_{\sim}^{\mathbf{F}X}$ are defined in the following way:

- $Q = \bigcup_{i=1}^{k} \left(\{i\} \times \mathit{Reach}(\mathbf{F}\varphi_i)_{/\sim}\right)$. *That is the states are a tagged union of the reachable states of the components.*
- $af_{\sim}^{\mathbf{F}X}(\langle i, [\psi]_{\sim}\rangle, \sigma) = \begin{cases} \langle (i \bmod k) + 1, [\mathbf{F}\varphi_{(i \bmod k)+1}]_{\sim}\rangle & \text{if } \psi \sim \mathbf{tt} \\ \langle i, [af(\psi, \sigma)]_{\sim}\rangle & \text{otherwise.} \end{cases}$

Proof sketch. The proof is analogous to the proof of Proposition 5.1, but we need to additionally take care of the interleaving. Observe that, if $w \models \mathbf{GF}\varphi_k$ all suffixes w_i satisfy $w_i \models \mathbf{F}\varphi_k$. Thus an automaton in 'suspension' might miss the next point where φ_k might hold, but there are infinitely more to come and that automaton is eventually re-enabled. On the other hand, if $w \not\models \mathbf{GF}\varphi_k$ holds for some k, one of the components eventually gets stuck and the whole automaton is never reaching $\langle k, [\mathbf{tt}]_{\sim}\rangle$ again. □

Example 7.7. *Let $X = \{\mathbf{F}a, \mathbf{F}b\}$ be a set of μLTL formulas. Using the general-purpose Büchi intersection in combination with Proposition 5.1 to construct $\mathcal{B}^2_{X,\emptyset}$ we arrive at the automaton depicted in Figure 7.6a. However, when we switch to the construction of Proposition 7.6 we construct the simpler and smaller automaton shown in Figure 7.6b. Observe that we could further remove two more states by using transition acceptance (Section 7.2.1).*

Finally, we apply the same idea also to the NBA intersection:

Proposition 7.8. *Let $X = \{\varphi_1, \varphi_2, \ldots \varphi_k\}$ be a set of μLTL formulas. The following NBA over the alphabet 2^{Ap} recognises $\bigcap_{i=1}^{k} \mathcal{L}(\mathbf{GF}\varphi_i)$:*

$$\mathcal{B}^X_{\mathbf{GF}\mu} = (\, Q, af_{\vee}^{\mathbf{F}X}, \langle 1, \{\mathbf{F}\varphi_1\}\rangle, \mathit{inf}(\langle k, \emptyset\rangle)\,)$$

where the states Q and the transition relation $af_{\vee}^{\mathbf{F}X}$ are defined in the following way:

- $Q = \bigcup_{i=1}^{k} (\{i\} \times \mathit{Reach}_{\vee}(\mathbf{F}\varphi_i))$
- $af_{\vee}^{\mathbf{F}X}(\langle i, \Psi\rangle, \sigma) = \begin{cases} \{\langle (i \bmod k) + 1, \{\mathbf{F}\varphi_{(i \bmod k)+1}\}\rangle\} & \text{if } \Psi = \emptyset \\ \{i\} \times af_{\vee}(\Psi, \sigma) & \text{otherwise.} \end{cases}$

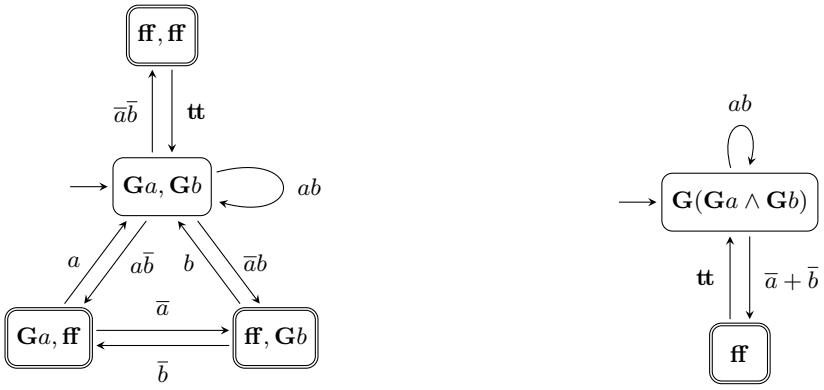
Proof sketch. The proof follows the idea as Proposition 7.6 and is analogous to the proof of Proposition 6.10. □

7.3.3 Formula Rewriting

The last alternative to the intersection is to work on the logical level instead of the automaton level. We can construct $\mathcal{B}^3_{X,Y}$ (Theorem 6.13) and $\mathcal{C}^3_{X,Y}$ (Theorem 5.8) in one-step without an intermediate intersection using the two LTL equivalences:

$$\bigwedge_{i=1}^{k} \mathbf{G}\psi_i \sim_l \mathbf{G}\left(\bigwedge_{i=1}^{k} \psi_i\right) \qquad \bigwedge_{\psi \in Y} \mathbf{FG}(\psi[X]_{\nu}) \sim_l \mathbf{FG}\left(\bigwedge_{\psi \in Y} \psi[X]_{\nu}\right)$$

The resulting formulas have at most $n + 1$ proper subformulas. Thus, the NBA $\mathcal{B}^3_{X,Y}$ obtained through Proposition 6.10 has at most 2^{n+1} states and the DCA $\mathcal{C}^3_{X,Y}$ obtained through Proposition 5.1 has at most $2^{2^{n+2}}$ states.



(a) General-purpose co-Büchi intersection. **(b)** Formula Rewriting.

Figure 7.7: Intersection constructions for DCA $\mathcal{C}^3_{\emptyset,Y}$ with the set of formulas $Y = \{\mathbf{G}a, \mathbf{G}b\}$ and using Proposition 5.1 with $\sim_l$ as *af*-congruence.

Example 7.9. *Let $Y = \{\mathbf{G}a, \mathbf{G}b\}$ be a set of νLTL formulas. On the on hand, we obtain with the general-purpose DCA intersection construction the automaton depicted in Figure 7.7a. On the other hand, we construct with Proposition 5.1 for the rewritten formula $\mathbf{FG}(\mathbf{G}a \wedge \mathbf{G}\wedge)$ the simpler and smaller automaton shown in Figure 7.7b.*

7.3.4 Complexity Analysis

A careful analysis shows that the intersection construction using a generalised Büchi acceptance condition is a double-edged sword: For some instances choosing this intersection construction in combination with a transition-based acceptance condition is clearly the best option: the set $F_n = \{\mathbf{F}a_1, \mathbf{F}a_2, \ldots, \mathbf{F}a_n\}$ is translated to a single-state generalised Büchi automaton with n Büchi sets, while the interleaving intersection obtains an automaton with n states and a single Büchi set. However, for the set $F'_n = \{\mathbf{F}(a_1 \wedge \mathbf{XX}b_1), \ldots, \mathbf{F}(a_n \wedge \mathbf{XX}b_n)\}$ the interleaving construction yields an automata with a polynomial number of states in n, while the intersection using generalised acceptance conditions is growing exponentially in n. In order to have the best of both worlds an implementation could choose heuristically one of the approaches trying to avoid 'bad' cases. Thus let us focus on the other two approaches in the revised complexity analysis:

DRA Construction. According to the preceding analysis the recommended choice from the upper-bound perspective is to use the interleaving intersection construction for $\mathcal{B}^2_{X,Y}$ and the formula rewriting approach for $\mathcal{C}^3_{X,Y}$. Let n be the length of formula φ and let $\sim_p$ be again the underlying equivalence relation. These changes then amount to a construction for the DRA $\mathcal{R}_{\varphi,X,Y}$ that uses has at most

$$2^{2^{n+1}} \cdot n \cdot 2^{2^{n+1}} \cdot 2^{2^{n+2}} = n \cdot 2^{2^{n+3}} = 2^{2^{n+(\log_2 \log_2 n)+3}}$$

states and one Rabin pair. Taking the union of all these yields a DRA $\mathcal{A}_{\mathsf{DRA}}(\varphi)$ with at most

$$\left(2^{2^{n+(\log_2 \log_2 n)+3}}\right)^{2^n} = 2^{2^n \cdot 2^{n+(\log_2 \log_2 n)+3}} = 2^{2^{2n+(\log_2 \log_2 n)+3}} \in 2^{2^{O(n)}}$$

states and at most 2^n Rabin pairs. While this did not change the upper bound by much, in practice these changes have a considerable effect.

NBA Construction. As in the DRA case the recommended choice is to use the interleaved intersection for construction $\mathcal{B}^2_{X,Y}$ and the formula rewriting approach for $\mathcal{B}^3_{X,Y}$. This yields the following new upper bound for a sufficiently large n, i.e. $n \geq 4$:

$$2^n + 2^{|\mu(\varphi)|+|\nu(\varphi)|}(2^n \cdot 2^{n+\log_2 n+1} \cdot 2^{n+1}) \leq 2^n + 2^{4n+\log_2 n+2} \leq 2^{5n} \in 2^{O(n)}$$

7.4 Augmented Propositional Equivalence

In Section 3.2 we have seen the three *af*-congruences, namely: d $\sim_c$, $\sim_p$, and $\sim_l$. While $\sim_c$ is also an $\cdot[\cdot]_\nu$-congruence, it is not suitable for constructing automata, since $Reach(\varphi)_{/\sim_c}$ might be infinite. At the other end of the spectrum we have a similar situation: $\sim_l$ generates a finite state space, but it is expensive to compute. More importantly: it is not an $\cdot[\cdot]_\nu$-congruence and thus cannot be used for constructing $\mathcal{C}^1_{\varphi,X}$. Only $\sim_p$ is an *af*- and $\cdot[\cdot]_\nu$-congruence, produces a finite state space, and is relatively cheap to compute. We will now define an augmented version of $\sim_p$ called[3] $\sim_q$ with the same properties but closer to LTL equivalence, i.e. $\sim_p \prec \sim_q \prec \sim_l$ holds. In fact we already used $\sim_q$ in some of the previous examples to obtain automata that are small enough for non-trivial formulas to be depicted with a few states. (Examples 5.7, 5.9 and 7.4).

Let us consider the following examples, where we wish that equivalence would hold, but does not for $\sim_p$: $a \vee \neg a \nsim_p \mathbf{tt}$ and $\mathbf{G}(a\mathbf{U}b) \nsim_p \mathbf{G}(a\mathbf{U}b) \wedge a\mathbf{U}b$. In the first example the issue is that $\sim_p$ does not relate proper formulas and their negations. While we cannot have this for all proper formulas without sacrificing $\cdot[\cdot]_\nu$-congruence, we can safely add it for literals. This motivates a revised propositional encoding ($\models_{p'}$), where all modal operators ($\mathbf{X}, \mathbf{U}, \mathbf{W}, \mathbf{R}, \mathbf{M}$) are encoded as distinct propositions x_ψ, but literals a and $\neg a$ are expressed using the same variable x_a and $\neg x_a$ respectively. In the second example $\sim_p$ is oblivious to LTL expansion rules. To address we expand all formulas by using the normalisation procedure $\mathfrak{Unf}$ from [EKS16].

Proposition 7.10. *Let $\mathcal{I}$ be a set of formulas and let φ be a formula. The satisfaction relation $\mathcal{I} \models_{p'} \varphi$ is inductively defined as:*

$\mathcal{I} \models_{p'} \mathbf{tt}$			$\mathcal{I} \models_{p'} \mathbf{X}\varphi$	*iff*	$\mathbf{X}\varphi \in \mathcal{I}$
$\mathcal{I} \not\models_{p'} \mathbf{ff}$					
$\mathcal{I} \models_{p'} a$	*iff*	$a \in \mathcal{I}$	$\mathcal{I} \models_{p'} \varphi\mathbf{U}\psi$	*iff*	$\varphi\mathbf{U}\psi \in \mathcal{I}$
$\mathcal{I} \models_{p'} \neg a$	*iff*	$a \notin \mathcal{I}$	$\mathcal{I} \models_{p'} \varphi\mathbf{M}\psi$	*iff*	$\varphi\mathbf{M}\psi \in \mathcal{I}$
$\mathcal{I} \models_{p'} \varphi \wedge \psi$	*iff*	$\mathcal{I} \models_{p'} \varphi$ *and* $\mathcal{I} \models_{p'} \psi$	$\mathcal{I} \models_{p'} \varphi\mathbf{R}\psi$	*iff*	$\varphi\mathbf{R}\psi \in \mathcal{I}$
$\mathcal{I} \models_{p'} \varphi \vee \psi$	*iff*	$\mathcal{I} \models_{p'} \varphi$ *or* $\mathcal{I} \models_{p'} \psi$	$\mathcal{I} \models_{p'} \varphi\mathbf{W}\psi$	*iff*	$\varphi\mathbf{W}\psi \in \mathcal{I}$

Let φ and ψ be formulas. The equivalence relation $\sim_{p'}$ is defined as:

$$\varphi \sim_{p'} \psi \coloneqq \forall \mathcal{I}.\, \mathcal{I} \models_{p'} \varphi \iff \mathcal{I} \models_{p'} \psi$$

Then $\sim_{p'}$ is an af- and $\cdot[\cdot]_\nu$-congruence.

Proof sketch. The proof is analogous to the proof of Lemma 3.8 and Lemma 5.5. □

[3] We name the equivalence relation $\sim_q$ since it is an improved version of $\sim_p$ and q comes after p in the alphabet.

Observe that this definition has a fundamental difference to $\models_p$ even tough the definition for $\models_{p'}$ only differs for the case $\neg a$. The satisfaction relation $\models_p$ can always be interpreted as a positive monotone Boolean function, while for $\models_{p'}$ this is not true anymore.

Definition 7.11 ([EKS16])**.** *Let φ be a formula. Then the expanded formula $\mathfrak{Unf}(\varphi)$ is inductively defined as:*

$$
\begin{array}{lll}
\mathfrak{Unf}(\mathbf{tt}) & = \mathbf{tt} & \mathfrak{Unf}(\mathbf{X}\varphi) = \mathbf{X}\varphi \\
\mathfrak{Unf}(\mathbf{ff}) & = \mathbf{ff} & \\
\mathfrak{Unf}(a) & = a & \mathfrak{Unf}(\varphi\mathbf{U}\psi) = \mathfrak{Unf}(\psi) \vee (\mathfrak{Unf}(\varphi) \wedge \varphi\mathbf{U}\psi) \\
\mathfrak{Unf}(\neg a) & = \neg a & \mathfrak{Unf}(\varphi\mathbf{M}\psi) = \mathfrak{Unf}(\psi) \wedge (\mathfrak{Unf}(\varphi) \vee \varphi\mathbf{M}\psi) \\
\mathfrak{Unf}(\varphi \wedge \psi) & = \mathfrak{Unf}(\varphi) \wedge \mathfrak{Unf}(\psi) & \mathfrak{Unf}(\varphi\mathbf{R}\psi) = \mathfrak{Unf}(\psi) \wedge (\mathfrak{Unf}(\varphi) \vee \varphi\mathbf{R}\psi) \\
\mathfrak{Unf}(\varphi \vee \psi) & = \mathfrak{Unf}(\varphi) \vee \mathfrak{Unf}(\psi) & \mathfrak{Unf}(\varphi\mathbf{W}\psi) = \mathfrak{Unf}(\psi) \vee (\mathfrak{Unf}(\varphi) \wedge \varphi\mathbf{W}\psi)
\end{array}
$$

Proposition 7.12. *Let φ and ψ be formulas and let $\sim_q$ be defined as:*

$$\varphi \sim_q \psi \coloneqq \mathfrak{Unf}(\varphi) \sim_{p'} \mathfrak{Unf}(\psi)$$

Then $\sim_p \prec \sim_q \prec \sim_l$ holds and $\sim_q$ is an af- and $\cdot[\cdot]_\nu$-congruence.[4]

Before discussing the proof let us go back to our motivating examples: We have $\mathfrak{Unf}(a \vee \neg a) = a \vee \neg a$ and $\mathcal{I} \models_{p'} a \vee \neg a$ for all propositional assignments $\mathcal{I}$. Thus $(a \vee \neg a) \sim_q \mathbf{tt}$ holds. Applying $\mathfrak{Unf}$ to the second example yields:

$$\mathfrak{Unf}(\mathbf{G}(a\mathbf{U}b)) \sim_p \mathbf{G}(a\mathbf{U}b) \wedge (b \vee (a \wedge (a\mathbf{U}b))) \sim_p \mathfrak{Unf}(\mathbf{G}(a\mathbf{U}b) \wedge (a\mathbf{U}b))$$

Since the formula does not contain $\neg a$ or $\neg b$, $\models_p$ and $\models_{p'}$ coincide and we have $\mathbf{G}(a\mathbf{U}b) \sim_q \mathbf{G}(a\mathbf{U}b) \wedge a\mathbf{U}b$. If we look closer at this phenomenon, it turns out that $\sim_q$ collapses 'suspendable' [BBD+13] formulas to a single equivalence class, since $\mathbf{GF}\varphi \sim_q \mathit{af}(\mathbf{GF}\varphi, w)$ and $\mathbf{FG}\varphi \sim_q \mathit{af}(\mathbf{FG}\varphi, w)$ holds for all formulas φ and finite words w.

Since the proof is almost exclusively concerned with technicalities of propositional logic, we moved it to the appendix of this chapter (Section 7.A). The only noteworthy detail is the relaxation we need for making $\sim_q$ an $\cdot[\cdot]_\nu$-congruence: we require $\varphi \sim \psi \implies \varphi[X]_\nu \sim \psi[X]_\nu$ only for formulas φ and ψ that are in *normal-form*. This means every formula φ is implicitly converted to φ' such that φ' is in normal-form before we apply $\cdot[\cdot]_\nu$. This normal-form is computed by a normalisation function f with $\varphi \sim \varphi' = f(\varphi)$. In the case of $\sim_q$ this function is $\mathfrak{Unf}$. In the case of $\sim_c$ and $\sim_p$ it is the identity.

7.5 Various Optimisations

7.5.1 DRA Construction

Immediately after Proposition 5.6 we already noted that $\cdot[\cdot]_\nu$-congruence is actually only needed for the first component and the second component can use another equivalence relation that is not necessarily a $\cdot[\cdot]_\nu$-congruence. We now give a formal definition for this:

[4] In the proof we introduce a minor relaxation to the definition of $\cdot[\cdot]_\nu$-congruence in order to make this true. Examples 5.7, 5.9 and 7.4 are chosen in such a way that this relaxation is not relevant for the construction of the automata.

Proposition 7.13. *Let $\sim$ be an af- and $\cdot[\cdot]_\nu$-congruence, let $\approx$ be af-congruence with $\sim \prec \approx$, let φ be a formula, and let X be a set of formulas. Then the following DCA over the alphabet 2^{Ap} recognises exactly the words w such that $\exists i.\, w_i \models af(\varphi, w_{0i})[X]_\nu$ holds:*

$$\mathcal{C}^1_{\varphi,X} = (Q, \delta, \langle [\varphi]_\sim, [\varphi[X]_\nu]_\approx \rangle, fin\,(\alpha))$$

where we define the states Q, the transition function δ, and the rejecting states α in the following way:

- $Q = Reach(\varphi)_{/\sim} \times \bigcup_{\psi \in Reach(\varphi)} Reach(\psi[X]_\nu)_{/\approx}$. *That is, a state is a tuple of equivalence classes $[\cdot]_\sim$, where the second tracks the formula after applying $\cdot[\cdot]_\nu$.*
- $$\delta(\langle [\xi]_\sim, [\zeta]_\approx \rangle, \sigma) = \begin{cases} \langle [af(\xi,\sigma)]_\sim, [af(\xi,\sigma)[X]_\nu]_\approx \rangle & \text{if } \zeta \approx \mathbf{ff} \\ \langle [af(\xi,\sigma)]_\sim, [af(\zeta,\sigma)]_\approx \rangle & \text{otherwise.} \end{cases}$$
  That is, a transition either resets a failed attempt to prove $w_i \models af(\varphi, w_{0i})[X]_\nu$ for some i or continues unfolding both equivalence classes using $af_\sim$ and $af_\approx$, respectively.
- $\alpha = Reach(\varphi)_{/\sim} \times \{[\mathbf{ff}]_\approx\}$.

Proof sketch. The proof is analogous to the proof of Proposition 5.6 with the small addition that we also need to show that switching from $\sim$ to $\approx$ is well-defined. We can follow this immediately from our assumption $\sim \prec \approx$ which guarantees a surjective mapping of equivalence classes of $\sim$ to equivalence classes of $\approx$. □

7.5.2 NBA Construction

LTL Fragment Detection. A practical approach to reduce the number of states in the constructed NBA is to monitor the current state Ψ and detect if the state is in a specially supported fragment, e.g. if $\Psi \subseteq \nu LTL$ or $\Psi \subseteq \mu LTL$, then the Proposition 6.10 can be used without the need for a full accepting component. The NBA from Figure 7.5 can be reduced by another two states, since the initial states belong to 'simple' fragments. The result for such an optimisation is illustrated in Figure 7.8. Furthermore if the clause Ψ only contains formulas without $\mathbf{U}, \mathbf{M}, \mathbf{R}$, and $\mathbf{W}$, a deterministic construction as in Proposition 5.1 can be used to obtain smaller automata, e.g. the formula $\mathbf{X}(a \vee \mathbf{X}b \vee \mathbf{XX}c)$ is translated to a smaller DBA than NBA.

Contradictory or Universal Clauses. From Lemma 6.9 we know that the a clause describes a language. If by some reasoning, e.g. LTL reasoning, we determine that the language of a clause is empty or universal, we are allowed to either remove it or replace it by $\emptyset$.

Suspendable Formulas and Simulation-based Techniques. Further, special handling for $\mathbf{FG}\varphi$ and $\mathbf{GF}\varphi$ formulas can be added in the style of [BBD+13], where such formulas are replaced by a placeholder and checking these formulas is delayed to accepting SCCs, in our case the accepting component. Lastly, there is substantial work done on simulation-based reductions, which can be added as a post-processing step.

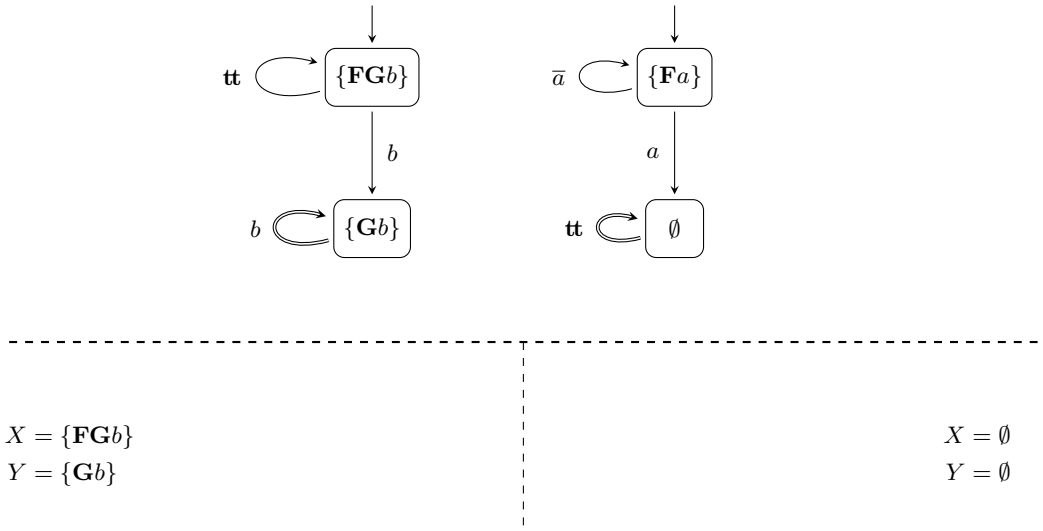


Figure 7.8: $\mathcal{A}_{\mathsf{NBA}}(\varphi)$ for $\varphi = \mathbf{F}a \vee \mathbf{FG}b$ built from components with transition acceptance (Proposition 7.5). Further, LTL fragment detections for clauses is applied.

7.A Omitted Proofs

Proof of Proposition 7.12

Before we dive into the proof of Proposition 7.12 let us collect some useful facts about $\mathfrak{Unf}$. Also notice that, since $\sim_p$ and $\sim_{p'}$ are almost identical, most of the results (with the exception of monotonicity lemmas) we have about $\sim_p$ also apply to $\sim_{p'}$. Thus in the following we often invoke results for $\sim_p$ that can also be proven for $\sim_{p'}$ without detailing the corresponding proof.

Lemma 7.14. *Let φ be a formula. Then the following holds:*

1. $\varphi \sim_l \mathfrak{Unf}(\varphi)$
2. $\mathfrak{Unf}(\varphi) \sim_{p'} \mathfrak{Unf}(\mathfrak{Unf}(\varphi))$
3. $\varphi \sim_{p'} \psi \implies \mathfrak{Unf}(\varphi) \sim_{p'} \mathfrak{Unf}(\psi)$

Proof sketch. (1) and (2) follow from a straight-forward induction on φ. For (1) one has to observe that $\mathfrak{Unf}$ just rewrites the formula using classic LTL expansion rules. For (2) the only interesting cases are the binary modal operators. We look at one representative case ($\varphi = \psi \mathbf{U} \chi$) and derive:

$$\begin{aligned}\mathfrak{Unf}(\mathfrak{Unf}(\psi \mathbf{U} \chi)) &\sim_{p'} \mathfrak{Unf}(\mathfrak{Unf}(\chi)) \vee (\mathfrak{Unf}(\mathfrak{Unf}(\psi)) \wedge (\mathfrak{Unf}(\chi) \vee (\mathfrak{Unf}(\psi) \wedge \psi \mathbf{U} \chi)))\\ &\sim_{p'} \mathfrak{Unf}(\chi) \vee (\mathfrak{Unf}(\psi) \wedge (\mathfrak{Unf}(\chi) \vee (\mathfrak{Unf}(\psi) \wedge \psi \mathbf{U} \chi)))\\ &\sim_{p'} \mathfrak{Unf}(\chi) \vee (\mathfrak{Unf}(\psi) \wedge \psi \mathbf{U} \chi)\\ &\sim_{p'} \mathfrak{Unf}(\psi \mathbf{U} \chi)\end{aligned}$$

(3) Notice that $\mathfrak{Unf}$ is in-fact a substitution only replacing binary modal operators. Thus constants ($\mathbf{tt}$, $\mathbf{ff}$), literals (a, $\neg a$), the next operator ($\mathbf{X}$), and the Boolean connectives ($\wedge$, $\vee$) are not replaced in the syntax tree. Observe that such substitutions are congruent on $\sim_{p'}$ and, since $\mathfrak{Unf}$ is an instance of such a substitution, we are done. □

Proposition 7.12. *Let φ and ψ be formulas and let $\sim_q$ be defined as:*

$$\varphi \sim_q \psi \coloneqq \mathfrak{Unf}(\varphi) \sim_{p'} \mathfrak{Unf}(\psi)$$

Then $\sim_p \prec \sim_q \prec \sim_l$ holds and $\sim_q$ is an af- and $\cdot[\cdot]_\nu$-congruence.[5]

Proof. We have to prove the following four points: (1a) $\sim_p \prec \sim_q$, (1b) $\sim_q \prec \sim_l$, (2) $\sim_q$ is an *af*-congruence, and (3) $\sim_q$ is an $\cdot[\cdot]_\nu$-congruence.

(1a) We start by noting $\sim_p \preccurlyeq \sim_{p'}$. By Lemma 7.14 we also have $\sim_{p'} \preccurlyeq \sim_q$ and immediately by transitivity we obtain $\sim_p \preccurlyeq \sim_q$. It remains to show $\sim_p \neq \sim_q$, which is directly proven by $\mathbf{tt} \not\sim_p a \vee \neg a \sim_q \mathbf{tt}$.

(1b) $\sim_q \prec \sim_l$. Let φ, ψ be formulas. Assume $\varphi \sim_q \psi$. We now need to prove $\varphi \sim_l \psi$. For this let w be an arbitrary word and let $\mathcal{I} = \{\psi \in sf(\varphi) : w \models \psi\}$ be the set of all

[5] In the proof we introduce a minor relaxation to the definition of $\cdot[\cdot]_\nu$-congruence in order to make this true. Examples 5.7, 5.9 and 7.4 are chosen in such a way that this relaxation is not relevant for the construction of the automata.

satisfied proper subformulas.

$$\begin{aligned} w \models \varphi &\iff w \models \mathfrak{Unf}(\varphi) && \text{(Lemma 7.14)} \\ &\iff \mathcal{I} \models_{p'} \mathfrak{Unf}(\varphi) && \text{(Lemma 2.7 for } \models_{p'}) \\ &\iff \mathcal{I} \models_{p'} \mathfrak{Unf}(\psi) && (\varphi \sim_q \psi) \\ &\iff w \models \mathfrak{Unf}(\psi) && \text{(Lemma 2.7 for } \models_{p'}) \\ &\iff w \models \psi && \text{(Lemma 7.14)} \end{aligned}$$

Again we need to show that $\sim_q$ is not $\sim_l$, which is immediately clear from $\mathbf{tt} \nsim_q \mathbf{X}a \vee \mathbf{X}\neg a \sim_l \mathbf{tt}$.

(2) The first requirement for being an *af*-congruence ($\sim_c \preccurlyeq \sim_q \preccurlyeq \sim_l$) follows from (1) and transitivity. For the second requirement we need to show $\varphi \sim_q \psi \implies \mathit{af}(\varphi, \sigma) \sim_q \mathit{af}(\varphi, \sigma)$ for all formulas φ, ψ and letters σ. For this we prove $\mathit{af}(\varphi, \sigma) \sim_{p'} \mathit{af}(\mathfrak{Unf}(\varphi), \sigma)$ for all formulas φ and letters σ by induction on φ. We then derive our second requirement with following steps:

$$\begin{aligned} \varphi \sim_q \psi &\iff \mathfrak{Unf}(\varphi) \sim_{p'} \mathfrak{Unf}(\psi) \\ &\implies \mathit{af}(\mathfrak{Unf}(\varphi), \sigma) \sim_{p'} \mathit{af}(\mathfrak{Unf}(\psi), \sigma) && \text{(Proposition 7.10)} \\ &\iff \mathit{af}(\varphi, \sigma) \sim_{p'} \mathit{af}(\psi, \sigma) \\ &\implies \mathit{af}(\varphi, \sigma) \sim_q \mathit{af}(\psi, \sigma) && (\sim_{p'} \preccurlyeq \sim_q) \end{aligned}$$

(3) In fact $\sim_q$ is not $\cdot[\cdot]_\nu$-congruence for the existing definition, e.g. $\mathbf{FG}a \sim_q \mathbf{FG}a \vee (\mathbf{G}a \wedge a)$, but $(\mathbf{FG}a)[\emptyset]_\nu = \mathbf{ff} \nsim_q \mathbf{ff} \vee (\mathbf{G}a \wedge a) = (\mathbf{FG}a \vee (\mathbf{G}a \wedge a))[\emptyset]_\nu$. We solve this by requiring $\varphi \sim \psi \implies \varphi[X]_\nu \sim \psi[X]_\nu$ only for formulas φ and ψ that are in *normal-form*. This means every formula φ is implicitly converted to φ' such that φ' is in normal-form before we apply $\cdot[\cdot]_\nu$. This normal-form is computed by a normalisation function f with $\varphi \sim \varphi' = f(\varphi)$. In the our case this function is $\mathfrak{Unf}$. We then derive:

$$\begin{aligned} \varphi \sim_q \psi &\iff \mathfrak{Unf}(\varphi) \sim_{p'} \mathfrak{Unf}(\psi) \\ &\implies (\mathfrak{Unf}(\varphi))[X]_\nu \sim_{p'} (\mathfrak{Unf}(\psi))[X]_\nu && \text{(Proposition 7.10)} \\ &\implies (\mathfrak{Unf}(\varphi))[X]_\nu \sim_q (\mathfrak{Unf}(\psi))[X]_\nu && (\sim_{p'} \preccurlyeq \sim_q) \\ &\implies (f(\varphi))[X]_\nu \sim_q (f(\psi))[X]_\nu \end{aligned}$$

This is exactly what we needed to show for $\cdot[\cdot]_\nu$-congruence and we are done. □

8 Experimental Evaluation

We provide experimental evidence showing that the better theoretical properties of constructions derived from the Master Theorem are *not* obtained at the expense of poor performance in practice. We support this claim by comparing sizes of automata produced by other translations to the sizes of our translation approach. Note that we do not include a resource consumption analysis, i.e. measurements of computation time and allocated memory. These values are highly dependent on implementation details and do not affect the quantities we want to measure as long as the computations terminate within reasonable bounds, which is the case for our experiments.

8.1 Method

Translations

We compare each translation derived from the Master Theorem to an alternative translation for that specific automaton model. In previous chapters we used the terms NBA, LDBA, and DRA also for generalised versions of these acceptance conditions, since there are no conceptual differences for the presented constructions. However, for our evaluation we measure the number of states and acceptance sets and these quantities can change when we consider generalised acceptance conditions. Consequently, we sharpen our terminology and for this chapter we refer with the term NBA only to nondeterministic Büchi automata and with the term NGBA to nondeterministic *generalised* Büchi automata. The same applies also to the terms LDGBA and DGRA. Further, in our evaluation we only consider automata with the acceptance condition defined on transitions.

For each translation target (NBA, NGBA, LDBA, LDGBA, DRA, DGRA) we select another translator that is known to be competitive or particular good for that automaton class. We focus here only on translators that support full LTL and not only fragments. Further, we disable for all tools in the comparison the applied post-processing steps in order to avoid artefacts due to automata-based reduction techniques. We select the following tools to compare against:

- NGBA:

  `ltl2tgba`[1] [DLF+16] is a mature and well maintained tool to translate LTL to nondeterministic transition-based generalised Büchi automata.[2] We do not compare the translation to NBAs, because the built-in degeneralisation only translates to *state-based* nondeterministic Büchi automata. We disable post-processing steps that are independent of the translation by the invocation `ltl2tgba -any -low -H` as recommend by the documentation.

[1] We use the version `ltl2tgba (spot) 2.7.2`.

[2] In recent years it also acquired the capability to produce deterministic parity automata by applying a determinisation construction.

– LDBA and LDGBA:

`ltl2ldba` and `ltl2ldgba` from the `Rabinizer 4` distribution [KMSZ18] implementing the translation to LDBAs and LDGBAs from [SEJK16]. Since the translations based on the Master Theorem and these translations are both based on [KMS18], some of the optimisations from Chapter 7 are also retroactively added to the original construction of [SEJK16].

– DRA and DGRA:

`ltl2dra` and `ltl2dgra` from the `Rabinizer 4` distribution [KMSZ18] implementing the translation to DRAs and DGRAs as described in [EKS16][3]. All available optimisations to the translation itself are switched on. In our comparison we enable for this group of tools the reduction of Rabin pairs with rules from [EKS16] and de-generalise DGRAs to DRAs, since the construction of [EKS16] does not support the construction of DRAs directly.

We implement translations based on the Master Theorem with the optimisations presented in Chapter 7 in tools named: `ltl2nba`, `ltl2ngba`, `ltl2ldba`, `ltl2ldgba`, `ltl2dra`, and `ltl2dgra`. To be more precise we use the Master Theorem with restricted guessing (Proposition 4.18), the equivalence relation $\sim_q$, and the specialised intersection constructions: interleaved intersection, formula rewriting, and generalised Büchi acceptance conditions, depending on the context.

Since these implementation occupy the same source-code repository[4] as [KMSZ18; KMS18], the user needs to disambiguate between the implementations using the flag `--symmetric` which selects translations based on the Master Theorem and the flag `--asymmetric` which selects the previously mentioned translations. This naming scheme is due to the symmetric nature of the Master Theorem, where least- and greatest-fixed-point operators are considered, and the asymmetric nature of the other translations, where only greatest-fixed-point operators play a role.

It is notable that we do not include tools, such as `ltl2dstar` [KB06] or `seminator` [BDK+17]. The reasons for this are twofold: first, in previous work it was shown that the approach implemented in `Rabinizer 4` [EKS16; SEJK16] often outperforms automata-based translations from NBAs, e.g. implemented by `ltl2dstar`; second, as noticed in [BDK+17] most of the tested formulas are already translated to limit-deterministic (or even deterministic) automata by the used LTL translator `ltl2tgba` which we already include in the comparison. Further, we do not compare our implementations to translations targeting deterministic parity automata (DPA) that are a subclass of Rabin automata. We do this because DRAs can be more succinct than DPAs and thus a comparison is most likely skewed towards the DRA side. Furthermore, the approaches of [EKRS17; KMWW17] use DRAs and LDBAs from translations implemented in `Rabinizer 4` which is already included in the comparison.

Formula Sets

We base the evaluation on three sets of formulas: the first set consists of the well-known 'Dwyer'-patterns [DAC98] that collects 55 LTL formulas specifying common properties;

[3] This is the revised and corrected version of the translation proposed by [EK14].

[4] We evaluate the state of the implementation found in the repository of [KMS18] defined by commit `30dd44558e10fbae2134bc335bfbc69492b7f2bf`.

$$
\begin{aligned}
\chi_{1,n} &= (\dots((a_1 \mathbf{U} a_2)\mathbf{U} a_3)\dots \mathbf{U} a_{n+1}) & \chi_{2,n} &= a_1 \mathbf{U}(a_2 \mathbf{U}(\dots(a_n \mathbf{U} a_{n+1})\dots)) \\
\chi_{3,n} &= \mathbf{G}(a_1 \rightarrow a_1 \mathbf{U}(\dots(a_n \wedge a_n \mathbf{U} a_{n+1}))) & \chi_{4,n} &= \textstyle\bigwedge_{i=1}^{n}(\mathbf{F} a_i \vee \mathbf{G} a_{i+1}) \\
\chi_{5,n} &= \textstyle\bigwedge_{i=1}^{n} \mathbf{FG} a_i & \chi_{6,n} &= \textstyle\bigwedge_{i=1}^{n} \mathbf{GF} a_i \\
\chi_{7,n} &= (\textstyle\bigwedge_{i=1}^{n} \mathbf{GF} a_i) \rightarrow \mathbf{GF} b & \chi_{8,n} &= (\textstyle\bigwedge_{i=1}^{n} \mathbf{GF} a_i) \leftrightarrow \mathbf{GF} b \\
\chi_{9,n} &= \textstyle\bigwedge_{i=1}^{n}(\mathbf{GF} a_i \vee \mathbf{FG} a_{i+1}) & \chi_{10,n} &= \mathbf{GF}(a \leftrightarrow \mathbf{X}^n a) \\
\chi_{11,n} &= \textstyle\bigvee_{i=0}^{n} \mathbf{FG}((\neg)^i a \vee \mathbf{X}^i b)
\end{aligned}
$$

Table 8.1: Parametrised formula set.

the second set is extracted from 'BEEM' [Pel07], a collection of benchmarks for explicit model checkers containing 20 formulas; the last set is obtained by instantiating the 11 parametrised formulas from Table 8.1. These families are partly taken from [TRV12; GH06; MS17] or are simple combinations of **U**, **GF**, and **FG** formulas. The last set of formulas is useful to isolate and analyse strong- and weak points of the compared translations. For completeness we also evaluated the translations on the sets from [EH00; SB00], but, since these two set did not bring major additional insights to the analysis, we left them out from the analysis and moved them to the appendix of this chapter. Furthermore, we abstained from using randomly generated formulas, because in our experience it us unclear what this implies for practice, since formulas from real-world examples usually have a high degree of structure compared to randomly generated formulas.

The formula sets are obtained by executing `genltl`[5] with the corresponding parameters. Each formula and its negation is then added to the set of formulas. We take the following steps to reduce the influence of specific simplification rules and to de-duplicate entries: first, we bring formulas into negation normal form; second, we apply a standard set of LTL simplification rules[6] allowing us to turn-off or at-least restrict the LTL simplifier in the evaluation; third, we normalise the literal names and remove formulas that are equal modulo literal renaming. This pre-processing has the effect that the number of formulas we consider is less than the number of formulas of the corresponding original publication. For example [DAC98] lists 55 formulas, but we remove six entries. For example only one of $\mathbf{G}a$, $\mathbf{G}\neg a$, and $\mathbf{F}a$ is added to the formula set. Note that we always evaluate the translation also on the negation of each formula. However, we do not remove duplicates across two different formula sets.

8.2 Results

The measured automata sizes for the LTL formulas are listed in Tables 8.2 to 8.5. We refer by φ to formulas of the 'Dwyer' set, by ψ to formulas of the 'BEEM' set, and by χ to formulas of the parametrised set. To reduce the noise we filter out rows that we consider 'minor' and moved them to Section 8.A. 'Minor' rows are pairs of rows, i.e., the row for a formula and the row for its negation, such that the difference in the number of states between compared translators is at most 3. We label columns by the respective tools: 'NBA', 'NGBA', 'LDBA', 'LDGBA', 'DRA', and 'DGRA' refer to translations derived from the Master Theorem; 'NGBA (Spot)' denotes the translations implemented

[5] `genltl` is a component of `Spot` [DLF+16] to generate LTL formulas from existing patterns. We use the version `genltl (spot) 2.7.2`.

[6] We refer the interested reader to [EH00; SB00; BKRS12; Sic16; MS17] for some material on LTL formula rewriting.

by `ltl2tgba`; 'LDBA (Rab. 4)', 'LDGBA (Rab. 4)', 'DRA (Rab. 4)', and 'DGRA (Rab. 4)' refers to the 'asymmetric' translations from [SEJK16] and [EKS16]. Columns are grouped by the pairs we compare. The smallest number of states and the smallest number of acceptance sets in each group are printed in boldface type. Lastly, we save space by writing $\overline{\varphi}$ instead of $\neg\varphi$.

8.3 Discussion

It is remarkable that in the 'Dwyer' set the differences for roughly 50% of the formula pairs are considered 'minor'. In the 'BEEM' set this percentage is even higher (70%). Inspecting the appendix shows that for most of these 'minor' cases the automata are of equal size or only differ in size by an optional rejecting trap state. Thus already on a large percentage of tested formulas the unified construction does not construct larger automata. Let us move on to the cases with 'major' differences. For these we can see several interesting patterns in the data:

- $\mu LTL + \nu LTL$

  If a formula is either from μLTL or νLTL, then the translations generally speaking all produce automata of the same (or roughly) the same size. However, the translators 'DRA (Rab. 4)' and 'DGRA (Rab. 4)' seem to have difficulties with deeply nested **R** operators as seen in χ_5 and χ_6. This is probably due to the inability to support **R** in the construction and the necessity to rewrite it to $\varphi\mathbf{M}\psi \vee \mathbf{G}\psi$. Further, the translators 'NBA' and 'NGBA' also have troubles with the same two formulas, but this is probably due to a different reason, and not inherent to the Master Theorem, as the other translations based on it are unaffected.

- Nesting of **U**-operators

  One fascinating pattern is the effect of nesting **U**-operators within the scope of a **G**-operator. The formulas φ_{13}, φ_{14}, φ_{28}, φ_{29}, φ_{39}, φ_{44}, φ_{48}, φ_{49} from the 'Dwyer' set, ψ_{17}, ψ_{18} from the 'BEEM' set, and the instantiations χ_8, χ_9 all have in common that translations based on the Master Theorem tend to produce disproportionally large automata compared to alternative translators. Surprisingly, for the negation the situation reverses and the Master Theorem produces in the LD(G)BA and D(G)RA groups smaller automata than the comparison. Thus it would seem that deep nesting of smallest-fixed-point operators is problematic for the Master Theorem. However, there are cases, such as φ_{33}, φ_{37}, and φ_{38}, where deep nesting does not have this effect. We speculate that the number of fixed-points that are considered can be reduced for cases where we nest smallest-fixed-point operators.

- Disjunctive Normal Form

  Formulas with a large disjunctive normal form (DNF) apparently pose a problem for the translations based on the Master Theorem, especially in combination with the **F**- and **G**-operators. This can be seen in χ_{11} and χ_{12}. Note that the negation which has a small DNF does not exhibit this size blow-up. This pattern can also be observed for the formulas χ_{22}, χ_{23}, and χ_{24}. However, this observation cannot be generalised, since χ_{27} has a large DNF, but automata based on the Master Theorem are smaller. We think that an approach to reliably obtain small automata for such

formulas is to use a compositional construction as the one proposed by [MS17] and to transform the obtained Emerson-Lei acceptance condition into a generalised Rabin acceptance condition.

– **X**-operators within a **GF** and **FG** scope

  In the parametrised set we observe that for the formulas from χ_{29} up-to χ_{33} the DRAs and DGRAs from the Master Theorem are smaller than the automata from the Rabinizer translators. We think that this is due to the interleaved intersection construction that is able to 'suspend' tracking of some of the formulas containing **X** operators. Notice that for LDBA and LDGBA groups the numbers are identical, since all implementations use the same optimisations.

– Improvable NBA and NGBA implementations

  The NBA and NGBA translations based on the Master Theorem tend to produce large automata compared to either `ltl2tgba` or even in some cases the translators for deterministic automata. This is not surprising, since these NBA and NGBA translators implement only a small fraction of the translation optimisations for NBAs and NGBAs that have been proposed over the last two decades: for example, it is clear how to implement a heuristic that obtains a single state NGBA for ψ_{20} or if we apply the reduction techniques implemented `ltl2tgba` with `autfilt --small` to NGBA for χ_8 we shrink the automaton from 19 to 7 states.

The collected data demonstrates that on this selection of formulas, coming from a variety of sources, the simplicity and generality of our new constructions does not lead to a general penalty in practice. For most benchmarks our construction produces automata of similar size to those computed by `Rabinizer 4`, in a few cases even slightly smaller ones, e.g. for φ_{33}. In other cases, notably for formulas with a deep nesting of **U**-operators, the new constructions still perform poorly. However, we feel confident that further analysis of these cases leads to an optimised version addressing this issue.

In comparison to `ltl2tgba` the translators based on the Master Theorem for NBAs and NGBAs fall behind. We think that for some classes of formulas the used two-stage construction with an initial and accepting component comes with additional costs. It seems a second implementation using the single-stage construction from [EKS18] could be beneficial. Further, we believe that a portfolio approach as implemented by `ltl2tgba` selecting different translation strategies depending on the input has advantages and could close the size gap.

LTL	NBA	NGBA (Spot)	NGBA	LDBA (Rab. 4)	LDBA	LDGBA (Rab. 4)	LDGBA	DRA (Rab. 4)	DRA	DGRA (Rab. 4)	DGRA
φ_4	**7**	**3**	7	**6**	7	**6**	7	4 **(4)**	**3 (4)**	4 **(3)**	**3 (3)**
$\overline{\varphi_4}$	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4 (2)**	**4 (2)**	**4**	**4**
φ_{13}	**151**	**7** (5)	151	**14**	116	**14**	116	**22 (4)**	32 **(4)**	**22 (3)**	32 **(3)**
$\overline{\varphi_{13}}$	**8**	**8**	**8**	**8**	**8**	**8** (2)	**8**	22 **(2)**	**8 (2)**	22	**8**
φ_{14}	**243**	**7** (4)	243	**14**	122	**14**	122	**11 (6)**	677 (28)	**11 (4)**	677 (28)
$\overline{\varphi_{14}}$	**16**	**7** (3)	15 **(2)**	45	**16**	41 (3)	**15 (2)**	16 **(2)**	**7 (2)**	16 (5)	**7 (4)**
φ_{18}	**10**	**3**	10	**6**	8	**6**	8	**4 (4)**	**4 (4)**	**4 (3)**	**4 (3)**
$\overline{\varphi_{18}}$	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4 (2)**	**4 (2)**	**4**	**4**
φ_{23}	**16**	**4 (2)**	13 **(2)**	**7**	13	**7**	10 (2)	**4 (4)**	6 (6)	4 **(3)**	**3** (6)
$\overline{\varphi_{23}}$	**4**	**4**	**4**	8	**4**	8	**4**	**4 (2)**	**4 (2)**	**4**	**4**
φ_{24}	**8**	**3**	8	**8**	**8**	**8**	**8**	6 (6)	**3 (4)**	6 (6)	**3 (3)**
$\overline{\varphi_{24}}$	**4**	**4**	**4**	7	**6**	7	**6**	**4 (2)**	**4 (2)**	**4**	**4**
φ_{27}	**6**	**4**	6	**5**	**5**	**5**	**5**	**4 (2)**	5 **(2)**	**4**	5
$\overline{\varphi_{27}}$	**4**	**4**	**4**	11	**7**	11 (2)	**7**	5 **(2)**	**4 (2)**	5	**4**
φ_{28}	**46**	**6 (2)**	38 **(2)**	**10**	28	**10**	25 (2)	**8 (4)**	27 (8)	**8 (3)**	18 (6)
$\overline{\varphi_{28}}$	**5**	**5**	**5**	9	**5**	9	**5**	8 **(2)**	**5 (2)**	8	**5**
φ_{29}	**46**	**4**	38 (2)	**9**	20	**9**	18 (2)	**4 (4)**	20 (8)	**4 (3)**	11 (6)
$\overline{\varphi_{29}}$	**4**	**4**	**4**	18	**4**	18	**4**	6 **(2)**	**4 (2)**	6	**4**
φ_{33}	**32**	**6**	32	14	**13**	14	**13**	20 (8)	**6 (4)**	20 (9)	**6 (3)**
$\overline{\varphi_{33}}$	**5**	**5**	**5**	**6**	**6**	**6**	**6**	**6 (2)**	**6 (2)**	**6**	**6**
φ_{34}	**48**	**8**	48	19	**15**	19	**15**	**8** (12)	13 **(4)**	**8** (9)	13 **(3)**
$\overline{\varphi_{34}}$	**11**	**10**	11	12	**6**	12	**6**	**6 (2)**	**6 (2)**	**6**	**6**
φ_{35}	**10**	**8 (2)**	9 **(2)**	11	**8**	11	**7** (2)	**4** (6)	7 **(4)**	**4** (6)	5 **(4)**
$\overline{\varphi_{35}}$	**3**	**3**	**3**	6	**5**	6	**5**	**4 (2)**	**4 (2)**	**4**	**4**
φ_{36}	**9**	12	**9**	11	**7**	11	**7**	**6 (2)**	**6 (2)**	**6** (2)	**6**
$\overline{\varphi_{36}}$	**4**	**4**	**4**	**5**	**5**	**5**	**5**	**5 (2)**	**5 (2)**	**5**	**5**
φ_{37}	**12**	**9 (2)**	11 **(2)**	13	**10**	13	**9** (2)	11 (6)	**9 (4)**	11 (7)	**6 (4)**
$\overline{\varphi_{37}}$	**4**	**4**	**4**	7	**6**	7	**6**	**5 (2)**	**5 (2)**	**5** (2)	**5**
φ_{38}	**80**	**12 (3)**	48 **(3)**	**27**	39	27	**24** (3)	**20 (4)**	55 (12)	20 **(3)**	**14** (12)
$\overline{\varphi_{38}}$	**5**	**5**	**5**	18	**9**	18	**9**	**8 (2)**	**8 (2)**	**8**	**8**
φ_{39}	**216**	**10 (2)**	134 (3)	**40**	69	**40**	42 (3)	**54** (18)	838 **(12)**	**54** (18)	263 **(16)**
$\overline{\varphi_{39}}$	**103**	**5**	91 (2)	**20**	48	**20**	42 (2)	29 **(6)**	**22 (6)**	29 (11)	**16 (5)**
φ_{42}	**7**	**3**	7	**7**	**7**	**7**	**7**	4 **(2)**	**3** (4)	4 **(3)**	**3 (3)**
$\overline{\varphi_{42}}$	**4**	**4**	**4**	**5**	**5**	**5**	**5**	**4 (2)**	5 **(2)**	**4** (2)	5
φ_{43}	**53**	**6 (3)**	37 **(3)**	**9**	34	**9**	23 (3)	**6 (4)**	21 (8)	**6 (3)**	7 (6)
$\overline{\varphi_{43}}$	**5**	**5**	**5**	19	**5**	19	**5**	8 **(2)**	**5 (2)**	8	**5**

φ_4	$\mathbf{G}(a \vee b \vee \mathbf{G}\overline{b} \vee c\mathbf{U}b)$
φ_{13}	$\mathbf{G}(a \vee \mathbf{G}\overline{b} \vee (\overline{b} \wedge \overline{c})\mathbf{U}(b \vee (\overline{b} \wedge c)\mathbf{U}(b \vee (\overline{b} \wedge \overline{c})\mathbf{U}(b \vee (\overline{b} \wedge c)\mathbf{U}(b \vee \overline{c}\mathbf{U}b)))))$
φ_{14}	$\mathbf{G}(a \vee (\overline{b} \wedge \overline{c})\mathbf{U}(c \vee (b \wedge \overline{c})\mathbf{U}(c \vee (\overline{b} \wedge \overline{c})\mathbf{U}(c \vee (b \wedge \overline{c})\mathbf{U}(c \vee \mathbf{G}b \vee \overline{b}\mathbf{W}c)))))$
φ_{18}	$\mathbf{G}(a \vee b \vee \mathbf{G}\overline{b} \vee c\mathbf{U}(b \vee d))$
φ_{23}	$\mathbf{G}(a \vee b \vee \mathbf{G}\overline{b} \vee (c \vee \overline{b}\mathbf{U}(\overline{b} \wedge d))\mathbf{U}b)$
φ_{24}	$\mathbf{G}(a \vee b \vee (c \vee \overline{b}\mathbf{U}(\overline{b} \wedge d))\mathbf{W}b)$
φ_{27}	$\mathbf{G}a \vee a\mathbf{U}(a \vee \mathbf{G}b \vee b\mathbf{U}(b \wedge c \wedge \mathbf{X}(b\mathbf{U}d)))$
φ_{28}	$\mathbf{G}(a \vee \mathbf{G}\overline{b} \vee c\mathbf{U}(b \vee (c \wedge d \wedge \mathbf{X}(c\mathbf{U}e))))$
φ_{29}	$\mathbf{G}(a \vee \mathbf{G}b \vee b\mathbf{U}(c \vee (b \wedge d \wedge \mathbf{X}(b\mathbf{U}e))))$
φ_{33}	$\mathbf{G}(a \vee \mathbf{G}\overline{b} \vee (b \vee c \vee \mathbf{X}(b\mathbf{R}(b \vee d)))\mathbf{U}(b \vee e))$
φ_{34}	$\mathbf{G}(a \vee \mathbf{G}(b \vee \mathbf{X}\mathbf{G}c) \vee (b \vee d \vee \mathbf{X}(d\mathbf{R}(c \vee d)))\mathbf{U}(d \vee e))$
φ_{35}	$\mathbf{G}(a \vee \mathbf{X}\mathbf{G}\overline{b} \vee \mathbf{X}\mathbf{F}(b \wedge \mathbf{F}c))$
φ_{36}	$\mathbf{G}\overline{a} \vee (b \vee \mathbf{X}(a\mathbf{R}\overline{c}) \vee \mathbf{X}(\overline{a}\mathbf{U}(c \wedge \mathbf{F}d)))\mathbf{U}a$
φ_{37}	$\mathbf{G}(a \vee \mathbf{G}(b \vee \mathbf{X}\mathbf{G}\overline{c} \vee \mathbf{X}\mathbf{F}(c \wedge \mathbf{F}d)))$
φ_{38}	$\mathbf{G}(a \vee \mathbf{G}\overline{b} \vee (c \vee \mathbf{X}(b\mathbf{R}\overline{d}) \vee \mathbf{X}(\overline{b}\mathbf{U}(d \wedge \mathbf{F}e)))\mathbf{U}b)$
φ_{39}	$\mathbf{G}(a \vee (b \vee \mathbf{X}(c\mathbf{R}\overline{d}) \vee \mathbf{X}(\overline{c}\mathbf{U}(d \wedge \mathbf{F}e)))\mathbf{U}(c \vee \mathbf{G}(b \vee \mathbf{X}(c\mathbf{R}\overline{d}) \vee \mathbf{X}(\overline{c}\mathbf{U}(d \wedge \mathbf{F}e)))))$
φ_{42}	$\mathbf{G}(a \vee \mathbf{G}(b \vee (c \wedge \mathbf{X}\mathbf{F}d)))$
φ_{43}	$\mathbf{G}(a \vee \mathbf{G}\overline{b} \vee (c \vee \overline{b}\mathbf{U}(\overline{b} \wedge d \wedge \mathbf{X}(\overline{b}\mathbf{U}e)))\mathbf{U}b)$

Table 8.2: Automaton sizes for the 'Dwyer'-formula set (1/4). $x\,(y)$ denotes that the number of states is x and the number of (non-trivial) acceptance sets is y.

LTL	NBA	NGBA (Spot)	NGBA	LDBA (Rab. 4)	LDBA	LDGBA (Rab. 4)	LDGBA	DRA (Rab. 4)	DRA	DGRA (Rab. 4)	DGRA
φ_{44}	**179**	**12 (3)**	132 **(3)**	**19**	79	**19**	61 (3)	**20 (6)**	543 (16)	**20 (6)**	103 (16)
$\overline{\varphi_{44}}$	**27**	**11** (2)	27	27	**21**	27 (2)	**21**	18 (14)	**14 (6)**	18 (16)	**14 (5)**
φ_{45}	**7**	**4** (2)	7	8	**6**	8	**6**	**4 (2)**	5 (4)	**4 (2)**	5 (3)
$\overline{\varphi_{45}}$	**3**	**3**	**3**	11	**5**	11	**5**	**5** (8)	**5 (2)**	**5** (8)	**5**
φ_{47}	**8**	**3**	8	**7**	8	**7**	8	4 **(2)**	**3** (4)	4 **(3)**	**3 (3)**
$\overline{\varphi_{47}}$	**4**	**4**	**4**	**5**	6	**5**	6	**4 (2)**	5 **(2)**	**4** (2)	5
φ_{48}	**53**	**6 (3)**	37 **(3)**	**9**	34	**9**	23 (3)	**6 (4)**	21 (8)	**6 (3)**	7 (6)
$\overline{\varphi_{48}}$	**5**	**5**	**5**	19	**5**	19	**5**	8 **(2)**	**5 (2)**	8	**5**
φ_{49}	**269**	**12 (3)**	192 **(3)**	**19**	103	**19**	80 (3)	**20 (6)**	822 (22)	**20 (6)**	139 (22)
$\overline{\varphi_{49}}$	**40**	**12** (2)	40	30	**23**	30	**23**	26 (14)	**19 (6)**	26 (14)	**19 (5)**

φ_{44}	$\mathbf{G}(a \vee (b \vee \overline{c}\mathbf{U}(\overline{c} \wedge d \wedge \mathbf{X}(\overline{c}\mathbf{U}e)))\mathbf{U}(c \vee \mathbf{G}(b \vee (d \wedge \mathbf{XF}e))))$
φ_{45}	$\mathbf{G}(a \vee \mathbf{F}(b \wedge c \wedge \mathbf{X}(c\mathbf{U}d)))$
φ_{47}	$\mathbf{G}(a \vee \mathbf{G}(b \vee (c \wedge d \wedge \mathbf{X}(d\mathbf{U}e))))$
φ_{48}	$\mathbf{G}(a \vee \mathbf{G}\overline{b} \vee (c \vee \overline{b}\mathbf{U}(\overline{b} \wedge d \wedge e \wedge \mathbf{X}((\overline{b} \wedge e)\mathbf{U}f)))\mathbf{U}b)$
φ_{49}	$\mathbf{G}(a \vee (b \vee \overline{c}\mathbf{U}(\overline{c} \wedge d \wedge e \wedge \mathbf{X}((\overline{c} \wedge e)\mathbf{U}f)))\mathbf{U}(c \vee \mathbf{G}(b \vee (d \wedge e \wedge \mathbf{X}(e\mathbf{U}f)))))$

Table 8.3: Automaton sizes for the 'Dwyer'-formula set (2/4). $x\,(y)$ denotes that the number of states is x and the number of (non-trivial) acceptance sets is y.

LTL	NBA	NGBA (Spot)	NGBA	LDBA (Rab. 4)	LDBA	LDGBA (Rab. 4)	LDGBA	DRA (Rab. 4)	DRA	DGRA (Rab. 4)	DGRA
ψ_7	**39**	**4**	39	**9**	23	**9**	23	**7** (8)	**7 (6)**	**7** (8)	**7 (4)**
$\overline{\psi_7}$	**10**	**5** (2)	10	**10**	**10**	**10** (2)	**10**	6 **(2)**	**5** (4)	6 **(2)**	**5** (3)
ψ_{10}	**9**	**2**	8 (2)	**4**	5	**4**	5 (2)	**2 (2)**	**2** (6)	**2 (2)**	**2** (5)
$\overline{\psi_{10}}$	**2**	**2**	**2**	4	**3**	4	**3**	**2 (2)**	**2 (2)**	**2**	**2**
ψ_{13}	**7**	**3**	7	**6**	7	**6**	7	4 **(4)**	**3 (4)**	4 **(3)**	**3 (3)**
$\overline{\psi_{13}}$	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4 (2)**	**4 (2)**	**4**	**4**
ψ_{17}	**10**	**3**	10	**6**	10	**6**	10	**3 (2)**	**3** (4)	**3 (2)**	**3** (3)
$\overline{\psi_{17}}$	**4**	**4**	**4**	9	**7**	9	**7**	**4** (4)	**4 (2)**	**4** (4)	**4**
ψ_{18}	**52**	**5**	52	**10**	45	**10**	45	**7 (2)**	15 (4)	**7 (2)**	15 (3)
$\overline{\psi_{18}}$	**6**	**6**	**6**	20	**11**	20 (2)	**11**	13 (4)	**6 (2)**	13 (3)	**6**
ψ_{20}	**10**	**1 (2)**	7 **(2)**	**4**	**4**	**3 (2)**	**3 (2)**	**2 (4)**	**2 (4)**	**1 (5)**	**1 (5)**
$\overline{\psi_{20}}$	**4**	**3**	4	**3**	4	**3**	4	**1** (4)	3 **(2)**	**1** (2)	3

ψ_7	$\mathbf{G}(a \vee \overline{b}\mathbf{U}(b\mathbf{U}(\overline{b} \wedge c\mathbf{R}\overline{b})))$
ψ_{10}	$\mathbf{G}(a \vee \mathbf{F}b \vee \mathbf{F}c)$
ψ_{13}	$\mathbf{G}(a \vee \mathbf{G}\overline{b} \vee c\mathbf{U}b)$
ψ_{17}	$\mathbf{G}(a \vee \overline{b}\mathbf{U}(b\mathbf{U}(b \wedge c)))$
ψ_{18}	$\mathbf{G}(a \vee \overline{b}\mathbf{U}(b\mathbf{U}(\overline{b}\mathbf{U}(b\mathbf{U}(b \wedge c)))))$
ψ_{20}	$(\mathbf{GF}a \vee \mathbf{GF}b) \wedge (\mathbf{GF}b \vee \mathbf{GF}c)$

Table 8.4: Automaton sizes for the 'BEEM'-formula set (1/2). $x\,(y)$ denotes that the number of states is x and the number of (non-trivial) acceptance sets is y.

LTL	NBA	NGBA (Spot)	NGBA	LDBA (Rab. 4)	LDBA	LDGBA (Rab. 4)	LDGBA	DRA (Rab. 4)	DRA	DGRA (Rab. 4)	DGRA
χ_5	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4 (2)**	**4 (2)**	**4** (2)	**4**
$\overline{\chi_5}$	**8**	**4**	8	**4**	**4**	**4**	**4**	5 **(2)**	**4 (2)**	5 (3)	**4**
χ_6	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5 (2)**	**5 (2)**	**5** (2)	**5**
$\overline{\chi_6}$	**16**	**5**	16	**5**	**5**	**5**	**5**	11 **(2)**	**5 (2)**	11	**5**
χ_8	**23**	**3 (2)**	19 **(2)**	**8**	20	**8**	16 (2)	**5 (2)**	11 (6)	**5 (2)**	7 (3)
$\overline{\chi_8}$	**5**	**5**	**5**	15	**9**	15	**9**	6 (4)	**5 (2)**	6 (4)	**5**
χ_9	**109**	**5** (3)	101 **(2)**	**20**	83	**20**	71 (2)	**15 (2)**	195 (10)	**15 (2)**	137 (9)
$\overline{\chi_9}$	**9**	**8**	9	43	**17**	43	**17**	26 (8)	**13 (2)**	26 (9)	**13**
χ_{11}	**25**	**18**	25	**27**	38	**27**	38	**18 (2)**	36 **(2)**	**18** (2)	36
$\overline{\chi_{11}}$	**6**	7	**6**	23	**20**	23	**20**	**17 (2)**	**17 (2)**	**17** (2)	**17**
χ_{12}	**66**	**42**	66	**67**	112	**67**	112	**42 (2)**	115 **(2)**	**42** (2)	115
$\overline{\chi_{12}}$	**8**	9	**8**	49	**45**	49	**45**	**41 (2)**	**41 (2)**	**41** (2)	**41**
χ_{15}	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**1 (2)**	**1 (2)**	**1**	**1**
$\overline{\chi_{15}}$	**5**	**1**	5	**6**	**6**	**6**	**6**	**1 (10)**	**1 (10)**	**1 (10)**	**1 (10)**
χ_{18}	**6**	**1 (5)**	2 **(5)**	**6**	**6**	**2 (5)**	**2 (5)**	**5 (2)**	**5 (2)**	**1 (6)**	**1 (6)**
$\overline{\chi_{18}}$	**10**	**6**	10	**6**	**6**	**6**	**6**	**1** (10)	5 **(2)**	**1** (5)	5
χ_{22}	**8**	8 (5)	**6 (3)**	**6**	7	**4 (3)**	5 **(3)**	**3** (6)	7 **(4)**	**1** (10)	3 **(5)**
$\overline{\chi_{22}}$	**7**	9 (4)	**6 (2)**	**5**	12	**4 (2)**	11 **(2)**	**2** (6)	14 **(4)**	**1** (7)	13 **(5)**
χ_{23}	**11**	10 (6)	**8 (4)**	**8**	12	**5 (4)**	9 **(4)**	**4** (8)	21 **(4)**	**1** (20)	7 **(6)**
$\overline{\chi_{23}}$	**10**	11 (5)	**8 (3)**	**7**	26	**5 (3)**	23 **(3)**	**3** (8)	57 **(4)**	**1** (10)	45 **(6)**
χ_{24}	**14**	12 (7)	**10 (5)**	**10**	21	**6 (5)**	17 **(5)**	**5** (10)	56 **(4)**	**1** (34)	15 **(7)**
$\overline{\chi_{24}}$	**13**	13 (6)	**10 (4)**	**9**	54	**6 (4)**	47 **(4)**	**4** (10)	201 **(4)**	**1** (13)	145 **(7)**
χ_{26}	**20**	27 (6)	**16 (3)**	**13**	**13**	**9 (3)**	**9 (3)**	**6 (10)**	10 **(10)**	**1 (13)**	**1 (13)**
$\overline{\chi_{26}}$	**6**	7 (6)	**6**	**4**	**4**	**4**	**4**	**1 (6)**	**1 (6)**	**1** (8)	**1 (6)**
χ_{27}	**44**	81 (8)	**32 (4)**	**29**	**29**	**17 (4)**	**17 (4)**	96 **(16)**	**58 (16)**	**1 (24)**	**1 (24)**
$\overline{\chi_{27}}$	**8**	9 (8)	**8**	**5**	**5**	**5**	**5**	**1 (8)**	**1 (8)**	**1** (11)	**1 (8)**
χ_{29}	**8**	27	**8**	**17**	**17**	**17**	**17**	25 **(4)**	**16 (4)**	25 **(4)**	**16 (4)**
$\overline{\chi_{29}}$	**15**	**15**	**15**	**15**	**15**	**15**	**15**	25 **(2)**	**15 (2)**	25	**15**
χ_{30}	**10**	81	**10**	**33**	**33**	**33**	**33**	65 **(4)**	**42 (4)**	65 **(4)**	**42 (4)**
$\overline{\chi_{30}}$	**31**	**31**	**31**	**31**	**31**	**31**	**31**	65 **(2)**	**31 (2)**	65	**31**
χ_{32}	**10**	**8**	10	**8**	**8**	**8**	**8**	**5** (6)	7 **(2)**	**5** (3)	7
$\overline{\chi_{32}}$	**7**	**6** (3)	**6 (2)**	**8**	**8**	**7 (2)**	**7 (2)**	15 **(2)**	**7 (2)**	**5** (4)	6 **(3)**
χ_{33}	**19**	**16**	19	**16**	**16**	**16**	**16**	19 (8)	**15 (2)**	19 (4)	**15**
$\overline{\chi_{33}}$	**11**	30 (4)	**10 (2)**	**16**	**16**	**15 (2)**	**15 (2)**	61 **(2)**	**15 (2)**	19 (5)	**14 (3)**

χ_5	$a\mathbf{U}(b\mathbf{U}(c\mathbf{U}d))$
χ_6	$a\mathbf{U}(b\mathbf{U}(c\mathbf{U}(d\mathbf{U}e)))$
χ_8	$\mathbf{G}(\overline{a} \vee a\mathbf{U}(b \wedge b\mathbf{U}c))$
χ_9	$\mathbf{G}(\overline{a} \vee a\mathbf{U}(b \wedge b\mathbf{U}(c \wedge c\mathbf{U}d)))$
χ_{11}	$(\mathbf{F}a \vee \mathbf{G}b) \wedge (\mathbf{F}b \vee \mathbf{G}c) \wedge (\mathbf{F}c \vee \mathbf{G}d)$
χ_{12}	$(\mathbf{F}a \vee \mathbf{G}b) \wedge (\mathbf{F}b \vee \mathbf{G}c) \wedge (\mathbf{F}c \vee \mathbf{G}d) \wedge (\mathbf{F}d \vee \mathbf{G}e)$
χ_{15}	$\mathbf{FG}a \wedge \mathbf{FG}b \wedge \mathbf{FG}c \wedge \mathbf{FG}d \wedge \mathbf{FG}e$
χ_{18}	$\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{GF}c \wedge \mathbf{GF}d \wedge \mathbf{GF}e$
χ_{22}	$(\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{GF}c) \vee (\mathbf{FG}\overline{c} \wedge (\mathbf{FG}\overline{a} \vee \mathbf{FG}\overline{b}))$
χ_{23}	$(\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{GF}c \wedge \mathbf{GF}d) \vee (\mathbf{FG}\overline{c} \wedge (\mathbf{FG}\overline{a} \vee \mathbf{FG}\overline{b} \vee \mathbf{FG}\overline{d}))$
χ_{24}	$(\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{GF}c \wedge \mathbf{GF}d \wedge \mathbf{GF}e) \vee (\mathbf{FG}\overline{c} \wedge (\mathbf{FG}\overline{a} \vee \mathbf{FG}\overline{b} \vee \mathbf{FG}\overline{d} \vee \mathbf{FG}\overline{e}))$
χ_{26}	$(\mathbf{FG}a \vee \mathbf{GF}b) \wedge (\mathbf{FG}c \vee \mathbf{GF}a) \wedge (\mathbf{FG}d \vee \mathbf{GF}c)$
χ_{27}	$(\mathbf{FG}a \vee \mathbf{GF}b) \wedge (\mathbf{FG}c \vee \mathbf{GF}a) \wedge (\mathbf{FG}d \vee \mathbf{GF}c) \wedge (\mathbf{FG}e \vee \mathbf{GF}d)$
χ_{29}	$\mathbf{GF}(a \wedge \mathbf{XXX}a) \vee \mathbf{GF}(\overline{a} \wedge \mathbf{XXX}\overline{a})$
χ_{30}	$\mathbf{GF}(a \wedge \mathbf{XXXX}a) \vee \mathbf{GF}(\overline{a} \wedge \mathbf{XXXX}\overline{a})$
χ_{32}	$\mathbf{FG}(a \vee b) \vee \mathbf{FG}(\overline{a} \vee \mathbf{X}b) \vee \mathbf{FG}(a \vee \mathbf{XX}b)$
χ_{33}	$\mathbf{FG}(a \vee b) \vee \mathbf{FG}(\overline{a} \vee \mathbf{X}b) \vee \mathbf{FG}(a \vee \mathbf{XX}b) \vee \mathbf{FG}(\overline{a} \vee \mathbf{XXX}b)$

Table 8.5: Automaton sizes for the parametrised formula set (1/2). $x\,(y)$ denotes that the number of states is x and the number of (non-trivial) acceptance sets is y.

8.A Omitted Results

The following tables contain results that have been moved to the appendix of this chapter because either only minor differences in size are present or the formulas are from [EH00] and [SB00].

LTL	NBA	NGBA (Spot)	NGBA	LDBA (Rab. 4)	LDBA	LDGBA (Rab. 4)	LDGBA	DRA (Rab. 4)	DRA	DGRA (Rab. 4)	DGRA
ξ_6	**4**	5	**4**	7	**3**	7	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**
$\overline{\xi_6}$	**5**	8	**5**	**5**	**5**	**5**	**5**	**4 (2)**	**4 (2)**	**4** (2)	**4**
ξ_8	**4**	5 (4)	**4**	**3**	**3**	**3**	**3**	**1 (4)**	**1 (4)**	**1 (4)**	**1 (4)**
$\overline{\xi_8}$	**5**	9 (4)	**4 (2)**	**4**	**4**	**3 (2)**	**3 (2)**	**2 (4)**	**2 (4)**	**1 (4)**	**1 (4)**
ξ_{12}	**12**	**7** (2)	12	10	**8**	10	**8**	6 **(2)**	**5** (4)	6 **(2)**	**5** (3)
$\overline{\xi_{12}}$	**2**	4	**2**	7	**6**	7	**6**	**5 (2)**	**5 (2)**	**5**	**5**
ξ_{17}	**107**	**18 (2)**	94 **(2)**	**9**	21	**9**	18 (2)	**5 (2)**	8 **(2)**	**5** (3)	6
$\overline{\xi_{17}}$	**11**	**5** (2)	11	**6**	7	**6**	7	**5 (2)**	**5 (2)**	**5**	**5**
ξ_{18}	**10**	**4 (2)**	9 **(2)**	**8**	9	**7 (2)**	8 **(2)**	8 **(2)**	**5 (2)**	**4 (3)**	**4** (4)
$\overline{\xi_{18}}$	**10**	17	**10**	**6**	**6**	**6**	**6**	**4** (4)	5 **(2)**	**4** (2)	5

ξ_6	$(\mathbf{F}a \wedge \mathbf{G}\overline{b}) \vee (\mathbf{F}b \wedge \mathbf{G}\overline{a})$
ξ_8	$(\mathbf{FG}\overline{a} \wedge \mathbf{GF}b) \vee (\mathbf{FG}\overline{b} \wedge \mathbf{GF}a)$
ξ_{12}	$\mathbf{G}(\overline{a} \vee \mathbf{F}b) \wedge ((\mathbf{X}a)\mathbf{U}b \vee \mathbf{X}(\overline{a}\mathbf{R}(\overline{a} \vee \overline{b})))$
ξ_{17}	$(a \wedge \mathbf{X}b)\mathbf{R}(\mathbf{X}(((c\mathbf{U}d)\mathbf{R}a)\mathbf{U}(c\mathbf{R}a)))$
ξ_{18}	$\mathbf{G}a \vee \mathbf{G}b \vee ((\mathbf{G}a \vee \mathbf{GF}c) \wedge (\mathbf{G}b \vee \mathbf{GF}\overline{c}))$

Table 8.6: Automaton sizes for the formula set from [SB00] (1/2). $x\,(y)$ denotes that the number of states is x and the number of (non-trivial) acceptance sets is y.

LTL	NBA	NGBA (Spot)	NGBA	LDBA (Rab. 4)	LDBA	LDGBA (Rab. 4)	LDGBA	DRA (Rab. 4)	DRA	DGRA (Rab. 4)	DGRA
ζ_3	**7**	**7**	**7**	**12**	**12**	**12**	**12**	**12 (2)**	**12 (2)**	**12** (2)	**12**
$\overline{\zeta_3}$	**20**	**12**	20	**12**	**12**	**12**	**12**	16 (6)	**12 (2)**	16 (10)	**12**
ζ_{10}	**6**	**1 (5)**	2 **(5)**	**6**	**6**	**2 (5)**	**2 (5)**	**5 (2)**	**5 (2)**	**1 (6)**	**1 (6)**
$\overline{\zeta_{10}}$	**10**	**6**	10	**6**	**6**	**6**	**6**	**1** (10)	5 **(2)**	**1** (5)	5
ζ_{11}	**7**	**2**	7	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2**	**2**
$\overline{\zeta_{11}}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2**	**2**
ζ_{12}	**27**	**4**	27	**8**	11	**8**	11	**8 (4)**	12 (6)	**8 (6)**	12 **(6)**
$\overline{\zeta_{12}}$	**13**	**6** (3)	12 **(2)**	**9**	10	**8 (2)**	9 **(2)**	8 **(2)**	**7 (2)**	**6 (3)**	**6** (4)

ζ_3	$a\mathbf{U}(b \wedge \mathbf{X}(c \wedge \mathbf{F}(d \wedge \mathbf{XF}(e \wedge \mathbf{XF}(f \wedge \mathbf{XF}g)))))$
ζ_{10}	$\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{GF}c \wedge \mathbf{GF}d \wedge \mathbf{GF}e$
ζ_{11}	$a\mathbf{U}(b\mathbf{U}c) \vee b\mathbf{U}(c\mathbf{U}a) \vee c\mathbf{U}(a\mathbf{U}b)$
ζ_{12}	$\mathbf{G}(a \vee b\mathbf{U}(\mathbf{G}c \vee \mathbf{G}d))$

Table 8.7: Automaton sizes for the formula set from [EH00] (1/2). $x\,(y)$ denotes that the number of states is x and the number of (non-trivial) acceptance sets is y.

LTL	NBA	NGBA (Spot)	NGBA	LDBA (Rab. 4)	LDBA	LDGBA (Rab. 4)	LDGBA	DRA (Rab. 4)	DRA	DGRA (Rab. 4)	DGRA
φ_1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1 (2)**	**1 (2)**	**1**	**1**
$\overline{\varphi_1}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2** (2)	**2** (2)	**2** (2)	**2**
φ_2	**3**	4	**3**	**4**	**4**	**4**	**4**	**3 (2)**	**3 (2)**	**3**	**3**
$\overline{\varphi_2}$	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**
φ_3	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2**	**2**
$\overline{\varphi_3}$	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**
φ_5	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2** (4)	**2 (2)**	**2** (3)	**2**
$\overline{\varphi_5}$	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**
φ_6	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2**	**2**
$\overline{\varphi_6}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2** (2)	**2**
φ_7	**4**	5	**4**	**4**	**4**	**4**	**4**	**3 (2)**	**3 (2)**	**3** (2)	**3**
$\overline{\varphi_7}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2**	**2**
φ_8	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2** (4)	**2 (2)**	**2** (3)	**2**
$\overline{\varphi_8}$	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**
φ_9	**5**	**2**	5	**4**	5	**4**	5	**2 (2)**	**2** (4)	**2 (2)**	**2** (3)
$\overline{\varphi_9}$	**3**	**3**	**3**	**5**	**5**	**5**	**5**	**3 (2)**	**3 (2)**	**3**	**3**
φ_{10}	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5 (2)**	**5 (2)**	**5** (3)	**5**
$\overline{\varphi_{10}}$	**7**	**6**	7	**6**	**6**	**6**	**6**	**6 (2)**	**6 (2)**	**6** (2)	**6**
φ_{11}	**7**	8	**7**	**8**	**8**	**8**	**8**	**7 (2)**	**7 (2)**	**7**	**7**
$\overline{\varphi_{11}}$	**8**	**7**	8	**7**	**7**	**7** (2)	**7**	8 **(2)**	**7 (2)**	8 (2)	**7**
φ_{12}	**7**	**6**	7	**7**	**7**	**7**	**7**	**6 (2)**	**6 (2)**	**6** (3)	**6**
$\overline{\varphi_{12}}$	**7**	**7**	**7** (2)	**7**	**7**	**7**	**7** (2)	**7 (2)**	**7 (2)**	**7** (2)	**7**
φ_{15}	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2**	**2**
$\overline{\varphi_{15}}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2** (2)	**2**
φ_{16}	**3**	4	**3**	**4**	**4**	**4**	**4**	**3 (2)**	**3 (2)**	**3**	**3**
$\overline{\varphi_{16}}$	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**

φ_1	$\mathbf{G}a$
φ_2	$\mathbf{G}\overline{a} \vee b\mathbf{U}a$
φ_3	$\mathbf{G}(a \vee \mathbf{G}b)$
φ_5	$\mathbf{G}(a \vee b \vee c\mathbf{W}b)$
φ_6	$a\mathbf{W}(a \wedge b)$
φ_7	$\mathbf{G}\overline{a} \vee \mathbf{F}(a \wedge \mathbf{F}b)$
φ_8	$\mathbf{G}(a \vee b \vee \overline{b}\mathbf{W}(\overline{b} \wedge c))$
φ_9	$\mathbf{G}(a \vee b \vee \overline{b}\mathbf{U}(\overline{b} \wedge c))$
φ_{10}	$\overline{a}\mathbf{W}(a\mathbf{W}(\overline{a}\mathbf{W}(a\mathbf{W}(\mathbf{G}\overline{a}))))$
φ_{11}	$\mathbf{G}\overline{a} \vee (\overline{a} \wedge \overline{b})\mathbf{U}(a \vee (\overline{a} \wedge b)\mathbf{U}(a \vee (\overline{a} \wedge \overline{b})\mathbf{U}(a \vee (\overline{a} \wedge b)\mathbf{U}(a \vee \overline{b}\mathbf{U}a))))$
φ_{12}	$\mathbf{G}\overline{a} \vee \overline{a}\mathbf{U}(a \wedge \overline{b}\mathbf{W}(b\mathbf{W}(\overline{b}\mathbf{W}(b\mathbf{W}(\mathbf{G}\overline{b})))))$
φ_{15}	$a\mathbf{W}b$
φ_{16}	$\mathbf{G}\overline{a} \vee b\mathbf{U}(a \vee c)$

Table 8.8: Automaton sizes for the 'Dwyer'-formula set (3/4). $x\,(y)$ denotes that the number of states is x and the number of (non-trivial) acceptance sets is y.

LTL	NBA	NGBA (Spot)	NGBA	LDBA (Rab. 4)	LDBA	LDGBA (Rab. 4)	LDGBA	DRA (Rab. 4)	DRA	DGRA (Rab. 4)	DGRA
φ_{17}	**4**	5	**4**	**7**	**7**	**7**	**7**	**4 (2)**	**4 (2)**	**4** (3)	**4**
$\overline{\varphi_{17}}$	**6**	**3** (2)	6	**5**	6	**5**	6	**3 (2)**	**3** (4)	**3 (2)**	**3** (3)
φ_{19}	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2** (4)	**2 (2)**	**2** (3)	**2**
$\overline{\varphi_{19}}$	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**
φ_{20}	**4**	**2**	4	**4**	**4**	**4**	**4**	**2 (2)**	**2** (4)	**2 (2)**	**2** (3)
$\overline{\varphi_{20}}$	**2**	**2**	**2**	4	**3**	4	**3**	**2 (2)**	**2 (2)**	**2**	**2**
φ_{21}	**4**	5	**4**	**4**	**4**	**4**	**4**	**3 (2)**	**3 (2)**	**3**	**3**
$\overline{\varphi_{21}}$	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3**	**3**
φ_{22}	**6**	**3**	6	**6**	**6**	**6**	**6**	4 **(2)**	**3** (4)	4 **(3)**	**3 (3)**
$\overline{\varphi_{22}}$	**3**	**3**	**3**	6	**4**	6	**4**	**3 (2)**	**3 (2)**	**3**	**3**
φ_{25}	**4**	**3**	4	**4**	**4**	**4**	**4**	**3 (2)**	**3 (2)**	**3**	**3**
$\overline{\varphi_{25}}$	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3**	**3**
φ_{26}	**4**	5	**4**	**5**	**5**	**5**	**5**	**4 (2)**	**4 (2)**	**4**	**4**
$\overline{\varphi_{26}}$	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4 (2)**	**4 (2)**	**4** (2)	**4**
φ_{30}	**4**	5	**4**	**5**	**5**	**5**	**5**	**3** (4)	**3 (2)**	**3** (5)	**3**
$\overline{\varphi_{30}}$	**4**	**4**	**4**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**
φ_{31}	**5**	6	**5**	**8**	**8**	**8**	**8**	**5 (2)**	**5 (2)**	**5** (4)	**5**
$\overline{\varphi_{31}}$	**4**	**4**	**4**	**5**	**5**	**5**	**5**	**5 (2)**	**5 (2)**	**5** (2)	**5**
φ_{32}	**6**	**5**	6	8	**7**	8	**7**	**4 (2)**	5 **(2)**	**4** (4)	5
$\overline{\varphi_{32}}$	**5**	**5**	**5** (2)	**7**	**7**	7	**6** (2)	**4 (2)**	**4 (2)**	**4** (2)	**4**
φ_{40}	**7**	**4 (2)**	6 **(2)**	8	**6**	8	**5** (2)	**4 (2)**	6 (4)	4 **(2)**	**3** (4)
$\overline{\varphi_{40}}$	**3**	**3**	**3**	6	**5**	6	**5**	**3** (4)	5 **(2)**	**3** (2)	5
φ_{41}	**6**	7	**6**	**5**	**5**	**5**	**5**	**4 (2)**	**4 (2)**	**4**	**4**
$\overline{\varphi_{41}}$	**4**	**4**	**4**	**4**	**4**	**4**	**4**	6 **(2)**	**4 (2)**	6	**4**
φ_{46}	**6**	7	**6**	**5**	**5**	**5**	**5**	**4 (2)**	**4 (2)**	**4**	**4**
$\overline{\varphi_{46}}$	**4**	**4**	**4**	**4**	**4**	**4**	**4**	6 **(2)**	**4 (2)**	6	**4**

φ_{17}	$\mathbf{G}\overline{a} \vee \mathbf{F}(a \wedge b\mathbf{W}c)$
φ_{19}	$\mathbf{G}(a \vee b \vee c\mathbf{W}(b \vee d))$
φ_{20}	$\mathbf{G}(a \vee \mathbf{F}b)$
φ_{21}	$\mathbf{G}\overline{a} \vee (b \vee \overline{a}\mathbf{U}(\overline{a} \wedge c))\mathbf{U}a$
φ_{22}	$\mathbf{G}(a \vee \mathbf{G}(b \vee \mathbf{F}c))$
φ_{25}	$\mathbf{G}a \vee a\mathbf{U}(a \wedge b \wedge \mathbf{X}(a\mathbf{U}c))$
φ_{26}	$\mathbf{G}\overline{a} \vee b\mathbf{U}(a \vee (b \wedge c \wedge \mathbf{X}(b\mathbf{U}d)))$
φ_{30}	$a\mathbf{U}b \vee \mathbf{G}(a \vee \mathbf{X}\mathbf{G}c)$
φ_{31}	$\mathbf{G}\overline{a} \vee (a \vee b \vee \mathbf{X}(a\mathbf{R}(a \vee c)))\mathbf{U}(a \vee d)$
φ_{32}	$\mathbf{G}\overline{a} \vee \overline{a}\mathbf{U}(a \wedge (b\mathbf{U}c \vee \mathbf{G}(b \vee \mathbf{X}\mathbf{G}d)))$
φ_{40}	$\mathbf{G}(a \vee \mathbf{F}(b \wedge \mathbf{X}\mathbf{F}c))$
φ_{41}	$\mathbf{G}\overline{a} \vee (b \vee \overline{a}\mathbf{U}(\overline{a} \wedge c \wedge \mathbf{X}(\overline{a}\mathbf{U}d)))\mathbf{U}a$
φ_{46}	$\mathbf{G}\overline{a} \vee (b \vee \overline{a}\mathbf{U}(\overline{a} \wedge c \wedge d \wedge \mathbf{X}((\overline{a} \wedge d)\mathbf{U}e)))\mathbf{U}a$

Table 8.9: Automaton sizes for the 'Dwyer'-formula set (4/4). $x\,(y)$ denotes that the number of states is x and the number of (non-trivial) acceptance sets is y.

LTL	NBA	NGBA (Spot)	NGBA	LDBA (Rab. 4)	LDBA	LDGBA (Rab. 4)	LDGBA	DRA (Rab. 4)	DRA	DGRA (Rab. 4)	DGRA
ψ_1	**4**	**2**	4	**4**	**4**	**4**	**4**	**2 (2)**	**2** (4)	**2 (2)**	**2** (3)
$\overline{\psi_1}$	**2**	**2**	**2**	4	**3**	4	**3**	**2 (2)**	**2 (2)**	**2**	**2**
ψ_2	**5**	**4** (2)	5	**4**	**4**	**4**	**4**	**1** (6)	2 **(4)**	**1** (4)	2 **(3)**
$\overline{\psi_2}$	**3**	**2 (2)**	**2 (2)**	**3**	**3**	**2 (2)**	**2 (2)**	**2 (2)**	**2 (2)**	**1 (3)**	**1 (3)**
ψ_3	**5**	**2**	5	**4**	5	**4**	5	**2 (2)**	**2** (4)	**2 (2)**	**2** (3)
$\overline{\psi_3}$	**3**	**3**	**3**	**5**	**5**	**5**	**5**	**3 (2)**	**3 (2)**	**3**	**3**
ψ_4	**3**	**2**	3	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2** (2)	**2**
$\overline{\psi_4}$	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1 (2)**	**1 (2)**	**1**	**1**
ψ_5	**2**	**1**	2	**3**	**3**	**3**	**3**	**1 (4)**	**1 (4)**	**1 (4)**	**1 (4)**
$\overline{\psi_5}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**1 (2)**	**1 (2)**	**1**	**1**
ψ_6	**4**	**4**	**4**	**6**	7	**6**	7	**4 (2)**	**4 (2)**	**4**	**4**
$\overline{\psi_6}$	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4 (2)**	**4 (2)**	**4** (2)	**4**
ψ_8	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2** (4)	**2 (2)**	**2** (3)	**2**
$\overline{\psi_8}$	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**
ψ_9	**1**	**1**	**1**	**2**	**2**	**2**	**2**	**1 (2)**	**1 (2)**	**1 (2)**	**1 (2)**
$\overline{\psi_9}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**1 (2)**	**1 (2)**	**1**	**1**
ψ_{11}	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2** (4)	**2 (2)**	**2** (3)	**2**
$\overline{\psi_{11}}$	**3**	4	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**
ψ_{12}	**3**	4	**3**	**4**	**4**	**4**	**4**	**3 (2)**	**3 (2)**	**3** (2)	**3**
$\overline{\psi_{12}}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2**	**2**
ψ_{14}	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2** (4)	**2 (2)**	**2** (3)	**2**
$\overline{\psi_{14}}$	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**
ψ_{15}	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2** (4)	**2 (2)**	**2** (3)	**2**
$\overline{\psi_{15}}$	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**
ψ_{16}	**4**	**2**	4	**4**	**4**	**4**	**4**	**2 (2)**	**2** (4)	**2 (2)**	**2** (3)
$\overline{\psi_{16}}$	**2**	**2**	**2**	4	**3**	4	**3**	**2 (2)**	**2 (2)**	**2**	**2**
ψ_{19}	**3**	**3** (2)	**3**	**3**	**3**	**3**	**3**	**1 (4)**	**1 (4)**	**1 (3)**	**1 (3)**
$\overline{\psi_{19}}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**1 (2)**	**1 (2)**	**1 (2)**	**1 (2)**

ψ_1	$\mathbf{G}(a \vee \mathbf{F}b)$
ψ_2	$\mathbf{FG}a \vee \mathbf{FG}b \vee \mathbf{GF}c$
ψ_3	$\mathbf{G}(a \vee (b \wedge c\mathbf{U}d))$
ψ_4	$\mathbf{F}a \vee \mathbf{F}b$
ψ_5	$\mathbf{GF}a \vee \mathbf{GF}b$
ψ_6	$\mathbf{G}a \vee b\mathbf{R}c \vee a\mathbf{U}d$
ψ_8	$\mathbf{G}(a \vee b\mathbf{R}c)$
ψ_9	$\mathbf{GF}a$
ψ_{11}	$(a \vee b)\mathbf{R}c \wedge \mathbf{G}(d \vee (a \vee b)\mathbf{R}c)$
ψ_{12}	$\mathbf{F}a \vee \mathbf{G}b$
ψ_{14}	$\mathbf{G}(a \vee b\mathbf{R}(b \vee c))$
ψ_{15}	$\mathbf{G}(\overline{a} \vee b \vee b\mathbf{R}(a \vee b))$
ψ_{16}	$\mathbf{G}(a \vee \mathbf{F}(b \wedge c))$
ψ_{19}	$\mathbf{FG}a \vee \mathbf{GF}b$

Table 8.10: Automaton sizes for the 'BEEM'-formula set (2/2). $x\,(y)$ denotes that the number of states is x and the number of (non-trivial) acceptance sets is y.

LTL	NBA	NGBA (Spot)	NGBA	LDBA (Rab. 4)	LDBA	LDGBA (Rab. 4)	LDGBA	DRA (Rab. 4)	DRA	DGRA (Rab. 4)	DGRA
χ_1	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4 (2)**	**4 (2)**	**4** (2)	**4**
$\overline{\chi_1}$	**3**	**3**	**3**	**4**	**4**	**4**	**4**	**4 (2)**	**4 (2)**	**4** (3)	**4**
χ_2	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**8 (2)**	**8 (2)**	**8** (2)	**8**
$\overline{\chi_2}$	**4**	**4**	**4**	**8**	**8**	**8**	**8**	**8 (2)**	**8 (2)**	**8** (8)	**8**
χ_3	**16**	**16**	**16**	**16**	**16**	**16**	**16**	**16 (2)**	**16 (2)**	**16** (2)	**16**
$\overline{\chi_3}$	**5**	**5**	**5**	**16**	**16**	**16**	**16**	**16 (2)**	**16 (2)**	**16** (10)	**16**
χ_4	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**
$\overline{\chi_4}$	**4**	**3**	4	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3**	**3**
χ_7	**5**	**2**	5	**4**	5	**4**	5	**2 (2)**	**2** (4)	**2 (2)**	**2** (3)
$\overline{\chi_7}$	**3**	**3**	**3**	**5**	**5**	**5**	**5**	**3 (2)**	**3 (2)**	**3**	**3**
χ_{10}	**9**	**8**	9	**11**	13	**11**	13	**8 (2)**	11 **(2)**	**8** (2)	11
$\overline{\chi_{10}}$	**4**	5	**4**	11	**8**	11	**8**	**7 (2)**	**7 (2)**	**7** (2)	**7**
χ_{13}	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**1 (2)**	**1 (2)**	**1**	**1**
$\overline{\chi_{13}}$	**3**	**1**	3	**4**	**4**	**4**	**4**	**1 (6)**	**1 (6)**	**1 (6)**	**1 (6)**
χ_{14}	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**1 (2)**	**1 (2)**	**1**	**1**
$\overline{\chi_{14}}$	**4**	**1**	4	**5**	**5**	**5**	**5**	**1 (8)**	**1 (8)**	**1 (8)**	**1 (8)**
χ_{16}	**4**	**1 (3)**	2 **(3)**	**4**	**4**	**2 (3)**	**2 (3)**	**3 (2)**	**3 (2)**	**1 (4)**	**1 (4)**
$\overline{\chi_{16}}$	**6**	**4**	6	**4**	**4**	**4**	**4**	**1** (6)	3 **(2)**	**1** (3)	3
χ_{17}	**5**	**1 (4)**	2 **(4)**	**5**	**5**	**2 (4)**	**2 (4)**	**4 (2)**	**4 (2)**	**1 (5)**	**1 (5)**
$\overline{\chi_{17}}$	**8**	**5**	8	**5**	**5**	**5**	**5**	**1** (8)	4 **(2)**	**1** (4)	4
χ_{19}	**5**	**4** (2)	5	**4**	**4**	**4**	**4**	**1** (6)	2 **(4)**	**1** (4)	2 **(3)**
$\overline{\chi_{19}}$	**3**	**2 (2)**	**2 (2)**	**3**	**3**	**2 (2)**	**2 (2)**	**2 (2)**	**2 (2)**	**1 (3)**	**1 (3)**
χ_{20}	**7**	**5** (2)	7	**5**	**5**	**5**	**5**	**1** (8)	3 **(4)**	**1** (5)	3 **(3)**
$\overline{\chi_{20}}$	**4**	**2 (3)**	**2 (3)**	**4**	**4**	**2 (3)**	**2 (3)**	**3 (2)**	**3 (2)**	**1 (4)**	**1 (4)**
χ_{21}	**9**	**6** (2)	9	**6**	**6**	**6**	**6**	**1** (10)	4 **(4)**	**1** (6)	4 **(3)**
$\overline{\chi_{21}}$	**5**	**2 (4)**	**2 (4)**	**5**	**5**	**2 (4)**	**2 (4)**	**4 (2)**	**4 (2)**	**1 (5)**	**1 (5)**
χ_{25}	**9**	9 (4)	**8 (2)**	**6**	**6**	**5 (2)**	**5 (2)**	**2 (6)**	**2 (6)**	**1 (6)**	**1 (6)**
$\overline{\chi_{25}}$	**4**	5 (4)	**4**	**3**	**3**	**3**	**3**	**1 (4)**	**1 (4)**	**1** (5)	**1 (4)**
χ_{28}	**6**	9	**6**	**9**	**9**	**9**	**9**	9 **(4)**	**6 (4)**	9 **(4)**	**6 (4)**
$\overline{\chi_{28}}$	**7**	**7**	**7**	**7**	**7**	**7**	**7**	9 **(2)**	**7 (2)**	9	**7**
χ_{31}	**5**	**4**	5	**4**	**4**	**4**	**4**	**2** (4)	3 **(2)**	**2** (2)	3
$\overline{\chi_{31}}$	**4**	**2 (2)**	3 **(2)**	**4**	**4**	**3 (2)**	**3 (2)**	4 **(2)**	**3 (2)**	**2 (3)**	**2 (3)**

χ_1	$(a\mathbf{U}b)\mathbf{U}c$
χ_2	$((a\mathbf{U}b)\mathbf{U}c)\mathbf{U}d$
χ_3	$(((a\mathbf{U}b)\mathbf{U}c)\mathbf{U}d)\mathbf{U}e$
χ_4	$a\mathbf{U}(b\mathbf{U}c)$
χ_7	$\mathbf{G}(\overline{a} \vee a\mathbf{U}b)$
χ_{10}	$(\mathbf{F}a \vee \mathbf{G}b) \wedge (\mathbf{F}b \vee \mathbf{G}c)$
χ_{13}	$\mathbf{FG}a \wedge \mathbf{FG}b \wedge \mathbf{FG}c$
χ_{14}	$\mathbf{FG}a \wedge \mathbf{FG}b \wedge \mathbf{FG}c \wedge \mathbf{FG}d$
χ_{16}	$\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{GF}c$
χ_{17}	$\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{GF}c \wedge \mathbf{GF}d$
χ_{19}	$\mathbf{FG}a \vee \mathbf{FG}b \vee \mathbf{GF}c$
χ_{20}	$\mathbf{FG}a \vee \mathbf{FG}b \vee \mathbf{FG}c \vee \mathbf{GF}d$
χ_{21}	$\mathbf{FG}a \vee \mathbf{FG}b \vee \mathbf{FG}c \vee \mathbf{FG}d \vee \mathbf{GF}e$
χ_{25}	$(\mathbf{FG}a \vee \mathbf{GF}b) \wedge (\mathbf{FG}c \vee \mathbf{GF}a)$
χ_{28}	$\mathbf{GF}(a \wedge \mathbf{XX}a) \vee \mathbf{GF}(\overline{a} \wedge \mathbf{XX}\overline{a})$
χ_{31}	$\mathbf{FG}(a \vee b) \vee \mathbf{FG}(\overline{a} \vee \mathbf{X}b)$

Table 8.11: Automaton sizes for the parametrised formula set (2/2). $x\,(y)$ denotes that the number of states is x and the number of (non-trivial) acceptance sets is y.

LTL	NBA	NGBA (Spot)	NGBA	LDBA (Rab. 4)	LDBA	LDGBA (Rab. 4)	LDGBA	DRA (Rab. 4)	DRA	DGRA (Rab. 4)	DGRA
ξ_1	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2** (2)	**2**
$\overline{\xi_1}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2**	**2**
ξ_2	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**
$\overline{\xi_2}$	**4**	**3**	4	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3**	**3**
ξ_3	**3**	**3** (2)	**3**	**3**	**3**	**3**	**3**	**1 (4)**	**1 (4)**	**1 (3)**	**1 (3)**
$\overline{\xi_3}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**1 (2)**	**1 (2)**	**1 (2)**	**1 (2)**
ξ_4	**4**	**4**	**4**	**3**	**3**	**3**	**3**	**2 (2)**	**2 (2)**	**2**	**2**
$\overline{\xi_4}$	**5**	**3**	5	5	**4**	5	**4**	**2 (4)**	**2 (4)**	**2 (3)**	**2 (3)**
ξ_5	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4 (2)**	**4 (2)**	**4**	**4**
$\overline{\xi_5}$	**4**	**4**	**4**	**5**	**5**	**5**	**5**	**4 (2)**	**4 (2)**	**4** (2)	**4**
ξ_7	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**1 (2)**	**1 (2)**	**1 (2)**	**1 (2)**
$\overline{\xi_7}$	**3**	**3** (2)	**3**	**3**	**3**	**3**	**3**	**1 (4)**	**1 (4)**	**1 (3)**	**1 (3)**
ξ_9	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2**	**2**
$\overline{\xi_9}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2** (2)	**2**
ξ_{10}	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1 (2)**	**1 (2)**	**1**	**1**
$\overline{\xi_{10}}$	**0**	1	**0**	**0**	**0**	**0**	**0**	1	**0**	1	**0**
ξ_{11}	**5**	**5**	**5**	**4**	5	**4**	5	**3 (2)**	**3 (2)**	**3**	**3**
$\overline{\xi_{11}}$	**0**	2	**0**	**0**	**0**	**0**	**0**	2	**0**	2	**0**
ξ_{13}	**4**	**2**	4	**4**	**4**	**4**	**4**	**2 (2)**	**2** (4)	**2 (2)**	**2** (3)
$\overline{\xi_{13}}$	**2**	**2**	**2**	4	**3**	4	**3**	**2 (2)**	**2 (2)**	**2**	**2**
ξ_{14}	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**
$\overline{\xi_{14}}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2** (6)	**2 (2)**	**2** (6)	**2**
ξ_{15}	**3**	**1 (2)**	2 **(2)**	**3**	**3**	**2 (2)**	**2 (2)**	**2 (2)**	**2 (2)**	**1 (3)**	**1 (3)**
$\overline{\xi_{15}}$	**4**	**3**	4	**3**	**3**	**3**	**3**	**1** (4)	2 **(2)**	**1** (2)	2
ξ_{16}	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4 (2)**	**4 (2)**	**4** (2)	**4**
$\overline{\xi_{16}}$	**2**	3	**2**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3**	**3**
ξ_{19}	**7**	**6**	7	**5**	6	**5**	6	4 **(2)**	**3 (2)**	4 (2)	**3**
$\overline{\xi_{19}}$	**10**	**9** (4)	10	**6**	**6**	**6**	**6**	**4 (4)**	**4 (4)**	**4 (4)**	**4 (4)**
ξ_{20}	**3**	4	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3**	**3**
$\overline{\xi_{20}}$	**5**	**4**	5	**4**	**4**	**4**	**4**	**4 (2)**	**4 (2)**	**4** (2)	**4**
ξ_{21}	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1 (2)**	**1 (2)**	**1**	**1**
$\overline{\xi_{21}}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2** (2)	**2**
ξ_{22}	**3**	**2**	3	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2** (2)	**2**
$\overline{\xi_{22}}$	**3**	**2**	3	**2**	**2**	**2**	**2**	**2 (2)**	**2 (2)**	**2**	**2**

ξ_1	$a\mathbf{U}b$
ξ_2	$a\mathbf{U}(b\mathbf{U}c)$
ξ_3	$\mathbf{FG}a \vee \mathbf{GF}b$
ξ_4	$(\mathbf{F}a)\mathbf{U}(\mathbf{G}b)$
ξ_5	$\mathbf{F}a \wedge (a \vee \mathbf{G}b)$
ξ_7	$\mathbf{FG}a \wedge \mathbf{GF}b$
ξ_9	$a\mathbf{R}(a \vee b)$
ξ_{10}	$\mathbf{tt}$
ξ_{11}	$(\mathbf{X}a)\mathbf{U}b \vee \mathbf{X}(\overline{a}\mathbf{R}(\overline{a} \vee \overline{b}))$
ξ_{13}	$\mathbf{G}(a \vee \mathbf{F}b)$
ξ_{14}	$\mathbf{F}(a \wedge \mathbf{X}(b\mathbf{U}c))$
ξ_{15}	$\mathbf{GF}a \wedge \mathbf{GF}b$
ξ_{16}	$\mathbf{F}a \wedge \mathbf{F}\overline{a}$
ξ_{19}	$\mathbf{G}a \vee \mathbf{G}b \vee ((\mathbf{G}a \vee \mathbf{FG}c) \wedge (\mathbf{G}b \vee \mathbf{FG}\overline{c}))$
ξ_{20}	$\mathbf{G}(a \vee \mathbf{XG}b) \wedge \mathbf{G}(c \vee \mathbf{XG}\overline{b})$
ξ_{21}	$\mathbf{G}a$
ξ_{22}	$a \vee b\mathbf{U}a$

Table 8.12: Automaton sizes for the formula set from [SB00] (2/2). $x\,(y)$ denotes that the number of states is x and the number of (non-trivial) acceptance sets is y.

LTL	NBA	NGBA (Spot)	NGBA	LDBA (Rab. 4)	LDBA	LDGBA (Rab. 4)	LDGBA	DRA (Rab. 4)	DRA	DGRA (Rab. 4)	DGRA
ζ_1	**2**	**2**	**2**	5	**4**	5	**4**	**3 (2)**	**3 (2)**	**3**	**3**
$\overline{\zeta_1}$	**7**	**4**	7	**6**	7	**6**	7	**4 (2)**	5 (4)	**4 (2)**	5 (3)
ζ_2	**3**	**3**	**3**	**4**	**4**	**4**	**4**	**4 (2)**	**4 (2)**	**4** (2)	**4**
$\overline{\zeta_2}$	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4** (6)	**4 (2)**	**4** (6)	**4**
ζ_4	**2**	**2**	**2**	**3**	**3**	**3**	**3**	**2 (2)**	**2 (2)**	**2**	**2**
$\overline{\zeta_4}$	**4**	**2**	4	5	**4**	5	**4**	**2 (2)**	**2** (4)	**2 (2)**	**2** (3)
ζ_5	**4**	**4**	**4**	**5**	**5**	**5**	**5**	**5 (2)**	**5 (2)**	**5** (2)	**5**
$\overline{\zeta_5}$	**4**	**3**	4	**4**	**4**	**4**	**4**	**4 (2)**	**4 (2)**	**4** (4)	**4**
ζ_6	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3 (2)**	**3 (2)**	**3** (2)	**3**
$\overline{\zeta_6}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2** (6)	**2 (2)**	**2** (6)	**2**
ζ_7	**3**	**3** (2)	**3**	**3**	**3**	**3**	**3**	**1 (4)**	**1 (4)**	**1 (3)**	**1 (3)**
$\overline{\zeta_7}$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**1 (2)**	**1 (2)**	**1 (2)**	**1 (2)**
ζ_8	**5**	**2**	5	**4**	5	**4**	5	**2 (2)**	**2** (4)	**2 (2)**	**2** (3)
$\overline{\zeta_8}$	**3**	**3**	**3**	**5**	**5**	**5**	**5**	**3 (2)**	**3 (2)**	**3**	**3**
ζ_9	**4**	**1 (3)**	2 **(3)**	**4**	**4**	**2 (3)**	**2 (3)**	**3 (2)**	**3 (2)**	**1 (4)**	**1 (4)**
$\overline{\zeta_9}$	**8**	**5**	8	**5**	**5**	**5**	**5**	**2** (8)	4 **(2)**	**2** (5)	4

ζ_1	$a\mathbf{U}(b \wedge \mathbf{G}c)$
ζ_2	$a\mathbf{U}(b \wedge \mathbf{X}(c\mathbf{U}d))$
ζ_4	$\mathbf{F}(a \wedge \mathbf{XG}b)$
ζ_5	$\mathbf{F}(a \wedge \mathbf{X}(b \wedge \mathbf{XF}c))$
ζ_6	$\mathbf{F}(a \wedge \mathbf{X}(b\mathbf{U}c))$
ζ_7	$\mathbf{FG}a \vee \mathbf{GF}b$
ζ_8	$\mathbf{G}(a \vee b\mathbf{U}c)$
ζ_9	$\mathbf{G}a \wedge \mathbf{GF}b \wedge \mathbf{GF}c \wedge \mathbf{GF}d$

Table 8.13: Automaton sizes for the formula set from [EH00] (2/2). $x\,(y)$ denotes that the number of states is x and the number of (non-trivial) acceptance sets is y.

9 Applications

In this chapter, we take a brief look at two applications of the limit-deterministic Büchi automata (LDBA) obtained through the unified construction (and the deterministic constructions for fragments) to model checking and synthesis. They are suitable due to their special structure for quantitive model checking of probabilistic systems where arbitrary LDBAs in general cannot be used. Furthermore, we focus here on the lesser known LDBAs and skip applications of nondeterministic Büchi automata and deterministic Rabin automata, since their applications are well-known and extensively studied in the existing literature.

9.1 Probabilistic Model Checking

The problem of model checking probabilistic systems against an LTL specification [BK08] is to determine the probability that an LTL formula φ holds on the infinite paths generated by a given Markov chain $\mathcal{M}$, written $\mathsf{Pr}^{\mathcal{M}}(\varphi)$, or more generally, for a Markov decision process $\mathcal{M}$ to determine the *maximal* probability[1] that φ is satisfied, written $\sup_{\mathfrak{S}} \mathsf{Pr}^{\mathcal{M}_{\mathfrak{S}}}(\varphi)$, where $\mathfrak{S}$ ranges over *strategies* resolving the nondeterminism of $\mathcal{M}$, and $\mathcal{M}_{\mathfrak{S}}$ is the Markov chain resulting from the application of $\mathfrak{S}$ to $\mathcal{M}$.[2] Probabilistic model checking has two commonly discussed variants: The first variant is *qualitative* probabilistic model checking, where one is interested in whether the satisfaction probability is 0, 1, or neither. The second variant is *quantitative* probabilistic model checking, where one is interested in the exact satisfaction probability.

The automata-theoretic approach to quantitative model checking of Markov decision processes (MDPs) proceeds in the following steps: (1) construct the product $\mathcal{M} \otimes \mathcal{A}$ of the given system $\mathcal{M}$ and an automaton $\mathcal{A}$ for the given LTL formula, (2) find all *maximal end components* (MECs) in the product, (3) determine which MECs are accepting, and (4) compute the maximal probability to reach the accepting MECs. However, as opposed to the non-probabilistic model checking case, in general the automaton $\mathcal{A}$ cannot be used if it is nondeterministic. Intuitively, resolving nondeterminism of the automaton may depend on the yet unknown, probabilistically given future.

In the qualitative case it is known [Var85; CY95] that LDBAs can be used and a fully deterministic automaton is not necessary, but it was unclear until [HLS+15; SEJK16] if in the quantitative case deterministic automata could also be replaced by LDBAs. The advantages of LDBAs compared to DRAs are that they are easier to construct, they can be considerably smaller, and determining for a MEC if it is accepting for Büchi conditions does not require an iterative analysis as it is necessary for Rabin conditions.

[1] One can also consider the minimal probability, but there are no substantial differences in the techniques.

[2] We give a short overview of notation and results on Markov chains and MDPs relevant for this part in Section 9.A.

We now show that *LDBAs of Theorem 6.2* can be used in the *quantitative* setting by slightly adapting the product construction of (1). But be aware of the fact that *general LDBAs* are *not* usable in such a way for quantitative probabilistic model checking and this only works because of the special structure of the automata from Theorem 6.2.

The key insight is that all by $\mathcal{M}_{\mathfrak{S}}$ generated words are ν- and μ-stable once a *bottom strongly connected component* (BSCC) is reached. Thus we can extend the strategy $\mathfrak{S}$ in such a way that it resolves the nondeterminism of the LDBA by moving to the accepting component once a BSCC is reached.

Definition 9.1. *Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, Ap, L)$ be a finite Markov decision process, let φ be a formula over the set of atomic propositions Ap, and let $\mathcal{A} = (Q, \delta, q_0, \beta)$ be the LDBA for φ obtained from Theorem 6.2. We denote the states in the initial component by $Q_{init} = Reach(\varphi)_{/\sim} \subseteq Q$, and denote the states in the accepting component for some $X \subseteq \mu(\varphi)$ and $Y \subseteq \nu(\varphi)$ by $Q_{X,Y} \subseteq Q$. Further, we abbreviate the deterministic transition relation within Q_{init} with δ_{init} and we abbreviate the deterministic transition relation within the accepting component with δ_{acc}. We then extend the set of actions by using X and Y as subscripts:*

$$Act' \coloneqq Act \uplus \{\alpha_{X,Y} : \alpha \in Act, X \subseteq \mu(\varphi), Y \subseteq \nu(\varphi)\}$$

Finally, the product MDP *is defined as follows*

$$\mathcal{M} \otimes \mathcal{A} \coloneqq \left(S \times Q, Act', \mathbf{P}', \iota_{init}, Q, L'\right)$$

where we define for all states $\langle s, q\rangle, \langle s', q'\rangle \in S \times Q$, actions $\alpha \in Act$, and sets $X \subseteq \mu(\varphi)$ and $Y \subseteq \nu(\varphi)$:

- $\mathbf{P}'\left(\langle s, q\rangle, \alpha, \langle s', q'\rangle\right) \coloneqq \begin{cases} \mathbf{P}(s, \alpha, s') & \text{if } q' = \delta_{init}(q, L(s')) \text{ or } q' = \delta_{acc}(q, L(s')) \\ 0 & \text{otherwise} \end{cases}$
- $\mathbf{P}'\left(\langle s, q\rangle, \alpha_{X,Y}, \langle s', q'\rangle\right) \coloneqq \begin{cases} \mathbf{P}(s, \alpha, s') & \text{if } q \in Q_{init} \text{ and } q' \in \delta(q, L(s')) \cap Q_{X,Y}{}^{3} \\ 0 & \text{otherwise} \end{cases}$
- $\iota'_{init}\left(\langle s, q\rangle\right) \coloneqq \begin{cases} \iota_{init}(s) & \text{if } q = \delta_{init}(q_0, L(s)) \\ 0 & \text{otherwise} \end{cases}$
- $L'\left(\langle s, q\rangle\right) \coloneqq \{q\}$

Observe that – opposed to building a product MDP $\mathcal{M} \otimes \mathcal{D}$ with a DRA $\mathcal{D}$ – we add additional nondeterminism which needs to be resolved by a strategy. In the DRA case we have a one-to-one correspondence of strategies for $\mathcal{M}$ and $\mathcal{M} \otimes \mathcal{D}$. Given a strategy for one MDP, we can always define a new strategy for the other MDP by means of projection (ignoring the automaton states) and by deterministic tracking of the DRA for the formula φ, respectively. Hence every strategy for $\mathcal{M}$ or $\mathcal{M} \otimes \mathcal{D}$ is part of a pair $\mathfrak{S}$ and $\mathfrak{S}'$ such that

$$\mathsf{Pr}^{\mathcal{M}_{\mathfrak{S}}}(s \models \varphi) = \mathsf{Pr}^{(\mathcal{M} \otimes \mathcal{D})_{\mathfrak{S}'}}\left(\langle s, \delta(q_0, L(s)\rangle \models \bigvee_{0<i\leq k} (\Diamond\Box\neg L_i \wedge \Box\Diamond K_i)\right)$$

[3] Notice that this intersection is either empty or a singleton set.

Thus we reduce the question about φ to a question of repeatedly reaching (and avoiding) states in the product MDP. While we lose this one-to-one correspondence in the LDBA setting, we can prove a slight variation that is sufficient for quantitive model checking:

Theorem 9.2 ([SEJK16])**.** *Let φ be a formula, and let $\mathcal{A} = \mathcal{A}_{\mathsf{LDBA}}(\varphi)$ be the LDBA from Theorem 6.2 with a set of accepting states F. Let $\mathcal{M}$ be a finite MDP with a set of states S, and let $Acc = S \times F$ be the set of accepting states. Then for any state $s \in S$ we have:*

$$\sup_{\mathfrak{S}} \mathsf{Pr}^{\mathcal{M}_{\mathfrak{S}}}(s \models \varphi) = \sup_{\mathfrak{S}} \mathsf{Pr}^{(\mathcal{M}\otimes\mathcal{A})_{\mathfrak{S}}}(\langle s, \delta_{init}(q_0, L(s))\rangle \models \Box\Diamond Acc)$$

Proof Sketch. ($\geq$) This direction is analogous to the DRA case. Indeed, every strategy over $\mathcal{M} \otimes \mathcal{A}$ induces by projection a strategy over $\mathcal{M}$. Let $\rho = \langle s_0, r_0\rangle\langle s_1, r_1\rangle \dots$ be a path. We then define the path $\rho' = s_0 s_1 \dots$ on $\mathcal{M}$ and the run $r = r_0 r_1 \dots$ on $\mathcal{A}$. Observe that whenever a path ρ is in $\Box\Diamond Acc$, then r is accepting and thus ρ' is a path satisfying φ. Thus the probability on the left-hand side is lower-bounded by the probability on the right-hand side.

($\leq$) We begin by formalising our previous intuition that we have ν- and μ-stability once a BSCC is reached. For this we define a random variable *index* which maps a path ρ of $\mathcal{M}$ to the minimal number of steps until stability is reached. Further we denote the trace belonging to the path ρ by w_ρ. Thus we have $(w_\rho)[i] = L(\rho[i])$ for all i. Formally we define *index* as follows:

$$index(\rho) \coloneqq \min\{i : (w_\rho)_i \text{ is } \nu\text{- and } \mu\text{-stable with respect to } \varphi\}$$

We call a state s of a Markov chain $\mathcal{M}'$ *decided* if almost surely every path ρ starting in s has $index(\rho) = 0$. In other words, almost surely every path of $\mathcal{M}'$ starting in s is ν- and μ-stable with respect to φ.

- Claim: Let C be a BSCC of a Markov chain $\mathcal{M}'$. Then all states of C are decided and almost surely all paths ρ have the same unique $X = \mathcal{GF}^{\varphi}_{w_\rho}$ and $Y = \mathcal{G}^{\varphi}_{w_\rho}$.

  For this proof we will drop the superscript $\mathcal{M}'$ from Pr. In order to show this claim we need to prove that we have $\mathsf{Pr}(c \models \mathbf{G}\psi) = \mathsf{Pr}(c \models \mathbf{FG}\psi) \in \{0, 1\}$ and $\mathsf{Pr}(c \models \mathbf{F}\chi) = \mathsf{Pr}(c \models \mathbf{GF}\chi) \in \{0, 1\}$ for all $c \in C$, for all $\psi \in \nu(\varphi)$, and for all $\chi \in \mu(\varphi)$.

  - Let $\psi \in \nu(\varphi)$. If $\mathsf{Pr}(c \models \mathbf{G}\psi) = 1$ for all $c \in C$, then we are done. Let now $c \in C$ be such that $\mathsf{Pr}(c \models \mathbf{G}\psi) < 1$. We show that for all $d \in C$, we have $\mathsf{Pr}(d \models \mathbf{G}\psi) = 0$, thus also proving $\mathsf{Pr}(d \models \mathbf{FG}\psi) = 0$. Since $\mathsf{Pr}(c \models \mathbf{G}\psi] < 1$, we have $\mathsf{Pr}(c \models \mathbf{F}\neg\psi) > 0$. Therefore, with every visit of c there is a positive probability p that ψ will be violated in the next n steps for some $n \in \mathbb{N}$. Since c is in a BSCC it will be visited infinitely often with probability 1 from any $d \in C$. Consequently, $\mathsf{Pr}(d \models \mathbf{G}\psi) \leq \lim_{k\to\infty}(1-p)^k = 0$.
  - Let $\psi \in \mu(\varphi)$. If $\mathsf{Pr}(c \models \mathbf{F}\psi) = 0$ for all $c \in C$, then we are done. Let now $c \in C$ be such that $\mathsf{Pr}(c \models \mathbf{F}\psi) > 0$. We show that for all $d \in C$, we have $\mathsf{Pr}(d \models \mathbf{F}\psi) = 1$, thus also proving $\mathsf{Pr}(d \models \mathbf{GF}\psi) = 1$. Since we have $\mathsf{Pr}(c \models \mathbf{F}\psi) > 0$, with every visit of c there is a positive probability p that ψ will be satisfied in the next n steps for some $n \in \mathbb{N}$. Since c is in a BSCC it will

be visited infinitely often with probability 1 from any $d \in C$. Consequently, $\mathsf{Pr}(d \models \mathbf{F}\psi) \geq 1 - \lim_{k\to\infty}(1-p)^k = 1$.

We are now ready to prove the inequality '$\leq$'. From Proposition 9.6 we know that a finite-memory strategy for the left-hand side $\sup_{\mathfrak{S}}$ suffices. Thus let $\mathfrak{S}$ be finite-memory strategy, such that the following holds:

$$\mathsf{Pr}^{\mathcal{M}_{\mathfrak{S}}}(s \models \varphi) = \sup_{\mathfrak{S}'} \mathsf{Pr}^{\mathcal{M}_{\mathfrak{S}'}}(s \models \varphi)$$

We derive from this a strategy $\mathfrak{S}'$ for $\mathcal{M} \otimes \mathcal{A}$ such that

$$\mathsf{Pr}^{\mathcal{M}_{\mathfrak{S}}}(s \models \varphi) \leq \mathsf{Pr}^{(\mathcal{M}\otimes\mathcal{A})_{\mathfrak{S}'}}(\Box\Diamond Acc)$$

The strategy $\mathfrak{S}'$ follows the behaviour of $\mathfrak{S}$ up to the point where a BSCC is reached in $\mathcal{M}_{\mathfrak{S}}$. A path in $\mathcal{M}_{\mathfrak{S}}$ will almost surely reach some BSCC in $\mathcal{M}_{\mathfrak{S}}$, since $\mathfrak{S}$ is a finite-memory strategy and thus $\mathcal{M}_{\mathfrak{S}}$ is a finite Markov chain. Let c be the first visited state in a BSCC of $\mathcal{M}_{\mathfrak{S}}$. By the claim, c is decided and thus there exists the same X and Y for a set of paths with probability 1, and $\mathfrak{S}'$ then chooses the unique action $\alpha_{X,Y}$ if $\mathfrak{S}$ would choose the action α. Having performed the switch to the accepting component, the strategy $\mathfrak{S}'$ then continues to follow the behaviour of $\mathfrak{S}$ indefinitely. Note that apart from emulating $\mathfrak{S}$, the constructed strategy $\mathfrak{S}'$ only decides when to switch to the accepting component and with which X and Y.

Notice that by construction every path ρ in $\mathcal{M}_{\mathfrak{S}}$ corresponds to a path ρ' in $(\mathcal{M}\otimes\mathcal{A})_{\mathfrak{S}'}$ with equal transition probabilities. It thus only remains to show that if a trace of ρ is accepted by φ then the corresponding path ρ', projected to the second component, is also an accepting run on $\mathcal{A}$. For this notice that when ρ reaches m at point i it has $index(\rho_i) = 0$ and thus the trace of ρ_i is ν- and μ-stable. Hence switching to the matching accepting component does not change acceptance of the trace of ρ. □

Implementation: MoChiBa. Using this theorem one can show that the LDBAs obtained through Theorem 6.2 can be used for quantitative probabilistic model checking. We implemented a prototype, called `MoChiBa` [SK16], experimentally supporting the claim that LDBAs are useful and can yield better performance. Notice that at that time only the LDBA construction of [SEJK16] existed and thus we tested in `MoChiBa` only the usefulness of that construction. It remains open how the LDBAs from Theorem 6.2 compare, but the results from Chapter 8 indicate they should be similar or even sometimes better.

9.2 Synthesis of Reactive Systems

Synthesis of reactive systems refers to the problem of finding for a formal specification of an input-output relation a matching implementation [PR89]. Concretely, let φ be a specification given as an LTL formula and let the atomic propositions $Ap = I \uplus O$ be partitioned into *inputs* I and *outputs* O. Then the *synthesis problem* is to decide if there exists and, if so, to compute a strategy $\mathfrak{S} : (2^I)^* \to 2^O$ such that for every sequence of inputs $w_I = \sigma_0\sigma_1 \cdots \in (2^I)^\omega$, there exits a sequence of outputs $w_O = \mathfrak{S}(\sigma_0)\mathfrak{S}(\sigma_0\sigma_1)\cdots \in (2^O)^\omega$ such that the point-wise union of letters from w_I and w_O satisfies φ:

$$\binom{\sigma_0 \cup}{\mathfrak{S}(\sigma_0)}\binom{\sigma_1 \cup}{\mathfrak{S}(\sigma_0\sigma_1)}\binom{\sigma_2 \cup}{\mathfrak{S}(\sigma_0\sigma_1\sigma_2)}\cdots\binom{\sigma_i \cup}{\mathfrak{S}(\sigma_0\ldots\sigma_i)}\cdots \models \varphi$$

If such a strategy exists, we call φ *realisable*. For realisable formulas we then extract an implementation of $\mathfrak{S}$, e.g. represented by a Mealy machine.

Algorithmically there are several approaches to reactive synthesis and most of them rely in some way on a translation of the specification to an automaton. The original solution proposed by [PR89] relied on a translation of LTL to deterministic Rabin automata that is followed by a reinterpretation of the DRA as a tree-automaton and a language emptiness check of that structure. While this procedure is known to be asymptotically optimal, tools able to deal with large specifications have been elusive. This is, besides other points, attributed to the usage of automata that suffer from the 'messy state space' [Kup12] of Safra's determinisation [Saf88] hindering efficient implementations. Thus alternative approaches have been devised [KPV06; FFT17; BBF+12; KJB13; Ehl11] to avoid the complicated state space and to alleviate the immanent state explosion problem for deterministic automata.

We, on the other hand, can address the 'messy state space' by the newly developed Master Theorem that gives rise to a collection of 'Safraless' LTL translations. In particular recent work [EKRS17] discovered that LDBAs from [SEJK16] and Theorem 6.2 can be efficiently translated to deterministic parity automata (DPA), a subclass of Rabin automata, for which the emptiness problem on tree-automata is efficiently solvable [GTW02]. Since the determinisation procedure of [EKRS17] has access to the semantic labelling – the LTL formulas the states are labeled with – of LDBAs ([SEJK16], Theorem 6.2), efficient approximative language inclusion checks can be used to remove redundant states. The procedure without the pruning might be seen as a special case of the Muller-Schupp determinisation procedure, but focusing on a particular structure allows to have simplified construction and adding specialised optimisations is easier.

Implementation: Strix. The DPA translation and the translations for fragments of LTL to DCAs and DBAs from Chapter 5 were successfully put to test in the reactive synthesis tool `Strix` [MSL18; LMS19]. `Strix` decomposes the specification using heuristics and translates each part of it using `Owl` [KMS18] to a deterministic automaton. Then it recombines these automata relying by means of union and intersection and solves the non-emptiness problem on the resulting on-the-fly constructed deterministic parity tree automaton – choosing the parity game interpretation – in a forward-search manner with an improved implementation of [ML16]. The decomposition here is a crucial part of the approach, since it reduces work by detecting isomorphic subformulas and by picking more efficient translations for fragments as seen in Chapter 5 even-though the whole formula might not belong to such a fragment. For an extended description of the approach the reader is referred to [MSL18; LMS19].

9.A Markov Chains and Markov Decision Processes

In this section we collect well-known standard terminology and results on Markov chains (MC) and Markov decision processes (MDP) from [BK08] for reference with slight modifications to enhance readability for our specific use-case. For an extended discussion of these formalisms we refer the interested reader to [BK08].

Definition 9.3 ([BK08])**.** *A Markov chain is a tuple* $\mathcal{M} \coloneqq (S, \mathbf{P}, \iota_{init}, Ap, L)$ *where*

- S *is a nonempty set of states,*
- $\mathbf{P}\colon S \times S \to [0,1]$ *is a* transition probability function *such that for all states* $s \in S$*:*

$$\sum_{s' \in S} \mathbf{P}(s, s') \in \{0, 1\},$$

- $\iota_{init}\colon S \to [0,1]$ *is an* initial distribution *with* $\sum_{s \in S} \iota_{init}(s) = 1$,
- Ap *is a set of atomic propositions, and* $L : S \to 2^{Ap}$ *is a labelling function.*

We call a Markov chain finite if S *and* Ap *are finite. Further, we denote the probability of eventually reaching some set* $B \subseteq S$ *of states from some state* $s \in S$ *by* $\mathsf{Pr}^{\mathcal{M}}(s \models \Diamond B)$*, the probability of repeatedly reaching* B *by* $\mathsf{Pr}^{\mathcal{M}}(s \models \Box\Diamond B)$*, and the probability of eventually staying within* B *by* $\mathsf{Pr}^{\mathcal{M}}(s \models \Diamond\Box B)$*. The probability of satisfying an LTL formula* φ *over the set of atomic propositions* Ap *from some state* $s \in S$ *is written* $\mathsf{Pr}^{\mathcal{M}}(s \models \varphi)$*. Lastly, we write* $\mathsf{Pr}^{\mathcal{M}}(X)$ *as an abbreviation for* $\sum_{s \in S} \iota_{init}(s) \cdot \mathsf{Pr}^{\mathcal{M}}(s \models X)$ *where* X *is some term.*

Definition 9.4 ([BK08])**.** *A finite Markov decision process is a tuple*

$$\mathcal{M} \coloneqq (S, Act, \mathbf{P}, \iota_{init}, Ap, L)$$

where

- S *is a set of states,*
- Act *is a set of actions,*
- $\mathbf{P}\colon S \times Act \times S \to [0,1]$ *is a* transition probability function *such that for all states* $s \in S$ *and actions* $\alpha \in Act$*:*

$$\sum_{s' \in S} \mathbf{P}(s, \alpha, s') \in \{0, 1\},$$

- $\iota_{init}\colon S \to [0,1]$ *is an* initial distribution *with* $\sum_{s \in S} \iota_{init}(s) = 1$,
- Ap *is a set of atomic propositions, and* $L : S \to 2^{Ap}$ *is a labelling function.*

We call an Markov decision process finite if S*,* Act*, and* Ap *are finite. An action* α *is* enabled *in a state* s *if and only if* $\sum_{s' \in S} \mathbf{P}(s, \alpha, s') = 1$*. We denote the set of enabled actions in* s *by* $Act(s)$*. For any state* $s \in S$*, it is require that* $Act(s) \neq \emptyset$*.*

Observe that any Markov chain $\mathcal{M}$ can be seen as an MDP by associating each state s with a singleton set $Act(s)$. The other way round, an MDP can be seen as a Markov chain, if $Act(s)$ is a singleton set for all states s. Further, we can resolve the nondeterminism introduced through the actions by a strategy that resolves the nondeterministic choice:

Proposition 9.5 ([BK08]). *Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, Ap, L)$ be a finite MDP. A* strategy *for $\mathcal{M}$ is a function $\mathfrak{S} : S^+ \to Act$ such that $\mathfrak{S}(s_0 s_1 \dots s_n) \in Act(s_n)$ for all $s_0 s_1 \dots s_n \in S^+$. The strategy $\mathfrak{S}$ induces a Markov chain $\mathcal{M}_\mathfrak{S} \coloneqq (S^+, \mathbf{P}_\mathfrak{S}, \iota_{init}, Ap, L')$ where for $\rho = s_0 s_1 \dots s_n$ we define:*

$$\mathbf{P}_\mathfrak{S}(\rho, \rho s_{n+1}) \coloneqq \mathbf{P}(s_n, \mathfrak{S}(\rho), s_{n+1}) \quad \text{and} \quad L'(\rho) \coloneqq L(s_n)$$

A finite-memory strategy *$\mathfrak{S}$ for $\mathcal{M}$ is a tuple $\mathfrak{S} \coloneqq (Q, act, \Delta, start)$ where*

- *Q is a finite set of states,*
- *$\Delta \colon Q \times S \to Q$ is a transition function,*
- *$act \colon Q \times S \to Act$ is a function that selects an action $act(q, s) \in Act(s)$ for any state $q \in Q$ and state $s \in S$,*
- *and $start \colon S \to Q$ is a function that selects a starting state for state $s \in S$.*

For a finite-memory strategy $\mathfrak{S}$ the induced Markov chain $\mathcal{M}_\mathfrak{S}$ is finite.

Proposition 9.6 ([BK08]). *Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, Ap, L)$ be a finite MDP, and let $B \subseteq S$ be a set of states. There exists a finite-memory strategy $\mathfrak{S}$ such that for any $s \in S$:*

$$\mathsf{Pr}^\mathfrak{S}(s \models \varphi) = \sup_{\mathfrak{S}'} \mathsf{Pr}^{\mathfrak{S}'}(s \models \varphi)$$

Definition 9.7 ([BK08]). *Let $\mathcal{M} = (S, Act, \mathbf{P}, \iota_{init}, Ap, L)$ be a finite MDP, and let $\mathcal{D} = (Q, \delta, q_0, \beta)$ be a DRA over the alphabet 2^{Ap}. The* product MDP *is defined as:*

$$\mathcal{M} \otimes \mathcal{D} \coloneqq \left(S \times Q, Act, \mathbf{P}', \iota'_{init}, Q, L'\right)$$

where we define for all states $\langle s, q\rangle, \langle s', q'\rangle \in S \times Q$, and actions $\alpha \in Act$:

- $\mathbf{P}'\left(\langle s, q\rangle, \alpha, \langle s', q'\rangle\right) \coloneqq \begin{cases} \mathbf{P}(s, \alpha, s') & \text{if } q' = \delta(q, L(s')) \\ 0 & \text{otherwise} \end{cases}$
- $\iota'_{init}\left(\langle s, q\rangle\right) \coloneqq \begin{cases} \iota_{init}(s) & \text{if } q = \delta(q_0, L(s)) \\ 0 & \text{otherwise} \end{cases}$
- $L'\left(\langle s, q\rangle\right) \coloneqq \{q\}$

10 Concluding Remarks

The Master Theorem we presented provides a decomposition of LTL formulas from which one can derive LTL translations in a straight-forward way. This result builds upon work from a series of publications [KE12; KL13; EK14; EKS16; SEJK16]. In particular, the idea that a word eventually stabilises with respect to a formula and the idea of inductively checking complex LTL expression by delegating to auxiliary automata are already outlined there. Other translations such as the translation to LDBAs of [KV15; KV17] or the obligation sets of [LPZ+13; LZZ+18] follow similar ideas.

The Master Theorem is made possible by the addition of the operators $\mathbf{W}$ and $\mathbf{M}$ to the core LTL syntax which makes it *complete* in the sense that for every modal operator there exits a weak and strong variant. The essential novelty is that the mappings $\cdot[\cdot]_\mu$ and $\cdot[\cdot]_\nu$ exploit the existence of these variants and that applying these mappings to an arbitrary formula φ yield a simpler formula, but *not in the sense one might expect.* In particular, $\varphi[Y]_\mu$ might be *stronger* than φ. For example, the information that, say, the formula $a\mathbf{W}b$ does not hold infinitely often reduces to a check of the *stronger* formula $a\mathbf{U}b = (a\mathbf{W}b)[\emptyset]_\mu$. However, this lends the Master Theorem its practical benefit: The formulas $\varphi[X]_\nu$ and $\varphi[Y]_\mu$ are *simpler to translate.* This completeness of the syntax is the basis of the symmetric treatment of modal operators, where we deal with greatest- and least-fixed-point operators in a dual way. Such a symmetric treatment of greatest- and least-fixed-point operators is present in [KE12; KL13], but could only be applied to $\mathbf{F}$ and $\mathbf{G}$ operators. In some sense the result presented in this dissertation successfully finishes the journey started in [KE12]: a single theorem provides an arguably elegant (unified, symmetric, syntax-independent, not overly complex) and efficient (asymptotically optimal, practically relevant, direct) translation of LTL into an ω-automaton of your choice.

Open Questions and Future Work. We focussed in this dissertation on three classes of automata, namely NBAs, LDBAs, and DRAs, and did not investigate other classes. Consequently, the obvious follow-up questions are how constructions for other classes would be defined. It is open if we can obtain a (direct) translation to deterministic parity automata using techniques of [EKRS17] or [Löd99] that is better than chaining the involved construction. Further, a translation to non-deterministic Emerson-Lei (NELA) or Rabin automata (NRA) could be of interest, since a translation to NRAs or NELAs is able to save states compared to NBA constructions. For example the formula $\mathbf{FG}a$ can be translated to a single-state NRA with transition-acceptance, but the smallest NBA with transition-acceptance for that formula has two states.

Furthermore, one could look deeper into the finer details of the congruences used. It seems to be interesting to search for another suitable and practical equivalence relation between $\sim_q$ and $\sim_l$ and for an equivalence relation between $\sim_c$ and $\sim_p$ that yields a finite state space and is significantly simpler to compute than $\sim_p$. Moreover, we think a further investigation of the results from Chapter 8 could lead to more restrictive version

of the Master Theorem, in the form of revised Proposition 4.18. Further, The presented constructions have a highly regular structure compared to the DRAs obtained through determinisation constructions, such as Safra's or similar constructions, and thus we believe that the translations to LDBAs and DRAs are a perfect fit for a representation as symbolic automata and a symbolic translation.

Finally, we think other applications of the decomposition provided by the Master Theorem could be:

- Bounded-Alternation Normal-Form for Linear Temporal Logic

  Let us consider the following normal-form where each formula is a Boolean combination of modal operators with at most one alternation of the fixed-point operator types whit-in their scope, i.e. every path starting a the root of the syntax tree can be split into two segments such that in one only greatest-fixed-point operators are present and in the other only least-fixed-point operators are present. We believe that such a normal-form can be obtained through the Master Theorem and indeed (2) and (3) already have the necessary shape and only (1) needs a bit of extra work. Further, we think that such a normal-form can be obtained by mere syntactic rewriting.

- Intuitive Explanations for Linear Temporal Logic on Lasso Words

  The challenges applying formal methods in an industrial context are often underestimated. In particular, explaining findings to the user poses an immense challenge. Consider for example that it might not be obvious why a counter-example (represented as a lasso word $w = uv^\omega$) violates the given specification. The approach of [BBT18] proposes the use of proof-trees for a step-by-step analysis of the problem by the user. We think that extending this approach by clearly separating the origin of the problem could help the user: Does the violation of the property occur in the finite part u? Are my assumptions about the recurring infinite behaviour correct? Is the problem in the infinite loop? Notice that the Master Theorem provides answers to this, since we can simplify if the shape of w is uv^ω. The index i from the Master Theorem can be set to a fixed value depending on the length of uv and X, Y can be computed recursively on the lasso v. We believe that in combination with [BBT18] we can provide intuitive and helpful explanations.

While this dissertation spent a considerable amount of space to identify optimisations to the proposed constructions, it should not to be forgotten that the Master Theorem is a blue-print for decomposing LTL formulas, giving a common and unified basis for translations to automata. Consequently, one can easily customise existing translations or even devise new translations for a specific use from it. Moreover, we see opportunities for using the Master Theorem besides the translation of LTL to automata and are looking forward to explore these ideas.

Bibliography

[AL04] Rajeev Alur and Salvatore La Torre. 'Deterministic generators and games for LTL fragments'. In: *ACM Trans. Comput. Log.* 5.1 (2004), pp. 1–25. DOI: 10.1145/963927.963928.

[Ant96] Valentin M. Antimirov. 'Partial Derivatives of Regular Expressions and Finite Automaton Constructions'. In: *Theor. Comput. Sci.* 155.2 (1996), pp. 291–319. DOI: 10.1016/0304-3975(95)00182-4.

[BBD+13] Tomáš Babiak, Thomas Badie, Alexandre Duret-Lutz, Mojmír Křetínský and Jan Strejček. 'Compositional Approach to Suspension and Other Improvements to LTL Translation'. In: *SPIN*. 2013, pp. 81–98. DOI: 10.1007/978-3-642-39176-7_6.

[BBD+15] Tomáš Babiak, František Blahoudek, Alexandre Duret-Lutz, Joachim Klein, Jan Křetínský, David Müller, David Parker and Jan Strejček. 'The Hanoi Omega-Automata Format'. In: *CAV*. 2015, pp. 479–486. DOI: 10.1007/978-3-319-21690-4_31.

[BBF+12] Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, Naiyong Jin and Jean-François Raskin. 'Acacia+, a Tool for LTL Synthesis'. In: *CAV*. 2012, pp. 652–657. DOI: 10.1007/978-3-642-31424-7_45.

[BBKS13] Tomáš Babiak, Frantisek Blahoudek, Mojmír Křetínský and Jan Strejcek. 'Effective Translation of LTL to Deterministic Rabin Automata: Beyond the (F, G)-Fragment'. In: *ATVA*. 2013, pp. 24–39. DOI: 10.1007/978-3-319-02444-8_4.

[BBT18] David A. Basin, Bhargav Nagaraja Bhatt and Dmitriy Traytel. 'Optimal Proofs for Linear Temporal Logic on Lasso Words'. In: *ATVA*. 2018, pp. 37–55. DOI: 10.1007/978-3-030-01090-4_3.

[BDK+17] Frantisek Blahoudek, Alexandre Duret-Lutz, Mikulás Klokocka, Mojmír Křetínský and Jan Strejcek. 'Seminator: A Tool for Semi-Determinization of Omega-Automata'. In: *LPAR*. Vol. 46. EPiC Series in Computing. EasyChair, 2017, pp. 356–367.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[BKK+16] Christel Baier, Stefan Kiefer, Joachim Klein, Sascha Klüppelholz, David Müller and James Worrell. 'Markov Chains and Unambiguous Büchi Automata'. In: *CAV*. 2016, pp. 23–42. DOI: 10.1007/978-3-319-41528-4_2.

[BKRS12] Tomáš Babiak, Mojmír Křetínský, Vojtech Rehák and Jan Strejcek. 'LTL to Büchi Automata Translation: Fast and More Deterministic'. In: *TACAS*. 2012, pp. 95–109. DOI: 10.1007/978-3-642-28756-5_8.

[Brz64] Janusz A. Brzozowski. 'Derivatives of Regular Expressions'. In: *J. ACM* 11.4 (1964), pp. 481–494. DOI: 10.1145/321239.321249.

[Büc60] J.R. Büchi. 'Weak second-order arithmetic and finite automata.' In: *Z. Math. Logik Grundlagen Math. 6* (1960).

[Büc66] J. Richard Büchi. 'Symposium on Decision Problems: On a Decision Method in Restricted Second Order Arithmetic'. In: *Logic, Methodology and Philosophy of Science*. Vol. 44. Studies in Logic and the Foundations of Mathematics. Elsevier, 1966, pp. 1–11. DOI: 10.1016/S0049-237X(09)70564-6.

[CE81] Edmund M. Clarke and E. Allen Emerson. 'Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic'. In: *Logics of Programs*. 1981, pp. 52–71. DOI: 10.1007/BFb0025774.

[CGK13] Krishnendu Chatterjee, Andreas Gaiser and Jan Křetínský. 'Automata with Generalized Rabin Pairs for Probabilistic Model Checking and LTL Synthesis'. In: *CAV*. 2013, pp. 559–575. DOI: 10.1007/978-3-642-39799-8_37.

[Chu57] Alonzo Church. 'Applications of recursive arithmetic to the problem of circuit synthesis'. In: *Summaries of the Summer Institute of Symbolic Logic* (1957).

[CHVB18] Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith and Roderick Bloem, eds. *Handbook of Model Checking*. Springer, 2018. DOI: 10.1007/978-3-319-10575-8.

[Cou99] Jean-Michel Couvreur. 'On-the-Fly Verification of Linear Temporal Logic'. In: *FM*. 1999, pp. 253–271. DOI: 10.1007/3-540-48119-2_16.

[CY95] Costas Courcoubetis and Mihalis Yannakakis. 'The Complexity of Probabilistic Verification'. In: *J. ACM* 42.4 (1995), pp. 857–907. DOI: 10.1145/210332.210339.

[DAC98] Matthew B. Dwyer, George S. Avrunin and James C. Corbett. 'Property specification patterns for finite-state verification'. In: *FMSP*. 1998, pp. 7–15. DOI: 10.1145/298595.298598.

[DGV99] Marco Daniele, Fausto Giunchiglia and Moshe Y. Vardi. 'Improved Automata Generation for Linear Temporal Logic'. In: *CAV*. 1999, pp. 249–260. DOI: 10.1007/3-540-48683-6_23.

[DLF+16] Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Etienne Renault and Laurent Xu. 'Spot 2.0 - A Framework for LTL and ω-Automata Manipulation'. In: *ATVA*. 2016, pp. 122–129. DOI: 10.1007/978-3-319-46520-3_8.

[EC82] E. Allen Emerson and Edmund M. Clarke. 'Using Branching Time Temporal Logic to Synthesize Synchronization Skeletons'. In: *Sci. Comput. Program.* 2.3 (1982), pp. 241–266. DOI: 10.1016/0167-6423(83)90017-5.

[EH00] Kousha Etessami and Gerard J. Holzmann. 'Optimizing Büchi Automata'. In: *CONCUR*. 2000, pp. 153–167. DOI: 10.1007/3-540-44618-4_13.

[Ehl11] Rüdiger Ehlers. 'Unbeast: Symbolic Bounded Synthesis'. In: *TACAS*. 2011, pp. 272–275. DOI: 10.1007/978-3-642-19835-9_25.

[EK14] Javier Esparza and Jan Křetínský. 'From LTL to Deterministic Automata: A Safraless Compositional Approach'. In: *CAV*. 2014, pp. 192–208. DOI: 10.1007/978-3-319-08867-9_13.

[EKRS17] Javier Esparza, Jan Křetínský, Jean-François Raskin and Salomon Sickert. 'From LTL and Limit-Deterministic Büchi Automata to Deterministic Parity Automata'. In: *TACAS*. 2017, pp. 426–442. DOI: 10.1007/978-3-662-54577-5_25.

[EKS16] Javier Esparza, Jan Křetínský and Salomon Sickert. 'From LTL to deterministic automata - A safraless compositional approach'. In: *Formal Methods in System Design* 49.3 (2016), pp. 219–271. DOI: 10.1007/s10703-016-0259-2.

[EKS18] Javier Esparza, Jan Křetínský and Salomon Sickert. 'One Theorem to Rule Them All: A Unified Translation of LTL into ω-Automata'. In: *LICS*. 2018, pp. 384–393. DOI: 10.1145/3209108.3209161.

[Eme90] E. Allen Emerson. 'Temporal and Modal Logic'. In: *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*. Ed. by Jan van Leeuwen. Elsevier and MIT Press, 1990, pp. 995–1072. DOI: 10.1016/b978-0-444-88074-1.50021-4.

[FFT17] Peter Faymonville, Bernd Finkbeiner and Leander Tentrup. 'BoSy: An Experimentation Framework for Bounded Synthesis'. In: *CAV (II)*. 2017, pp. 325–332. DOI: 10.1007/978-3-319-63390-9_17.

[FKVW15] Seth Fogarty, Orna Kupferman, Moshe Y. Vardi and Thomas Wilke. 'Profile trees for Büchi word automata, with application to determinization'. In: *Inf. Comput.* 245 (2015), pp. 136–151. DOI: 10.1016/j.ic.2014.12.021.

[Flo67] Robert W. Floyd. 'Assigning meanings to programs'. In: *Proc. Sympos. Appl. Math., Vol. XIX*. Amer. Math. Soc., Providence, R.I., 1967, pp. 19–32.

[Fri03] Carsten Fritz. 'Constructing Büchi Automata from Linear Temporal Logic Using Simulation Relations for Alternating Büchi Automata'. In: *CIAA*. 2003, pp. 35–48. DOI: 10.1007/3-540-45089-0_5.

[GH06] Jaco Geldenhuys and Henri Hansen. 'Larger Automata and Less Work for LTL Model Checking'. In: *SPIN*. 2006, pp. 53–70. DOI: 10.1007/11691617_4.

[GL02] Dimitra Giannakopoulou and Flavio Lerda. 'From States to Transitions: Improving Translation of LTL Formulae to Büchi Automata'. In: *FORTE*. 2002, pp. 308–326. DOI: 10.1007/3-540-36135-9_20.

[GO01] Paul Gastin and Denis Oddoux. 'Fast LTL to Büchi Automata Translation'. In: *CAV*. 2001, pp. 53–65. DOI: 10.1007/3-540-44585-4_6.

[GPSS80] Dov Gabbay, Amir Pnueli, Saharon Shelah and Jonathan Stavi. 'On the Temporal Analysis of Fairness'. In: *POPL*. 1980, pp. 163–173. DOI: 10.1145/567446.567462.

[GPVW95] Rob Gerth, Doron A. Peled, Moshe Y. Vardi and Pierre Wolper. 'Simple on-the-fly automatic verification of linear temporal logic'. In: *Protocol Specification, Testing and Verification XV, Proceedings of the Fifteenth IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification*. 1995, pp. 3–18.

[GTW02] Erich Grädel, Wolfgang Thomas and Thomas Wilke, eds. *Automata, Logics, and Infinite Games: A Guide to Current Research*. 2002. DOI: 10.1007/3-540-36387-4.

[HLS+15] Ernst Moritz Hahn, Guangyuan Li, Sven Schewe, Andrea Turrini and Lijun Zhang. 'Lazy Probabilistic Model Checking without Determinisation'. In: *CONCUR*. 2015, pp. 354–367. DOI: 10.4230/LIPIcs.CONCUR.2015.354.

[Hoa69] C. A. R. Hoare. 'An Axiomatic Basis for Computer Programming'. In: *Commun. ACM* 12.10 (1969), pp. 576–580. DOI: 10.1145/363235.363259.

[HP06] Thomas A. Henzinger and Nir Piterman. 'Solving Games Without Determinization'. In: *CSL*. 2006, pp. 395–410. DOI: 10.1007/11874683_26.

[KB06] Joachim Klein and Christel Baier. 'Experiments with deterministic ω-automata for formulas of linear temporal logic'. In: *Theor. Comput. Sci.* 363.2 (2006), pp. 182–195. DOI: 10.1016/j.tcs.2006.07.022.

[KE12] Jan Křetínský and Javier Esparza. 'Deterministic Automata for the (F,G)-Fragment of LTL'. In: *CAV*. 2012, pp. 7–22. DOI: 10.1007/978-3-642-31424-7_7.

[KJB13] Ayrat Khalimov, Swen Jacobs and Roderick Bloem. 'PARTY Parameterized Synthesis of Token Rings'. In: *CAV*. 2013, pp. 928–933. DOI: 10.1007/978-3-642-39799-8_66.

[KK14] Zuzana Komárková and Jan Křetínský. 'Rabinizer 3: Safraless Translation of LTL to Small Deterministic Automata'. In: *ATVA*. 2014, pp. 235–241. DOI: 10.1007/978-3-319-11936-6_17.

[KL13] Jan Křetínský and Ruslán Ledesma-Garza. 'Rabinizer 2: Small Deterministic Automata for LTL \ GU'. In: *ATVA*. 2013, pp. 446–450. DOI: 10.1007/978-3-319-02444-8_32.

[Kle56] S. C. Kleene. 'Representation of Events in Nerve Nets and Finite Automata'. In: *Automata Studies, Annals of Math. Studies 34*. Ed. by C. Shannon and J. McCarthy. New Jersey, 1956.

[KMBK14] Joachim Klein, David Müller, Christel Baier and Sascha Klüppelholz. 'Are Good-for-Games Automata Good for Probabilistic Model Checking?' In: *LATA*. 2014, pp. 453–465. DOI: 10.1007/978-3-319-04921-2_37.

[KMS18] Jan Křetínský, Tobias Meggendorfer and Salomon Sickert. 'Owl: A Library for ω-Words, Automata, and LTL'. In: *ATVA*. 2018, pp. 543–550. DOI: 10.1007/978-3-030-01090-4_34.

[KMSZ18] Jan Křetínský, Tobias Meggendorfer, Salomon Sickert and Christopher Ziegler. 'Rabinizer 4: From LTL to Your Favourite Deterministic Automaton'. In: *CAV (I)*. 2018, pp. 567–577. DOI: 10.1007/978-3-319-96145-3_30.

[KMWW17] Jan Křetínský, Tobias Meggendorfer, Clara Waldmann and Maximilian Weininger. 'Index Appearance Record for Transforming Rabin Automata into Parity Automata'. In: *TACAS*. 2017, pp. 443–460. DOI: 10.1007/978-3-662-54577-5_26.

[KNP11] Marta Z. Kwiatkowska, Gethin Norman and David Parker. 'PRISM 4.0: Verification of Probabilistic Real-Time Systems'. In: *CAV*. 2011, pp. 585–591. DOI: 10.1007/978-3-642-22110-1_47.

[KPR98] Yonit Kesten, Amir Pnueli and Li-on Raviv. 'Algorithmic Verification of Linear Temporal Logic Specifications'. In: *ICALP*. 1998, pp. 1–16. DOI: 10.1007/BFb0055036.

[KPV06] Orna Kupferman, Nir Piterman and Moshe Y. Vardi. 'Safraless Compositional Synthesis'. In: *CAV*. 2006, pp. 31–44. DOI: 10.1007/11817963_6.

[Kup12] Orna Kupferman. 'Recent Challenges and Ideas in Temporal Synthesis'. In: *SOFSEM*. 2012, pp. 88–98. DOI: 10.1007/978-3-642-27660-6_8.

[KV01] Orna Kupferman and Moshe Y. Vardi. 'Model Checking of Safety Properties'. In: *Formal Methods in System Design* 19.3 (2001), pp. 291–314. DOI: 10.1023/A:1011254632723.

[KV15] Dileep Kini and Mahesh Viswanathan. 'Limit Deterministic and Probabilistic Automata for LTL \ GU'. In: *TACAS*. 2015, pp. 628–642. DOI: 10.1007/978-3-662-46681-0_57.

[KV17] Dileep Kini and Mahesh Viswanathan. 'Optimal Translation of LTL to Limit Deterministic Automata'. In: *TACAS*. 2017, pp. 113–129. DOI: 10.1007/978-3-662-54580-5_7.

[KV98] Orna Kupferman and Moshe Y. Vardi. 'Freedom, Weakness, and Determinism: From Linear-Time to Branching-Time'. In: *LICS*. 1998, pp. 81–92. DOI: 10.1109/LICS.1998.705645.

[KW08] Detlef Kähler and Thomas Wilke. 'Complementation, Disambiguation, and Determinization of Büchi Automata Unified'. In: *ICALP (I)*. 2008, pp. 724–735. DOI: 10.1007/978-3-540-70575-8_59.

[Lee90] Jan van Leeuwen, ed. *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*. Elsevier and MIT Press, 1990.

[LMS19] Michael Luttenberger, Philipp J. Meyer and Salomon Sickert. 'Practical Synthesis of Reactive Systems from LTL Specifications via Parity Games'. In: *CoRR* abs/1903.12576 (2019). arXiv: 1903.12576.

[Löd99] Christof Löding. 'Optimal Bounds for Transformations of ω-Automata'. In: *FSTTCS*. 1999, pp. 97–109. DOI: 10.1007/3-540-46691-6_8.

[LP19] Christof Löding and Anton Pirogov. 'Determinization of Büchi Automata: Unifying the Approaches of Safra and Muller-Schupp'. In: *ICALP*. 2019, 120:1–120:13. DOI: 10.4230/LIPIcs.ICALP.2019.120.

[LPZ+13] Jianwen Li, Geguang Pu, Lijun Zhang, Zheng Wang, Jifeng He and Kim Guldstrand Larsen. 'On the Relationship between LTL Normal Forms and Büchi Automata'. In: *Theories of Programming and Formal Methods - Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday*. 2013, pp. 256–270. DOI: 10.1007/978-3-642-39698-4_16.

[LZZ+18] Jianwen Li, Lijun Zhang, Shufang Zhu, Geguang Pu, Moshe Y. Vardi and Jifeng He. 'An explicit transition system construction approach to LTL satisfiability checking'. In: *Formal Asp. Comput.* 30.2 (2018), pp. 193–217. DOI: 10.1007/s00165-017-0442-2.

[McN66] Robert McNaughton. 'Testing and Generating Infinite Sequences by a Finite Automaton'. In: *Information and Control* 9.5 (1966), pp. 521–530. DOI: 10.1016/S0019-9958(66)80013-X.

[MH84] Satoru Miyano and Takeshi Hayashi. 'Alternating Finite Automata on omega-Words'. In: *Theor. Comput. Sci.* 32 (1984), pp. 321–330. DOI: 10.1016/0304-3975(84)90049-5.

[ML16] Philipp J. Meyer and Michael Luttenberger. 'Solving Mean-Payoff Games on the GPU'. In: *ATVA*. 2016, pp. 262–267. DOI: 10.1007/978-3-319-46520-3_17.

[MP90] Zohar Manna and Amir Pnueli. 'A Hierarchy of Temporal Properties'. In: *PODC*. 1990, pp. 377–410. DOI: 10.1145/93385.93442.

[MS08] Andreas Morgenstern and Klaus Schneider. 'From LTL to Symbolically Represented Deterministic Automata'. In: *VMCAI*. 2008, pp. 279–293. DOI: 10.1007/978-3-540-78163-9_24.

[MS17] David Müller and Salomon Sickert. 'LTL to Deterministic Emerson-Lei Automata'. In: *GandALF*. 2017, pp. 180–194. DOI: 10.4204/EPTCS.256.13.

[MS95] David E. Muller and Paul E. Schupp. 'Simulating Alternating Tree Automata by Nondeterministic Automata: New Results and New Proofs of the Theorems of Rabin, McNaughton and Safra'. In: *Theor. Comput. Sci.* 141.1&2 (1995), pp. 69–107. DOI: 10.1016/0304-3975(94)00214-4.

[MSL18] Philipp J. Meyer, Salomon Sickert and Michael Luttenberger. 'Strix: Explicit Reactive Synthesis Strikes Back!' In: *CAV (I)*. 2018, pp. 578–586. DOI: 10.1007/978-3-319-96145-3_31.

[MSS88] David E. Muller, Ahmed Saoudi and Paul E. Schupp. 'Weak Alternating Automata Give a Simple Explanation of Why Most Temporal and Dynamic Logics are Decidable in Exponential Time'. In: *LICS*. 1988, pp. 422–427. DOI: 10.1109/LICS.1988.5139.

[Par81] David Michael Ritchie Park. 'Concurrency and Automata on Infinite Sequences'. In: *Theoretical Computer Science, 5th GI-Conference*. 1981, pp. 167–183. DOI: 10.1007/BFb0017309.

[Pel07] Radek Pelánek. 'BEEM: Benchmarks for Explicit Model Checkers'. In: *SPIN*. 2007, pp. 263–267. DOI: 10.1007/978-3-540-73370-6_17.

[Pit07] Nir Piterman. 'From Nondeterministic Büchi and Streett Automata to Deterministic Parity Automata'. In: *Logical Methods in Computer Science* 3.3 (2007). DOI: 10.2168/LMCS-3(3:5)2007.

[Pnu77] Amir Pnueli. 'The Temporal Logic of Programs'. In: *FOCS*. 1977, pp. 46–57. DOI: 10.1109/SFCS.1977.32.

[Pnu81] Amir Pnueli. 'The Temporal Semantics of Concurrent Programs'. In: *Theor. Comput. Sci.* 13 (1981), pp. 45–60. DOI: 10.1016/0304-3975(81)90110-9.

[PR89] Amir Pnueli and Roni Rosner. 'On the Synthesis of a Reactive Module'. In: *POPL*. 1989, pp. 179–190. DOI: 10.1145/75277.75293.

[PZ08] Amir Pnueli and Aleksandr Zaks. 'On the Merits of Temporal Testers'. In: *25 Years of Model Checking - History, Achievements, Perspectives*. 2008, pp. 172–195. DOI: 10.1007/978-3-540-69850-0_11.

[Red12] Roman R. Redziejowski. 'An Improved Construction of Deterministic ω-automaton Using Derivatives'. In: *Fundam. Inform.* 119.3-4 (2012), pp. 393–406. DOI: 10.3233/FI-2012-744.

[Red99] Roman R. Redziejowski. 'Construction of a deterministic ω-automaton using derivatives'. In: *ITA* 33.2 (1999), pp. 133–158. DOI: 10.1051/ita:1999111.

[Saf88] Shmuel Safra. 'On the Complexity of omega-Automata'. In: *FOCS*. 1988, pp. 319–327. DOI: 10.1109/SFCS.1988.21948.

[SB00] Fabio Somenzi and Roderick Bloem. 'Efficient Büchi Automata from LTL Formulae'. In: *CAV*. 2000, pp. 248–263. DOI: 10.1007/10722167_21.

[Sch01] Klaus Schneider. 'Improving Automata Generation for Linear Temporal Logic by Considering the Automaton Hierarchy'. In: *LPAR*. 2001, pp. 39–54. DOI: 10.1007/3-540-45653-8_3.

[Sch09] Sven Schewe. 'Tighter Bounds for the Determinisation of Büchi Automata'. In: *FoSSaCS*. 2009, pp. 167–181. DOI: 10.1007/978-3-642-00596-1_13.

[SEJK16] Salomon Sickert, Javier Esparza, Stefan Jaax and Jan Křetínský. 'Limit-Deterministic Büchi Automata for Linear Temporal Logic'. In: *CAV*. 2016, pp. 312–332. DOI: 10.1007/978-3-319-41540-6_17.

[Sic16] Salomon Sickert. 'Linear Temporal Logic'. In: *Archive of Formal Proofs* (2016).

[Sis94] A. Prasad Sistla. 'Safety, Liveness and Fairness in Temporal Logic'. In: *Formal Asp. Comput.* 6.5 (1994), pp. 495–512. DOI: 10.1007/BF01211865.

[SK16] Salomon Sickert and Jan Křetínský. 'MoChiBA: Probabilistic LTL Model Checking Using Limit-Deterministic Büchi Automata'. In: *ATVA*. 2016, pp. 130–137. DOI: 10.1007/978-3-319-46520-3_9.

[ST18] Martin Sulzmann and Peter Thiemann. 'LTL Semantic Tableaux and Alternating ω-automata via Linear Factors'. In: *ICTAC*. 2018, pp. 11–34. DOI: 10.1007/978-3-030-02508-3_2.

[TRV12] Deian Tabakov, Kristin Y. Rozier and Moshe Y. Vardi. 'Optimized temporal monitors for SystemC'. In: *Formal Methods in System Design* 41.3 (2012), pp. 236–268. DOI: 10.1007/s10703-011-0139-8.

[TS15] Peter Thiemann and Martin Sulzmann. 'From ω-Regular Expressions to Büchi Automata via Partial Derivatives'. In: *LATA*. 2015, pp. 287–298. DOI: 10.1007/978-3-319-15579-1_22.

[Tur37] A. M. Turing. 'On Computable Numbers, with an Application to the Entscheidungsproblem'. In: *Proceedings of the London Mathematical Society* s2-42.1 (Jan. 1937), pp. 230–265. DOI: 10.1112/plms/s2-42.1.230.

[Var85] Moshe Y. Vardi. 'Automatic Verification of Probabilistic Concurrent Finite-State Programs'. In: *FOCS*. 1985, pp. 327–338. DOI: 10.1109/SFCS.1985.12.

[Var94] Moshe Y. Vardi. 'Nontraditional Applications of Automata Theory'. In: *TACS*. 1994, pp. 575–597. DOI: 10.1007/3-540-57887-0_116.

[VW86a] Moshe Y. Vardi and Pierre Wolper. 'An Automata-Theoretic Approach to Automatic Program Verification (Preliminary Report)'. In: *LICS*. 1986, pp. 332–344.

[VW86b] Moshe Y. Vardi and Pierre Wolper. 'Automata-Theoretic Techniques for Modal Logics of Programs'. In: *J. Comput. Syst. Sci.* 32.2 (1986), pp. 183–221. DOI: 10.1016/0022-0000(86)90026-7.

[WVS83] Pierre Wolper, Moshe Y. Vardi and A. Prasad Sistla. 'Reasoning about Infinite Computation Paths (Extended Abstract)'. In: *FOCS*. IEEE Computer Society, 1983, pp. 185–194. DOI: 10.1109/SFCS.1983.51.