A Unified Translation of Linear Temporal Logic to ω-Automata

JAVIER ESPARZA, JAN KŘETÍNSKÝ, and SALOMON SICKERT,
Technische Universität München, Germany

We present a unified translation of linear temporal logic (LTL) formulas into deterministic Rabin automata (DRA), limit-deterministic Büchi automata (LDBA), and nondeterministic Büchi automata (NBA). 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. In particular, Safra's, ranking, and breakpoint constructions used in other translations are not needed. We further give evidence that this theoretical clean and compositional approach does not lead to large automata per se and in fact in the case of DRAs yields significantly smaller automata compared to the previously known approach using determinisation of NBAs.

CCS Concepts: • **Theory of computation → Automata over infinite objects**; **Modal and temporal logics**;

Additional Key Words and Phrases: Linear temporal logic, automata over infinite words, deterministic automata, nondeterministic automata

ACM Reference format:
Javier Esparza, Jan Křetínský, and Salomon Sickert. 2020. A Unified Translation of Linear Temporal Logic to ω-Automata. *J. ACM* 67, 6, Article 33 (October 2020), 61 pages.
https://doi.org/10.1145/3417995

1 INTRODUCTION

Linear temporal logic (LTL) [63] is a prominent specification language, used both for model checking and the automatic synthesis of reactive systems. In the automata-theoretic approach to model checking, the specification formula is first translated into a (nondeterministic) ω-automaton, and then the product of this automaton with the system is further analysed [80]; similarly, in the automata-theoretic approach to synthesis the specification is translated into a deterministic ω-automaton [65]. In both cases, the size (in terms of states, transitions, or acceptance condition complexity) of the product and the deterministic automaton, respectively, is often the bottleneck

Authors are listed in alphabetical order, and the order does not indicate primary authorship.
This work is (partly) supported by the German Research Foundation (DFG) project "Verified Model Checkers" (317422601) and (partly) funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 787367 (PaVeS).
Authors' addresses: J. Esparza, J. Křetínský, and S. Sickert, Institut für Informatik (I7), Technische Universität München, Boltzmannstraße 3, D-85748 Garching bei München / Germany; emails: esparza@in.tum.de, {jan.kretinsky, s.sickert}@tum.de.

0004-5411/2020/10-ART33
https://doi.org/10.1145/3417995

of the approach. Consequently, much effort has been invested on translating LTL into small automata. Until recently, most of this work has focused on nondeterministic Büchi automata (NBA), which are suitable for model-checking [6, 16–18, 24, 25, 28, 30, 31, 75]. However, automatic synthesis and other applications, like model-checking probabilistic systems, require either deterministic automata or at least automata with some degree of determinism. This fundamental question has led to the introduction and analysis of numerous automata classes, such as deterministic parity automata (DPA) and deterministic Rabin automata (DRA) [7], deterministic generalised Rabin automata (DGRA) [13], limit-deterministic Büchi automata[1] (LDBA) [15, 33, 72, 78], unambiguous Büchi automata (UBA) [8, 37], good-for-games automata [35], and good-for-Markov-decision-processes automata [34].

The usual translations of LTL to deterministic ω-automata first convert the formula into an NBA and then apply some variant of Safra's determinisation procedure [61, 66, 67]. Once the NBA is constructed, such translations "forget" the formula, and so in the determinisation step they cannot take advantage of the formula structure or the fact that LTL only captures a subset of the ω-regular languages. Further, they are known to be difficult to implement efficiently and to be practically inefficient in many cases due to their generality. For this reason, two of the authors of this article initiated in Reference [45] a line of research on how to directly translate LTL to DRAs and DGRAs. This idea was then picked up by the community and since then several constructions to DRAs and DGRAs [20, 22, 45, 46], LDBAs [38, 39, 72], or DPAs [21, 49], without an intermediate step through NBAs, have been proposed. All these works share the principle of describing each state by a collection of formulas, as happens in the classical tableaux construction for the translation of LTL to NBA. This makes the approach particularly apt for semantics-based state reductions, e.g., for merging states corresponding to equivalent formulas. Such reductions cannot be applied to Safra-based constructions, where this semantic structure gets lost.

Example: The Challenge of Determinism

Every LTL formula has an equivalent NBA, but even simple formulas like $\mathbf{FG}a$ have no equivalent deterministic Büchi automaton. This already indicates that translating LTL to ω-automata is substantially harder in the deterministic case. We discuss the fundamental difficulties with the help of some examples.

Classic translation procedures from LTL to (nondeterministic) automata are based on tableaux. Intuitively, they construct automata that, when reading a word, monitor what remains to be satisfied of the original formula. To this end, their transition function uses the expansion laws for LTL to decompose the current formula into a part that can be checked on the current letter and the rest to be checked in future. For instance, the expansion law $a \,\mathbf{U}\, b \equiv b \vee (a \wedge \mathbf{X}(a \,\mathbf{U}\, b))$ for the "until" operator is used to determine that, if a word that starts with the letter $\{a\}$ must satisfy $c \vee a \,\mathbf{U}\, b \equiv c \vee b \vee (a \wedge \mathbf{X}(a \,\mathbf{U}\, b))$, then after reading $\{a\}$ the rest of the form must still satisfy $a \,\mathbf{U}\, b$.

In the nondeterministic case, the disjunctions in a formula produced by this process can be turned into a nondeterministic choice among the outgoing transitions of the state corresponding to the formula. For instance, a word starting with $\{a\}$ can satisfy the formula $\varphi = \mathbf{FG}a \equiv \mathbf{XFG}a \vee (\mathbf{XG}a \wedge a)$ by satisfying the first or the second disjunct. The corresponding nondeterministic automaton can stick to φ after reading $\{a\}$ (first disjunct), or move to $\mathbf{G}a$ (second disjunct), as illustrated in Figure 1 (left).

[1]Some authors refer to these automata as semi-deterministic or deterministic-in-the-limit automata. However, we use the term semi-deterministic automata in the way it was defined by Reference [81].

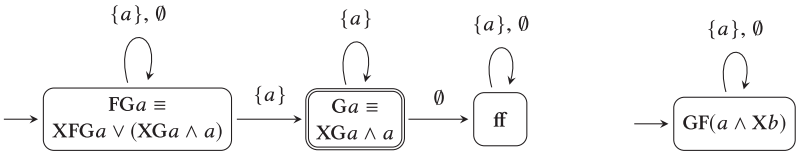

Fig. 1. NBA for $\mathbf{FG}a$ and an incorrect try for a DBA for $\mathbf{GF}(a \wedge \mathbf{X}b)$.

The possibility to map disjunctions to nondeterministic choices allows one to translate formulas into nondeterministic automata with very simple acceptance conditions, like NBAs. In contrast, deterministic automata cannot resolve disjunctions in this way, by definition. This has two consequences: (i) All disjuncts have to be treated simultaneously. (ii) Applying the transition function to the whole disjunction is problematic. For example, for the above-mentioned formula $\varphi = \mathbf{FG}a$, we have $\varphi \equiv \mathbf{XFG}a$; indeed, whether a word w satisfies φ or not does not depend on any finite prefix of w. Therefore, applying the transition function to φ yields an automaton with one single state and two self-loop transitions labeled by $\emptyset$ and $\{a\}$, which does not recognise the language of φ.[2]

This problem stands out even more clearly with the formula $\varphi = \mathbf{GF}(a \wedge \mathbf{X}b)$, as illustrated in Figure 1 (right). As for $\mathbf{FG}a$, satisfaction of φ does not depend on any finite prefix, and so the automaton has again one single state. Such an automaton cannot even detect that an infix of a word satisfies $a \wedge \mathbf{X}b$, because this is a property stretching over two steps. This example suggests to monitor not only the evolution of the complete formula, but *also the evolution of its subformulas*, such as $a \wedge \mathbf{X}b$, to check which ones are (repeatedly) satisfied and which ones not.

A consequence of (i) is that disjunctions in the formula must be handled at the level of the acceptance condition. For instance, consider the formula $\varphi = \mathbf{F}a \wedge (\mathbf{FG}b \vee \mathbf{FG}c)$. Due to the disjunction, there are two ways to satisfy the formula, depending on whether the first or the second disjunct holds. For each case, the automaton has to check that (a) it is indeed that particular case, and that (b) given the case, φ is satisfied. Our translation follows this pattern of a big disjunction over the possible cases, in each of which several simpler properties are checked. For this, our "Master Theorem" decomposes the language of any LTL formula φ into a Boolean combination

$$\mathcal{L}(\varphi) = \bigcup_{i=1}^{n} U_i \cap V_i$$

of simpler languages U_i, V_i for (a) and (b), respectively. As a result, the formula can be translated into an automaton as a union of intersections of simpler automata.

A Unified Translation of LTL to ω-Automata

Our contribution is a unified approach to the translation of LTL to NBAs, LDBAs, and DRAs that further enjoys the following properties, discussed below: semantic state labels, compositionality, asymptotic optimality, symmetry, independence of syntactic restrictions, and practical relevance.

Unified Approach. Our translations rely on a novel *Master Theorem*, which decomposes the language of a formula into a positive Boolean combination of *plain* languages,[3] easy to translate into automata. Not only is the approach simpler than previous ones [22, 72], but it also results in very similar algorithms for the translation into DRAs, NBAs, and LDBAs, differing only in the translations of the plain languages. More precisely, the automaton for the formula is obtained from

[2]In other words, the language of φ has a trivial right congruence, in the sense of, e.g., Reference [2], thus yielding only a trivial automaton.

[3]Corresponding to the "safety," "guarantee" (co-safety), "recurrence," and "persistence" classes of LTL formulas [56, 74].

automata recognising the plain languages by means of standard operations for closure under union and intersection. Intuitively, the user of the Master Theorem provides translations to the target class of automata for simple fragments of LTL and obtains a compositional translation for all LTL formulas.

Semantic Translation. Our translations use the expansion laws for LTL operators, splitting a formula into a part immediately checkable on a current letter and the rest to be checked on the rest of the word. As a consequence, states can be described "semantically" in terms of a collection of formulas, similarly to the classic tableaux-based translations or to derivatives of regular expressions [11] as used for LTL [76]. This allows us to apply efficient semantics-based state reductions. For example, states corresponding to equivalent formulas can be merged. Such semantics-based reductions cannot be applied in general to Safra-based constructions, because the semantic structure gets lost in translation. Besides, semantic description of states can be further exploited for generic [55] and learning-based heuristics [42] to speed up synthesis.

Compositionality. In contrast to the monolithic Safra-based or derivatives-based approaches, the decomposition by the Master Theorem yields a translation procedure in which the final automaton is a Boolean combination of many, typically small, automata.

Asymptotic Optimality. Deterministic generalised Rabin automata are the most compact among the deterministic automata used in practice,[4] in particular compared to deterministic parity automata (DPA). Previous translations to D(G)RA were either limited to fragments of LTL [5, 45, 46] or had triple exponential complexity [20, 22]. 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 or NBAs. Thus, our construction is the first direct translation from LTL to D(G)RAs with a proven double exponential size bound.

Symmetry. The first direct translations of LTL to deterministic automata used auxiliary automata to monitor each **F**- (*Finally*) and **G**- (*Globally*) subformula [45, 46]. While this approach worked for fragments of LTL, subsequent constructions for full LTL [20, 22, 72] 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., breakpoint constructions similar to Reference [58]. 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 in 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 unnecessarily large formulas. While our translations still require 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 or even better in practice than previous translations to NBAs, LDBAs, and DRAs. The LTL synthesis tool `Strix` [55, 57], which won the LTL synthesis track of the 2018, 2019, and 2020 editions of the SyntComp competition [36], picks from a portfolio of the translations presented here (LDBAs, DBAs, DCAs, DRAs) and the LDBA-to-DPA algorithm of Reference [21] to translate formulas into

[4] In theory, Emerson-Lei automata can be more succinct [59], but they are not yet used widely in practice.

DPAs. Moreover, we show that the special structure of the LDBAs delivered by the translation makes them suitable for the verification of probabilistic models.

Summarising, we think this work finally achieves the goals formulated in Reference [45], where the first translation of this kind—valid only for what is in comparison only a small fragment of LTL—was presented.

Origin of the Results

The first direct translation from LTL to deterministic Rabin automata, already containing some of the main ideas of this article, was presented in Reference [45] for a fragment of LTL, and extended to the whole logic in References [20, 22]. However, in the extended case, we were only able to prove a triple exponential bound on the size of the automaton. This problem was overcome in References [23, 69], which also showed how to uniformly translate LTL to other models. The results of Reference [23] are presented in more detail and extended in the dissertation of the third author [69]. This article is based on References [23] and [69]. The proofs of the central theorems have been formally verified, and a verified program for the translation of LTL to DRAs has been extracted from the proofs [10].

Structure of the Article

The article is organised as follows: Section 2 introduces fundamental terminology and results about ω-automata and LTL. Section 3 recalls the "after"-function from References [20, 22, 72], and Section 4 shows how to use it to construct DRAs for fragments of LTL. Section 5 presents the central result—the Master Theorem—that decomposes LTL formulas into simpler fragments. Sections 6 to 8 use the Master Theorem to extend the constructions of Section 4 to full LTL. Section 9 highlights some applications of the translations of the previous sections, and Section 10 compares experimentally the new translation to DRAs with other approaches.

2 PRELIMINARIES

Given a set A whose elements are sets, we let $\bigcup A$ denote the union of all the elements of A.

A word w over a finite alphabet Σ is an infinite sequence of letters $a_0a_1a_2\ldots$ where $a_i \in \Sigma$ for all $i \geq 0$. A language is a set of words. The set of all words is denoted Σ^ω. Given a word w, the ith letter of w is denoted by $w[i]$ (starting at 0), the finite infix $w[i]w[i+1]\ldots w[j-1]$ by w_{ij}, and the infinite suffix $w[i]w[i+1]\ldots$ by w_i. The following identities are useful: $w_{i(i+1)} = w[i]$, $w_{(i+j)i} = \epsilon$ and $w_{0i}w_i = w$ for all $i, j \geq 0$. Finally, we denote the infinite repetition of a finite word $\sigma_1\ldots\sigma_n$ by $(\sigma_1\ldots\sigma_n)^\omega = \sigma_1\ldots\sigma_n\sigma_1\ldots\sigma_n\sigma_1\ldots$.

2.1 ω-Automata

For the sake of presentation, we focus on ω-automata with acceptance conditions defined on states rather than on transitions. It is straightforward to modify our constructions so they yield automata with transition-based acceptance conditions.

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, $\Delta : Q \times \Sigma \to 2^Q$ is a transition function, and Q_0 is a set of initial states. A transition is a triple (q, a, q') such that $q' \in \Delta(q, a)$. Sometimes, we also view the transition function as a relation $\Delta \subseteq Q \times \Sigma \times Q$. 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 $a \in \Sigma$.

A *run* of $\mathcal{P}$ on a word w is an infinite sequence of states $r = q_0q_1q_2\ldots$ such that $q_0 \in Q_0$ and $q_{i+1} \in \Delta(q_i, w[i])$ for all $i \geq 0$. We denote the set of states occurring infinitely often in r by $\mathrm{Inf}(r)$.

An *acceptance condition* is an expression over the syntax:

$$\alpha ::= \mathrm{Inf}(S) \mid \mathrm{Fin}(S) \mid \alpha_1 \vee \alpha_2 \mid \alpha_1 \wedge \alpha_2 \quad \text{with } S \subseteq Q.$$

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

$$\begin{array}{lll} r \models \mathrm{Inf}(S) & \text{iff} & \mathrm{Inf}(r) \cap S \neq \emptyset, \\ r \models \mathrm{Fin}(S) & \text{iff} & \mathrm{Inf}(r) \cap S = \emptyset, \\ r \models \alpha_1 \vee \alpha_2 & \text{iff} & r \models \alpha_1 \textit{ or } r \models \alpha_2, \\ r \models \alpha_1 \wedge \alpha_2 & \text{iff} & r \models \alpha_1 \textit{ and } r \models \alpha_2. \end{array}$$

An acceptance condition α is a

- Büchi condition if $\alpha = \mathrm{Inf}(S)$ for some set $S \subseteq Q$ of states.
- co-Büchi condition if $\alpha = \mathrm{Fin}(S)$ for some set $S \subseteq Q$ of states.
- Rabin condition if $\alpha = \bigvee_{i=1}^{k} (\mathrm{Fin}(F_i) \wedge \mathrm{Inf}(I_i))$ 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 $\alpha = \bigvee_{i=1}^{k} (\mathrm{Fin}(F_i) \wedge \bigwedge_{j=1}^{l_i} \mathrm{Inf}(I_{ij}))$ 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 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 $\mathcal{L}(\mathcal{A}) := \{w \in \Sigma^\omega : w \text{ is accepted by } \mathcal{A}\}$. An ω-automaton is a Büchi (co-Büchi, Rabin, generalised Rabin) automaton if its acceptance condition is a Büchi (co-Büchi, Rabin, generalised Rabin) condition.

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, and a deterministic component. The automaton can only accept by switching from the *initial* (nondeterministic) to the *accepting* (deterministic) component, but after the switch it must stay in the deterministic component forever. Formally, an NBA $\mathcal{B} = (Q, \Delta, Q_0, \mathrm{Inf}(S))$ 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) $S \subseteq Q_\mathcal{D}$.

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 the acceptance condition by B (Büchi), C (co-Büchi), R (Rabin), or GR (generalised Rabin).

Visual Representation. In the following sections, we consider automata over alphabets of the form $\Sigma = 2^X$ 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) [26, 63, 64] with the following reduced syntax:

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

where a is an element of a finite set Ap of atomic propositions. Left-out, but often used LTL operators are then added as abbreviations. This reduced syntax has the advantage that only a few cases have to be considered in proofs by structural induction. However, our result needs an LTL syntax containing **W** (weak until) in addition to **U** (until), and such that formulas are 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) so negations can be pushed inwards:

Definition 1 (LTL).

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

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

$$\varphi \equiv \psi := (\mathcal{L}(\varphi) = \mathcal{L}(\psi)).$$

The semantics makes 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. Note that we can express the **W** and **M** modalities using **R** and **U** modalities as follows:

$$\varphi \,\mathbf{W}\, \psi \equiv \psi \,\mathbf{R}\, (\varphi \vee \psi) \qquad \varphi \,\mathbf{M}\, \psi \equiv \psi \,\mathbf{U}\, (\varphi \wedge \psi).$$

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.[5] Last, we use the two common abbreviations $\mathbf{F}\varphi := \mathbf{tt}\,\mathbf{U}\,\varphi$ (eventually) and $\mathbf{G}\varphi := \mathbf{ff}\,\mathbf{R}\,\varphi$ (always) with well-known semantics:

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

2.2.1 Propositional Semantics. In addition to the language-based semantics of LTL, we assign propositional semantics to LTL formulas by treating *temporal* subformulas as propositional variables. A formula ψ is called *temporal* if it is neither a conjunction nor a disjunction, i.e., if the root of its syntax tree is labelled by either a temporal operator (**U**, **R**, **M**, **W**, or **X**) or a, $\neg a$. We denote by $\mathrm{sf}(\varphi)$ the set of temporal subformulas of φ. Formally, we define the propositional semantics as follows:

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

Definition 2 (Propositional Semantics of LTL). Let $\mathcal{I}$ be a set of LTL formulas and let φ be an LTL formula. The propositional satisfaction relation $\mathcal{I} \models_p \varphi$ is inductively defined as follows:

$$\begin{array}{llll} \mathcal{I} \models_p \mathbf{tt} & & \mathcal{I} \models_p \varphi \wedge \psi & \text{iff } \mathcal{I} \models_p \varphi \text{ and } \mathcal{I} \models_p \psi \\ \mathcal{I} \not\models_p \mathbf{ff} & & \mathcal{I} \models_p \varphi \vee \psi & \text{iff } \mathcal{I} \models_p \varphi \text{ or } \mathcal{I} \models_p \psi \\ \mathcal{I} \models_p a & \text{iff } a \in \mathcal{I} & \mathcal{I} \models_p \mathbf{X}\varphi & \text{iff } \mathbf{X}\varphi \in \mathcal{I} \\ \mathcal{I} \models_p \neg a & \text{iff } \neg a \in \mathcal{I} & \mathcal{I} \models_p \varphi \; op \; \psi & \text{iff } \varphi \; op \; \psi \in \mathcal{I} \quad \text{for } op \in \{\mathbf{U}, \mathbf{M}, \mathbf{R}, \mathbf{W}\}. \end{array}$$

We also call the set $\mathcal{I}$ a propositional assignment. Two formulas φ, ψ are *propositionally equivalent*, denoted $\varphi \sim \psi$, if $\mathcal{I} \models_p \varphi \iff \mathcal{I} \models_p \psi$ holds for all sets $\mathcal{I}$. The *(propositional) equivalence class* of a formula φ is denoted $[\varphi]_\sim$ and defined as $[\varphi]_\sim := \{\psi : \varphi \sim \psi\}$. The *(propositional) quotient set* of a set of formulas Ψ is denoted $\Psi_{/\sim}$ and defined as $\Psi_{/\sim} := \{[\psi]_\sim : \psi \in \Psi\}$.

Example 3. 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 $\varphi \sim \psi_1$. Thus, $\mathbf{X}b$ is propositionally equivalent to φ and $\mathbf{X}b$ is an element of the equivalence class $[\varphi]_\sim$.

Observe that $\models$ and $\models_p$ interpret $\mathbf{tt}$, $\mathbf{ff}$, $\wedge$, and $\vee$ in the same way. Note also that there are no consistency requirements for the set $\mathcal{I}$, in particular, it may contain both a and $\neg a$. Hence, we have $\{a, \neg a\} \models_p a \wedge \neg a$, even though we also have $a \wedge \neg a \equiv \mathbf{ff}$.

We conclude the section with two properties of propositional semantics that will be useful later. The following lemma is proved by a straightforward structural induction on φ.

Lemma 4. *Let φ be a formula and let w be word. Then:*

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

Example 5. Let $\varphi = a \vee \mathbf{F}b$ and $w = (\emptyset\{b\})^\omega$. In this case, we have $w \models \varphi$, $w \models \mathbf{F}b$, and $\{\mathbf{F}b\} \models_p \varphi$.

The second property states that propositional equivalence is a congruence for any function from formulas to formulas that maps $\mathbf{tt}$ and $\mathbf{ff}$ to themselves, and preserves disjunctions and conjunctions. The proof can be found in Appendix A.

Lemma 6. *Let f be a function on formulas such that $f(\mathbf{tt}) = \mathbf{tt}$, $f(\mathbf{ff}) = \mathbf{ff}$, and $f(\chi_1 \wedge \chi_2) = f(\chi_1) \wedge f(\chi_2)$, $f(\chi_1 \vee \chi_2) = f(\chi_1) \vee f(\chi_2)$ for all formulas χ_1 and χ_2. For every pair of formulas φ and ψ, if $\varphi \sim \psi$, then $f(\varphi) \sim f(\psi)$.*

In particular, Lemma 6 shows that propositional equivalence (according to our Definition 2) is preserved by substitution of a temporal subformula by another arbitrary formula.

2.2.2 Four LTL-Fragments: μLTL, νLTL, $\mathbf{GF}(\mu LTL)$, and $\mathbf{FG}(\nu LTL)$. The following four fragments of LTL play a central role in this article. They are the "simple" fragments of LTL mentioned in the introduction.

- The fragment μLTL and the fragment νLTL.
  μLTL is the fragment of LTL restricted to temporal operators $\mathbf{U}$ and $\mathbf{M}$, on top of Boolean connectives ($\wedge, \vee$), literals ($a, \neg a$), and the next operator ($\mathbf{X}$). νLTL is defined analogously, but with the operators $\mathbf{R}$ and $\mathbf{W}$. 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.

The claim that these languages are "simple" is substantiated in Section 4, which gives a straightforward translation to (deterministic) automata. 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. So μLTL and νLTL are the formulas of LTL without any alternation of least and greatest fixed-points, while $\mathbf{GF}(\mu LTL)$ and $\mathbf{FG}(\nu LTL)$ are fragments containing one single alternation.

3 THE "AFTER"-FUNCTION

The function $\mathrm{aft}(\varphi, w)$, read "φ after w," is the foundation for the translations to automata presented here [20, 22, 23]. The function assigns to a formula φ and a finite word w a formula $\mathrm{aft}(\varphi, w)$ such that, intuitively, φ holds for a word ww' if and only if $\mathrm{aft}(\varphi, w)$ holds "after reading w," that is, if and only if $w' \models \mathrm{aft}(\varphi, w)$. The definition of $\mathrm{aft}(\varphi, w)$ follows easily from the expansion laws for LTL.

Definition 7. Let φ be a formula and $\sigma \in 2^{Ap}$ a single letter. The formula $\mathrm{aft}(\varphi, \sigma)$ is inductively defined as follows:

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

Furthermore, we generalise aft to finite words by defining

$$\mathrm{aft}(\varphi, \epsilon) := \varphi \qquad \text{and} \qquad \mathrm{aft}(\varphi, \sigma w) := \mathrm{aft}(\mathrm{aft}(\varphi, \sigma), w)$$

for every $\sigma \in 2^{Ap}$ and every finite word w. Finally, we define the set of formulas *reachable from* φ as $\mathrm{Reach}(\varphi) := \{\mathrm{aft}(\varphi, w) : w \in (2^{Ap})^*\}$.

Definitions for **F** *and* **G**. In several examples, we use formulas containing **F** and **G**. To keep the examples readable, we introduce the following shortened definitions:

$$\mathrm{aft}(\mathbf{F}\varphi, \sigma) := \mathrm{aft}(\varphi, \sigma) \vee \mathbf{F}\varphi \qquad\qquad \mathrm{aft}(\mathbf{G}\varphi, \sigma) := \mathrm{aft}(\varphi, \sigma) \wedge \mathbf{G}\varphi.$$

Example 8. Let $\varphi = a \vee (b\, \mathbf{U}\, c)$ be a formula. We then have $\mathrm{aft}(\varphi, \{a\}) = \mathbf{tt} \vee (\mathbf{ff} \vee (\mathbf{ff} \wedge b\, \mathbf{U}\, c)) \sim \mathbf{tt}$, $\mathrm{aft}(\varphi, \{b\}) = \mathbf{ff} \vee (\mathbf{ff} \vee (\mathbf{tt} \wedge b\, \mathbf{U}\, c)) \sim b\, \mathbf{U}\, c$, and $\mathrm{aft}(\varphi, \emptyset) = \mathbf{ff} \vee (\mathbf{ff} \vee (\mathbf{ff} \wedge b\, \mathbf{U}\, c) \sim \mathbf{ff}$.

Lemma 9. *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:*

$$ww' \models \varphi \iff w' \models \mathrm{aft}(\varphi, w).$$

Proof. We prove by induction on φ that the property holds for a single letter $\sigma \in 2^{Ap}$. The result for an arbitrary finite word w is then shown by induction on the length of w. The single-letter case requires to prove:

$$\sigma w' \models \varphi \iff w' \models \mathrm{aft}(\varphi, \sigma).$$

We only show two representative cases of the induction.

- Case $\varphi = a$:

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

- Case $\varphi = \psi_1 \,\mathbf{U}\, \psi_2$:

$$\begin{aligned} & \sigma w' \models \varphi \\ \iff\ & \sigma w' \models \psi_2 \vee (\psi_1 \wedge \mathbf{X}\varphi) && \text{(LTL expansion)} \\ \iff\ & w' \models \mathrm{aft}(\psi_2, \sigma) \vee (\mathrm{aft}(\psi_1, \sigma) \wedge \varphi) && \text{(induction hypothesis)} \\ \iff\ & w' \models \mathrm{aft}(\varphi, \sigma) \end{aligned}$$

□

Lemma 9 was introduced already in References [22, 45] and describes the main purpose of the after function. It can be read as $w^{-1}\mathcal{L}(\varphi) = \mathcal{L}(\mathrm{aft}(\varphi, w))$, where $w^{-1}L := \{w' : ww' \in L\}$ denotes the derivative under w of the language L. Thus, aft computes the derivative of a language, represented by a formula.

It is easy to give formulas φ for which the set $\mathrm{Reach}(\varphi)$ contains infinitely many syntactically different formulas, e.g., $\varphi = a \,\mathbf{U}\, b$. The following lemma shows that $\mathrm{Reach}(\varphi)$ only contains finitely many formulas up to propositional equivalence.

Lemma 10. *Let φ and ψ be formulas and let w be a finite word. Then:*

(1) $\mathrm{sf}(\mathrm{aft}(\varphi, w)) \subseteq \mathrm{sf}(\varphi)$

(2) If φ has n temporal subformulas, then $\mathrm{Reach}(\varphi)_{/\sim}$ has 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. (1) Intuitively, this holds, because aft does not create new elements in the syntax tree, except Boolean combinations of existing temporal subformulas. The formal proof proceeds by a straightforward nested induction on φ and on the length of w.

(2) Let φ be a formula with n temporal subformulas. By (1), every formula of $\mathrm{Reach}(\varphi)$ is a positive Boolean combination of temporal subformulas of φ. So each equivalence class of $\mathrm{Reach}(\varphi)_{/\sim}$ can be interpreted as a monotonic Boolean function over n variables. There are at most $M(n)$ such functions and so $|\mathrm{Reach}(\varphi)_{/\sim}| \leq M(n)$. Furthermore, $M(n)$ can be bounded by 2^{2^n}, since there exist at most that many Boolean functions over n variables. □

By Lemma 10.2, the set $\mathrm{Reach}(\varphi)_{/\sim}$ is finite. We show how to compute it. For every $k \geq 0$ define

$$\mathrm{Reach}(\varphi)^k_{/\sim} = \{\, [\mathrm{aft}(\varphi, w)]_\sim : w \in (2^{Ap})^* \wedge |w| \leq k \}.$$

By definition, we have $\mathrm{Reach}(\varphi)_{/\sim} = \bigcup_{k \geq 0} \mathrm{Reach}(\varphi)^k_{/\sim}$, and $\mathrm{Reach}(\varphi)^k_{/\sim} \subseteq \mathrm{Reach}(\varphi)^{k+1}_{/\sim}$ for every $k \geq 0$. So there is an index $\alpha \leq M(n)$ such that $\mathrm{Reach}(\varphi)^\alpha_{/\sim} = \mathrm{Reach}(\varphi)^{\alpha+1}_{/\sim}$. We prove that $\mathrm{Reach}(\varphi)_{/\sim} = \mathrm{Reach}(\varphi)^\alpha_{/\sim}$. It suffices to show $\mathrm{Reach}(\varphi)^\alpha_{/\sim} = \mathrm{Reach}(\varphi)^{\alpha+j}_{/\sim}$ for every $j \geq 1$. We proceed by induction. The case $j = 1$ is true by hypothesis. For $j > 1$, we apply the following lemma, which follows immediately from Lemma 6 by taking $f(\varphi) = \mathrm{aft}(\varphi, \sigma)$:

Lemma 11. *For all formulas φ, ψ and every letter σ, if $\varphi \sim \psi$, then $\mathrm{aft}(\varphi, \sigma) \sim \mathrm{aft}(\psi, \sigma)$.*

The lemma allows us to lift the after function to equivalence classes of formulas by defining $\mathrm{aft}_\sim([\varphi]_\sim, \sigma) := [\mathrm{aft}(\varphi, \sigma)]_\sim$ for every formula φ. We then have:

$$\begin{aligned} \mathrm{Reach}(\varphi)^{\alpha+j}_{/\sim} &= \{\mathrm{aft}_\sim([\psi]_\sim, \sigma) : [\psi]_\sim \in \mathrm{Reach}(\varphi)^{\alpha+j-1}_{/\sim} \wedge \sigma \in 2^{Ap}\} && \text{(Lemma 11)} \\ &= \{\mathrm{aft}_\sim([\psi]_\sim, \sigma) : [\psi]_\sim \in \mathrm{Reach}(\varphi)^{\alpha}_{/\sim} \wedge \sigma \in 2^{Ap}\} && \text{(induction hypothesis)} \\ &= \mathrm{Reach}(\varphi)^{\alpha+1}_{/\sim} && \text{(Lemma 11)} \\ &= \mathrm{Reach}(\varphi)^{\alpha}_{/\sim}. \end{aligned}$$

Related Work. The definition of the after function is almost identical to the transition relation of very-weak alternating automata constructed from LTL formulas [60, 79]. Thus, the function 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" [7] to construct the final result.

As mentioned in the introduction, regular-expression derivatives are an elegant technique for translating regular expressions to deterministic [11] as well as nondeterministic [3] finite automata. The after function follows in these footsteps for LTL.

4 DBAS AND DCAS FOR μLTL, νLTL, GF(μLTL), AND FG(νLTL)

We show that the after function can be used to characterise the languages of formulas of simple LTL fragments. We use this result to define simple translations of these fragments to DBAs or DCAs.

Lemma 12 ([22, 23]). *Let φ be a formula and let w be a word.*

(1) If $\varphi \in \mu LTL$, then $w \models \varphi \iff \exists i.\, \mathrm{aft}(\varphi, w_{0i}) \sim \mathbf{tt}$.
(2) If $\varphi \in \nu LTL$, then $w \models \varphi \iff \forall i.\, \mathrm{aft}(\varphi, w_{0i}) \nsim \mathbf{ff}$.

Proof. We only present the proof for (1) and skip the proof for (2), which is analogous. Let w be a word and let φ be a formula with $\varphi \in \mu LTL$.

($\Rightarrow_1$) Assume $w \models \varphi$. We proceed by structural induction on φ to prove the existence of an i such that $\mathrm{aft}(\varphi, w_{0i}) \sim \mathbf{tt}$. We only consider two of the possible cases: $\mathbf{tt}, \mathbf{ff}, a, \neg a, \psi_1 \wedge \psi_2, \psi_1 \vee \psi_2, \mathbf{X}\psi, \psi_1 \,\mathbf{U}\, \psi_2, \psi_1 \,\mathbf{M}\, \psi_2$. The others are similar.

- Case $\varphi = a$: Since $w \models \varphi$, we have $a \in w[0] = w_{01}$ and we get $\mathrm{aft}(\varphi, w_{01}) = \mathbf{tt} \sim \mathbf{tt}$.
- 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 $\mathrm{aft}(\psi_1, w_{\ell i}) \sim \mathbf{tt}$ and there exists an $i \geq k$ such that $\mathrm{aft}(\psi_2, w_{ki}) \sim \mathbf{tt}$. Let j be the maximum of all those i's. We prove $\mathrm{aft}(\psi_1 \,\mathbf{U}\, \psi_2, w_{0j}) \sim \mathbf{tt}$ via induction on k.
  $-k = 0$.

$$\begin{aligned} & \mathrm{aft}(\psi_1 \,\mathbf{U}\, \psi_2, w_{0j}) \\ = \;& \mathrm{aft}(\psi_2, w_{0j}) \vee (\mathrm{aft}(\psi_1, w_{0j}) \wedge \mathrm{aft}(\psi_1 \,\mathbf{U}\, \psi_2, w_{1j})) && \text{(Definition 7)} \\ \sim \;& \mathbf{tt} \vee (\mathrm{aft}(\psi_1, w_{0j}) \wedge \mathrm{aft}(\psi_1 \,\mathbf{U}\, \psi_2, w_{1j})) && (k = 0 \to \mathrm{aft}(\psi_2, w_{0j}) \sim \mathbf{tt}) \\ \sim \;& \mathbf{tt} \end{aligned}$$

  $-k > 0$.

$$\begin{aligned} & \mathrm{aft}(\psi_1 \,\mathbf{U}\, \psi_2, w_{0j}) \\ = \;& \mathrm{aft}(\psi_2, w_{0j}) \vee (\mathrm{aft}(\psi_1, w_{0j}) \wedge \mathrm{aft}(\psi_1 \,\mathbf{U}\, \psi_2, w_{1j})) && \text{(Definition 7)} \\ \sim \;& \mathrm{aft}(\psi_2, w_{0j}) \vee (\mathbf{tt} \wedge \mathrm{aft}(\psi_1 \,\mathbf{U}\, \psi_2, w_{1j})) && (k > 0 \to \mathrm{aft}(\psi_1, w_{0j}) \sim \mathbf{tt}) \\ \sim \;& \mathrm{aft}(\psi_2, w_{0j}) \vee (\mathbf{tt} \wedge \mathbf{tt}) && \text{(induction hypothesis)} \\ \sim \;& \mathbf{tt} \end{aligned}$$

($\Leftarrow_1$) Assume $w \not\models \varphi$. By Lemma 9, we have $w_i \not\models \mathrm{aft}(\varphi, w_{0i})$ for all i, and thus $\mathrm{aft}(\varphi, w_{0i}) \not\equiv \mathbf{tt}$ for all i. Since $x \sim y$ implies $x \equiv y$, we also have $\mathrm{aft}(\varphi, w_{0i}) \nsim \mathbf{tt}$, and we are done. □

Proposition 13. *Let $\varphi \in \mu LTL$.*

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

$$\mathcal{B}_\mu^\varphi = (\mathrm{Reach}(\varphi)_{/\sim}, \mathrm{aft}_\sim, [\varphi]_\sim, \mathrm{Inf}(\{[\mathbf{tt}]_\sim\})).$$

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

$$C_\mu^\varphi = (\mathrm{Reach}(\varphi)_{/\sim}, \mathrm{aft}_\sim, [\varphi]_\sim, \mathrm{Fin}(\overline{\{[\mathbf{tt}]_\sim\}})).$$

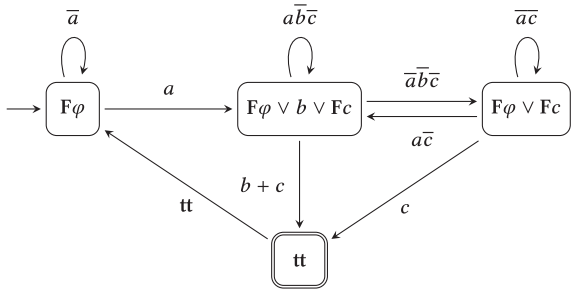

Fig. 2. DBA $\mathcal{B}^{\varphi}_{\mathrm{GF}\mu}$ for $\varphi = a \wedge \mathbf{X}(b \vee \mathrm{F}c)$ with the symbolic notation for transitions introduced in Section 2.1.

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

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

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

Let $\varphi \in \nu LTL$.

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

$$C^{\varphi}_{\nu} = (\mathrm{Reach}(\varphi)_{/\sim}, \mathrm{aft}_{\sim}, [\varphi]_{\sim}, \mathrm{Fin}(\{[\mathbf{ff}]_{\sim}\})).$$

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

$$\mathcal{B}^{\varphi}_{\nu} = (\mathrm{Reach}(\varphi)_{/\sim}, \mathrm{aft}_{\sim}, [\varphi]_{\sim}, \mathrm{Inf}(\overline{\{[\mathbf{ff}]_{\sim}\}})).$$

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

$$C^{\varphi}_{\mathrm{FG}\nu} = (\mathrm{Reach}(\mathbf{G}\varphi)_{/\sim}, \mathrm{aft}^{\mathrm{G}\varphi}_{\sim}, [\mathbf{G}\varphi]_{\sim}, \mathrm{Fin}(\{[\mathbf{ff}]_{\sim}\}))$$

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

Observe that $\mathcal{B}^{\varphi}_{\mu}$ and C^{φ}_{μ}, as well as $\mathcal{B}^{\varphi}_{\nu}$ and C^{φ}_{ν}, share the same structure and only differ in the definition of the acceptance condition. C^{φ}_{μ} can be understood as first applying the classic complementation procedure for deterministic finite automata (which simply swaps final and non-final states) to $\mathcal{B}^{\varphi}_{\mu}$, which yields a Büchi automaton recognising $\neg\varphi$, and then flipping the acceptance condition to finally arrive at the co-Büchi condition recognising $\neg\neg\varphi \equiv \varphi$. This works because 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 C^{φ}_{ν}. The proof of the proposition is straightforward and relies mostly on Lemma 12. It can be found in Appendix A.

Example 14. Let $\varphi = a \wedge \mathbf{X}(b \vee \mathrm{F}c) \in \mu LTL$. The DBA $\mathcal{B}^{\varphi}_{\mathrm{GF}\mu}$ constructed from Proposition 13 recognising $\mathcal{L}(\mathbf{GF}\varphi)$ is depicted in Figure 2. Notice that from now on, we label the states, which are in-fact equivalence classes of $\sim$, by a representative of the respective class.

5 THE MASTER THEOREM

We present the Master Theorem, a characterisation of the words satisfying a given (general LTL) formula, from which we can easily extract deterministic, limit-deterministic, and nondeterministic automata of asymptotically optimal size. We start with a motivational example, the simplest one outside of the fragments discussed above; it is also the simplest one not covered by the approaches

of References [45, 46]. We follow with an overview of the results, indicating in boldface several technical parts postponed until the subsequent subsections.

5.1 An Appetising Example

Consider the formula $\varphi = \mathbf{G}(a \vee b\,\mathbf{U}\,c)$, which does not belong to any of the fragments of Section 4.[6] Split the set of all words over the alphabet $2^{\{a,b,c\}}$ into words that satisfy $\mathbf{GF}(b\,\mathbf{U}\,c)$, and words that do not. First, let w be a word satisfying $\mathbf{GF}(b\,\mathbf{U}\,c)$. Since w satisfies c infinitely often, we have $w \models \mathbf{G}(a \vee b\,\mathbf{U}\,c)$ iff $w \models \mathbf{G}(a \vee b\,\mathbf{W}\,c)$. In other words, under the hypothesis $\mathbf{GF}(b\,\mathbf{U}\,c)$, checking φ reduces to checking $\mathbf{G}(a \vee b\,\mathbf{W}\,c)$. Since the hypothesis belongs to the $\mathbf{GF}(\mu LTL)$ fragment, and $\mathbf{G}(a \vee b\,\mathbf{W}\,c)$ is a formula of νLTL, we can construct automata for both. Second, let w be a word that does not satisfy $\mathbf{GF}(b\,\mathbf{U}\,c)$, in particular, there are only finitely many suffixes of w that do satisfy $b\,\mathbf{U}\,c$. What must w do to satisfy φ? Besides satisfying φ on a prefix of w, some suffix of w must satisfy $\mathbf{G}(a \vee \mathbf{ff})$, for which we can also construct an automaton. The Master Theorem generalises this idea to arbitrary formulas. It decomposes the language of any LTL formula φ into a Boolean combination

$$\mathcal{L}(\varphi) = \bigcup_{i=1}^{n} U_i \cap V_i, \tag{1}$$

where the U_i correspond to the different hypotheses, and the V_i to their corresponding satisfaction conditions. Crucially, these languages are *"plain"*, meaning that they correspond to formulas of the fragments $\mathbf{GF}(\mu LTL)$ and $\mathbf{FG}(\nu LTL)$, introduced in Section 2.2.2, and to a simple variation of νLTL. As a result, the formula can be translated into an automaton as a union of intersections of simple automata for formulas of these fragments.

5.2 Overview

First, we focus on "infinitary" formulas only, whose satisfaction does not depend on any finite prefix. Consider, e.g., $\varphi = \mathbf{G}(\mathbf{F}a \vee \mathbf{F}b)$. (While one can see that it could be re-written into the special fragment form as $\mathbf{GF}(a \vee b)$, its original form illustrates the general point here.) Similarly to the previous example, to determine whether $w \models \varphi$ holds it is useful to know which of the disjuncts holds infinitely often. While we may not know whether $\mathbf{GF}a$ or $\mathbf{GF}b$ holds, or none, or both, *restricting* $\mathcal{L}(\varphi)$ to any of these four possible cases makes deciding satisfaction easy. For instance, checking $w \models \varphi$ already knowing that $w \models \mathbf{GF}b$ holds is trivial. This suggests to decompose $\mathcal{L}(\varphi)$ into a union of languages, one for each possible case, where a case is determined by which subformulas hold *infinitely often* and which *almost always*. In each case, we can then use this "limit-behaviour" information.

5.2.1 Limit-behaviour Partition. Let $\mu(\varphi)$ and $\nu(\varphi)$ be the sets containing the subformulas of a fixed φ of the form $\psi_1\ op\ \psi_2$ for least-fixed-point operators $op \in \{\mathbf{U}, \mathbf{M}\}$ and greatest-fixed-point operators $op \in \{\mathbf{W}, \mathbf{R}\}$, respectively. Given a word w, let $\mathcal{GF}_w$ denote the set of formulas of $\mu(\varphi)$ that hold infinitely often along w (meaning that infinitely many of its suffixes satisfy the formulas), and define $\mathcal{FG}_w \subseteq \nu(\varphi)$ analogously, substituting "almost always" for "infinitely often." We say that two words w, v are *limit-equivalent* if $\mathcal{GF}_w = \mathcal{GF}_v$ and $\mathcal{FG}_w = \mathcal{FG}_v$. This equivalence relation induces a partition $\mathcal{P} = \{P_{M,N} : M \subseteq \mu(\varphi), N \subseteq \nu(\varphi)\}$, where $P_{M,N}$ consists of the words w such that $M = \mathcal{GF}_w$ and $N = \mathcal{FG}_w$.

Example 15. The partition for $\varphi = \mathbf{G}(\mathbf{F}a \vee \mathbf{F}b)$ is illustrated in Figure 3 (left) with vacuous cases denoted by crosses. Observe that the grey classes completely belong to the language and the white

[6] Moreover, it is a prototypical example of a formula outside of the $\mathrm{LTL}_{\setminus \mathbf{GU}}$ fragment [46], for which a simpler translation to DRAs exists [46] and for which singly exponential LDBA can be constructed [38].

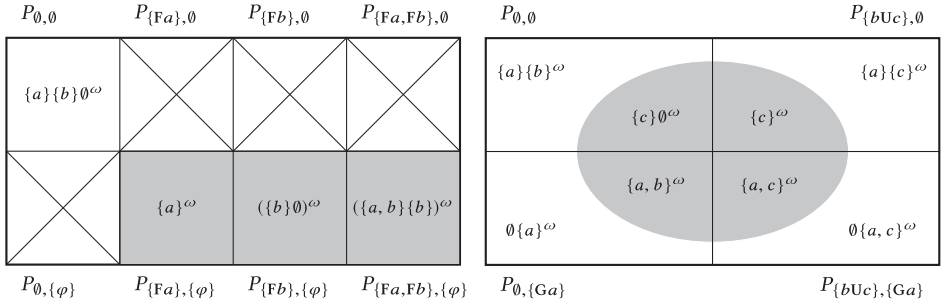

Fig. 3. Partition of Σ^ω according to the limit-behaviour with respect to $\varphi = \mathbf{G}(\mathbf{F}a \vee \mathbf{F}b)$ (left) and $\varphi = \mathbf{G}a \vee b\,\mathbf{U}\,c$ (right), with examples of words in the respective classes (crossed out if empty). The grey area denotes the language of the formula.

ones completely avoid it. This is generally true for infinitary properties. In contrast, for the non-infinitary property $a \wedge \mathbf{G}(\mathbf{F}a \vee \mathbf{F}b)$, the membership depends on the first letter. Consequently, for general formulas, we obtain also "mixed" classes as illustrated in Figure 3 (right) for $\varphi = \mathbf{G}a \vee b\,\mathbf{U}\,c$.

Following this idea, the decomposition takes the form of

$$\mathcal{L}(\varphi) = \bigcup_{\substack{M \subseteq \mu(\varphi) \\ N \subseteq \nu(\varphi)}} P_{M,N} \cap V_{M,N}, \tag{2}$$

where $V_{M,N}$ captures "$\mathcal{L}(\varphi)$ *given the case* M, N".

5.2.2 Stable Words. We first show how to decompose the language of *stable* words of $\mathcal{L}(\varphi)$. A word w is *stable* with respect to φ if every formula in $\mu(\varphi)$ holds either never or infinitely often along w (i.e., either none or infinitely many of its suffixes satisfy the formula), and every formula of $\nu(\varphi)$ fails never or infinitely often along w. In particular, no formula of $\mu(\varphi)$ can hold a finite, nonzero number of times before it fails forever, and no formula of $\nu(\varphi)$ can fail a finite, nonzero number of times before it holds forever. It is easy to see that, while not every word is stable, every word eventually stabilises, meaning that almost all its suffixes are stable. The **formal definition of stable words is presented in Section 5.3**.

Example 16. Consider again the previous example. The word $w = \{a\}\{b\}\emptyset\{a,c\}(\emptyset\{a\})^\omega$ is not stable with respect to $\mathbf{G}(\mathbf{F}a \vee \mathbf{F}b)$. However, w_i is stable for any $i \geq 2$, hence w stabilises at position 2. The word w is not stable with respect to $\mathbf{G}a \vee b\,\mathbf{U}\,c$ either, but w_4 is.

Given a language L, let L^s denote the set of stable words of L. Our subgoal is to decompose $\mathcal{L}(\varphi)^s$. Trivially:

$$\mathcal{L}(\varphi)^s = \bigcup_{\substack{M \subseteq \mu(\varphi) \\ N \subseteq \nu(\varphi)}} P_{M,N} \cap \mathcal{L}(\varphi)^s. \tag{3}$$

To follow Equation (2), we want to design plain formulas $\varphi[M]_\nu$ and $\varphi[N]_\mu$, from the fragments νLTL and μLTL, respectively, which utilise the "advice" to φ that the word is in the class of M, N:

$$\mathcal{L}(\varphi)^s = \bigcup_{\substack{M \subseteq \mu(\varphi) \\ N \subseteq \nu(\varphi)}} P_{M,N} \cap \mathcal{L}(\varphi[M]_\nu)^s = \bigcup_{\substack{M \subseteq \mu(\varphi) \\ N \subseteq \nu(\varphi)}} P_{M,N} \cap \mathcal{L}(\varphi[N]_\mu)^s. \tag{4}$$

Notice that we always utilise only one of M, N, since the point is to get rid of one of the fixed-point types, relying on the advice, and check the other one.

5.2.3 Utilising Advice on Stable Words. The desired **formulas $\varphi[M]_\nu \in \nu LTL$ and $\varphi[N]_\mu \in \mu LTL$ are defined in Section 5.4** from φ, M, and N by means of a simple syntactic transformation. Intuitively, the known advice is substituted into the formula.

Example 17. For $\varphi = \mathbf{F}(\mathbf{G}a \vee \mathbf{G}b)$ and advice $N = \{\mathbf{G}a\}$, thus not containing $\mathbf{G}b$, we shall obtain $\varphi[N]_\mu = \mathbf{F}(\mathbf{tt} \vee \mathbf{ff})$. The substitution is more complex for operators such as $\mathbf{U}$. For $\varphi = \mathbf{G}(a \vee b\,\mathbf{U}\,c)$ and advice $M = \{b\,\mathbf{U}\,c\}$ saying $b\,\mathbf{U}\,c$ holds infinitely often, it would be obviously wrong to simply plug in $\mathbf{tt}$ yielding $\mathbf{G}(a \vee \mathbf{tt})$, since we have to check *where* $b\,\mathbf{U}\,c$ holds. Nevertheless, we can consider $\mathbf{G}(a \vee b\,\mathbf{W}\,c)$ instead, where we are freed from checking the least fixed point from $\mathbf{U}$ and only check the local correctness of applying the expansion laws for until.

Even more surprisingly, for $\varphi = \mathbf{F}(a \vee b\,\mathbf{W}\,c)$, an advice that *misses* $b\,\mathbf{W}\,c$ makes us check $\mathbf{F}(a \vee b\,\mathbf{U}\,c)$, which is (logically) *stronger* than φ! However, it is plain (in μLTL) and logically equivalent provided $b\,\mathbf{W}\,c$ holds finitely often only.

Due to Equation (3), for Equation (4) to hold it suffices to show:

$$P_{M,N} \cap \mathcal{L}(\varphi)^s = P_{M,N} \cap \mathcal{L}(\varphi[M]_\nu)^s = P_{M,N} \cap \mathcal{L}(\varphi[N]_\mu)^s. \tag{5}$$

Section 5.5 proves (5). Now only two issues remain:

(i) $P_{M,N}$ might not be plain.
(ii) The decomposition has to be lifted back to all words, not just the stable ones.

5.2.4 Checking Advice by a Plain Language. The issue (i) concerning $P_{M,N}$ is overcome by means of two further results. By the definition of the partition, we have:

$$P_{M,N} = \mathcal{L}\left(\bigwedge_{\psi\in M} \mathbf{GF}\psi \wedge \bigwedge_{\psi\in N} \mathbf{FG}\psi \wedge \bigwedge_{\psi\in\mu(\varphi)\setminus M} \neg\mathbf{GF}\psi \wedge \bigwedge_{\psi\in\nu(\varphi)\setminus N} \neg\mathbf{FG}\psi\right).$$

While this language might not be plain, the previous syntactic substitution provides an alternative that is plain:

$$U_{M,N} := \mathcal{L}\left(\bigwedge_{\psi\in M} \mathbf{GF}(\psi[N]_\mu) \wedge \bigwedge_{\psi\in N} \mathbf{FG}(\psi[M]_\nu)\right). \tag{6}$$

There are two differences between the expressions. First, in the latter expression we do not check that the negations of the formulas not included in the advice indeed do not hold. Actually, we can safely do this, since assuming negations of subformulas does not make (propositional) implication of φ any easier due to its negation normal form. Hence, this additional assumption does not increase the satisfiability of φ. However, not assuming them does not decrease it either, since the cases where the omitted formulas actually do hold are treated by other M, N-classes.

Second and even more importantly, since $\mathbf{GF}(\psi[N]_\mu) \in \mathbf{GF}(\mu LTL)$ and $\mathbf{FG}(\psi[M]_\nu) \in \mathbf{FG}(\nu LTL)$, the language $U_{M,N}$ is plain. This comes at the cost of a further over-approximation of $P_{M,N}$

$$P_{M,N} \subseteq U_{M,N}, \tag{7}$$

proven in Section 5.6 together with tightness of the over-approximation:

$$U_{M,N} \cap \mathcal{L}(\varphi[M]_\nu)^s \subseteq \mathcal{L}(\varphi)^s. \tag{8}$$

Intuitively, Equation (7) captures the soundness of using the advice in the simplification of the subformulas to plain ones, and Equation (8) claims that, given an advice that is indeed valid on the word, it is sound to use it when proving the global formula.

Table 1. An Example of a Word w and the Induced Formulas for $\varphi = \mathbf{F}(a \wedge \mathbf{G}b)$ with $M = \emptyset$ after Reading Its Prefixes

i	1	2	3	4	5	6	
$w[i-1]$	$\emptyset$	$\{a,b\}$	$\emptyset$	$\{a,b\}$	$\{b\}$	$\{b\}$	$\cdots$
$\mathrm{aft}(\varphi, w_{0i})$	φ	$\varphi \vee \mathbf{G}b$	φ	$\varphi \vee \mathbf{G}b$	$\varphi \vee \mathbf{G}b$	$\varphi \vee \mathbf{G}b$	$\cdots$
$\mathrm{aft}(\varphi, w_{0i})[M]_\nu$	$\mathbf{ff}$	$\mathbf{G}b$	$\mathbf{ff}$	$\mathbf{G}b$	$\mathbf{G}b$	$\mathbf{G}b$	$\cdots$
$w_i \models \mathrm{aft}(\varphi, w_{0i})[M]_\nu$	$\times$	$\times$	$\times$	$\checkmark$	$\checkmark$	$\checkmark$	$\cdots$

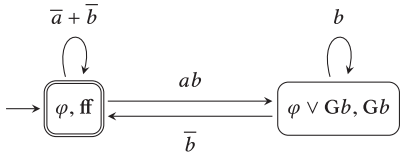

Fig. 4. DCA for $V_{M,N}$ for $\varphi = \mathbf{F}(a \wedge \mathbf{G}b)$ and $M = \emptyset$.

Combining Equation (4) with Equations (7) and (8), we obtain the Master Theorem for stable words:

$$\mathcal{L}(\varphi)^s = \bigcup_{\substack{M \subseteq \mu(\varphi) \\ N \subseteq \nu(\varphi)}} U_{M,N} \cap \mathcal{L}(\varphi[M]_\nu)^s. \tag{9}$$

5.2.5 Utilizing Advice on Unstable Words. The issue (ii) requires us to obtain an identity like Equation (9), but for $\mathcal{L}(\varphi)$ instead of $\mathcal{L}(\varphi)^s$. Let us first see why simply replacing L^s with L does not work:

Example 18. Consider the formula $\mathbf{F}a$ (It belongs to the plain fragments, but it illustrates the point.) and the word $w = \emptyset\{a\}\emptyset^\omega$. We have $w \in P_{\emptyset,\emptyset} \subseteq U_{\emptyset,\emptyset}$. To show $w \in \mathcal{L}(\varphi)$, we would need to show $w \in \mathcal{L}(\varphi[\emptyset]_\nu) = \mathcal{L}(\mathbf{ff})$, which is obviously not true. The issue is that we only need a to hold once, not infinitely often.

Lifting the result to non-stable words requires monitoring the evolution of the formula to be satisfied, as done in the usual tableaux constructions. To satisfy φ, an arbitrary word w must have a suffix w_i that satisfies a certain "derivative" φ corresponding to the moment i: This formula is neither $\varphi[M]_\nu$ (as refuted in the preceding example), nor just the correspondingly evolved formula $\mathrm{aft}(\varphi, w_{0i})$ (since this is hard to check using our *plain* advice), but rather $\mathrm{aft}(\varphi, w_{0i})[M]_\nu$. Formally, we define:

$$V_{M,N} := \{w : \exists i \geq 0.\ w_i \models \mathrm{aft}(\varphi, w_{0i})[M]_\nu\}. \tag{10}$$

Using this definition, **we prove in Section 5.7 the decomposability, as seen in Equation (2).**

Example 19. The evolution of the monitored formula and the corresponding substitution are illustrated for $\varphi = \mathbf{F}(a \wedge \mathbf{G}b)$ and $w = (\emptyset\{a,b\})^2\{b\}^\omega$ with $M = \emptyset$ in Table 1.

The language $V_{M,N}$ is not in any of the fragments of Section 2.2.2 but, since $\mathrm{aft}(\varphi, w_{0i})[M]_\nu \in$ *vLTL*, it is easy to construct an automaton for it, thus reaching our goal. Observe that, actually, $V_{M,N}$ is independent of N. A symmetric version of the Master Theorem with $\mathrm{aft}(\varphi, w_{0i})[N]_\mu$ is also possible, but the automata construction is not so simple in that case. We comment on this further in Section 5.7.

Example 20. Figure 4 depicts the automaton for $V_{M,N}$ with $\varphi = \mathbf{F}(a \wedge \mathbf{G}b)$ of the previous example.

The rest of the section proves the above boldfaced statements, completing the argumentation. Section 5.7 then formally re-states and proves the Master Theorem.

5.3 Stability

Fix a formula φ. The set of subformulas of φ with the shape $\psi_1 \mathbf{U} \psi_2$ 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.

Definition 21 (μ-stability). Let φ be a formula, and let $\mu(\varphi)$ be the set of subformulas of φ with the shape $\psi_1 \mathbf{U} \psi_2$ or with the shape $\psi_1 \mathbf{M} \psi_2$. For every word w, the sets $\mathcal{GF}_w^\varphi$ and $\mathcal{F}_w^\varphi$ are defined as follows:

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

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

Example 22. 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 with the shape $\psi_1 \mathbf{R} \psi_2$ and $\psi_1 \mathbf{W} \psi_2$, i.e., with greatest fixed-point operators on the top, is denoted by $\nu(\varphi)$. This time, we are interested in whether these formulas hold everywhere or almost everywhere.

Definition 23 (ν-stability). Let φ be a formula, let w be a word, and let $\nu(\varphi)$ be the set of subformulas of φ with the shape $\psi_1 \mathbf{R} \psi_2$ or with the shape $\psi_1 \mathbf{W} \psi_2$. Then the sets $\mathcal{FG}_w^\varphi$ and $\mathcal{G}_w^\varphi$ are defined as follows:

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

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

Example 24. Let φ, w, and w' as in Example 22. We have $\nu(\varphi) = \{\mathbf{G}a\}$. The word w is ν-stable, but w' is not, because $\mathcal{FG}_{w'}^\varphi = \{\mathbf{G}a\} \supset \emptyset = \mathcal{G}_{w'}^\varphi$.

A word is *stable* with respect to φ if it is both μ-stable and ν-stable. The inclusions $\mathcal{GF}_w^\varphi \subseteq \mathcal{F}_w^\varphi$ and $\mathcal{FG}_w^\varphi \supseteq \mathcal{G}_w^\varphi$ hold for every formula φ and every word w. We prove that all but finitely many suffixes of a word are stable. If the suffix w_i is stable with respect to φ, then we call i a *stabilisation point* for w with respect to φ.

Lemma 25. *Let φ be a formula and let w be a word. Then there exists an index ℓ such that for every $k \geq 0$ the suffix $w_{\ell+k}$ is stable with respect to φ.*

Proof. It suffices to find 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. 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. In the rest of the proof, we omit the superscript φ from $\mathcal{GF}_w^\varphi$, $\mathcal{FG}_w^\varphi$, and so on. 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 26. 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.

5.4 The Formulas $\varphi[M]_\nu$ and $\varphi[N]_\mu$

In the overview section, we have introduced the language $P_{M,N}$, consisting of the words w such that $M = \mathcal{GF}_w^\varphi$ and $N = \mathcal{FG}_w^\varphi$. In this section, we define simpler formulas $\varphi[M]_\nu \in \nu LTL$ and $\varphi[N]_\mu \in \mu LTL$, and in Section 5.5 below, we show that they satisfy identity (5) of the overview.[7]

Let us first define the formula $\varphi[M]_\nu$. According to identity (5), it has to fulfil

$$P_{M,N} \cap \mathcal{L}(\varphi)^s = P_{M,N} \cap \mathcal{L}(\varphi[M]_\nu)^s\ ,$$

i.e., the stable words of $P_{M,N}$ must satisfy φ iff they satisfy $\varphi[M]_\nu$. Since $\mathcal{GF}_w^\varphi = M = \mathcal{F}_w^\varphi$ holds for every stable word $w \in P_{M,N}$, we can set our goal as follows: Define a formula $\varphi[M]_\nu$ such that:

$$\left(\mathcal{GF}_w^\varphi = M = \mathcal{F}_w^\varphi\right) \Rightarrow (w \models \varphi \iff w \models \varphi[M]_\nu). \tag{11}$$

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

- $\varphi = \mathbf{F}a \wedge \mathbf{G}b$ and $M = \{\mathbf{F}a\}$. Since $M = \mathcal{GF}_w^\varphi$, we have $\mathbf{F}a \in \mathcal{GF}_w^\varphi$, which implies in particular $w \models \mathbf{F}a$. So $w \models \mathbf{F}a \wedge \mathbf{G}b$ iff $w \models \mathbf{G}b$, and so we can set $\varphi[M]_\nu := \mathbf{tt} \wedge \mathbf{G}b$, i.e., we can obtain $\varphi[M]_\nu$ by substituting $\mathbf{tt}$ for $\mathbf{F}a$ in φ.
- $\varphi = \mathbf{F}a \wedge \mathbf{G}b$ and $M = \emptyset$. Since $M = \mathcal{F}_w^\varphi$, we have $\mathbf{F}a \notin \mathcal{F}_w^\varphi$, and so $w \not\models \mathbf{F}a$. In other words, $w \models \mathbf{F}a \wedge \mathbf{G}b$ iff $w \models \mathbf{ff}$, and so we can set $\varphi[M]_\nu := \mathbf{ff} \wedge \mathbf{G}b$.
- $\varphi = \mathbf{G}(b\ \mathbf{U}\ c)$ and $M = \{b\ \mathbf{U}\ c\}$. Since $M = \mathcal{GF}_w^\varphi$, we have $b\ \mathbf{U}\ c \in \mathcal{GF}_w^\varphi$, 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 $w \models \mathbf{G}(b\ \mathbf{U}\ c)$ iff $w \models \mathbf{G}(b\ \mathbf{W}\ c)$, which is a formula of νLTL, and so we define $\varphi[M]_\nu := \mathbf{G}(b\ \mathbf{W}\ c)$.

The formal definition of $\varphi[M]_\nu$ is as follows:

Definition 27. Let φ be a formula and let $M \subseteq \mu(\varphi)$ be a set of formulas. The formula $\varphi[M]_\nu$ is inductively defined as follows for the interesting cases $\mathbf{U}$ and $\mathbf{M}$:

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

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

In all other cases it is defined as a recursive descent:

- If $\varphi \in \{\mathbf{tt}, \mathbf{ff}\} \cup \{a, \neg a : a \in Ap\}$, then $\varphi[M]_\nu = \varphi$.
- If $\varphi = \psi_1\ op\ \psi_2$ with $op \in \{\wedge, \vee, \mathbf{R}, \mathbf{W}\}$, then $\varphi[M]_\nu = (\psi_1[M]_\nu)\ op\ (\psi_2[M]_\nu)$.
- If $\varphi = \mathbf{X}\psi$, then $\varphi[M]_\nu = \mathbf{X}(\psi[M]_\nu)$.

[7] Observe that $M \subseteq \mu(\varphi)$ but $\varphi[M]_\nu$ is a formula of νLTL. So the ν-subscript in the notation $\varphi[M]_\nu$ indicates the fragment of LTL the formula belongs to, and not the fragment containing the formulas of M.

Let us now define the dual formula $\varphi[N]_\mu \in \mu LTL$.

Definition 28. Let φ be a formula and let $N \subseteq \nu(\varphi)$ be a set of formulas. The formula $\varphi[N]_\mu$ is inductively defined as follows for the interesting cases $\mathbf{R}$ and $\mathbf{W}$:

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

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

In all other cases it is defined as a recursive descent:

- If $\varphi \in \{\mathbf{tt}, \mathbf{ff}\} \cup \{a, \neg a : a \in Ap\}$, then $\varphi[M]_\mu = \varphi$.
- If $\varphi = \psi_1\ op\ \psi_2$ with $op \in \{\wedge, \vee, \mathbf{U}, \mathbf{M}\}$, then $\varphi[M]_\mu = (\psi_1[M]_\mu)\ op\ (\psi_2[M]_\mu)$.
- If $\varphi = \mathbf{X}\psi$, then $\varphi[M]_\mu = \mathbf{X}(\psi[M]_\mu)$.

Definitions for $\mathbf{F}$ *and* $\mathbf{G}$. Observe that $\mathbf{G}\varphi = \varphi\,\mathbf{W}\,\mathbf{ff}$ and that $\varphi\,\mathbf{U}\,\mathbf{ff} \equiv \mathbf{ff}$. Dually, we have $\mathbf{F}\varphi = \mathbf{tt}\,\mathbf{U}\,\varphi$ and $\mathbf{tt}\,\mathbf{W}\,\varphi \equiv \mathbf{tt}$. Since $\mathbf{F}$ and $\mathbf{G}$ frequently appear in examples, we make use of this observation and overload in the scope of examples the definitions of $\varphi[M]_\nu$ and $\varphi[N]_\mu$ for the sake of simplicity:

$$(\mathbf{F}\varphi)[M]_\nu := \begin{cases} \mathbf{tt} & if\ \mathbf{F}\varphi \in M \\ \mathbf{ff} & otherwise. \end{cases} \qquad (\mathbf{G}\varphi)[N]_\mu := \begin{cases} \mathbf{tt} & if\ \mathbf{G}\varphi \in N \\ \mathbf{ff} & otherwise. \end{cases}$$

Example 29. Let $\varphi = ((a\,\mathbf{W}\,b) \wedge \mathbf{F}c) \vee a\,\mathbf{U}\,d$. We have:

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

5.5 Properties of $\varphi[M]_\nu$ and $\varphi[N]_\mu$

The following lemma states the fundamental properties of $\varphi[M]_\nu$ and $\varphi[N]_\mu$. In particular, it proves identity (5) of the overview section.

Lemma 30. *Let φ be a formula and let w be a word. Let $M \subseteq \mu(\varphi)$ be a set of formulas. We have:*

(1) If $\mathcal{F}_w^\varphi \subseteq M$ and $w \models \varphi$, then $w \models \varphi[M]_\nu$.
(2) If $M \subseteq \mathcal{GF}_w^\varphi$ and $w \models \varphi[M]_\nu$, then $w \models \varphi$.

In particular:

(3) If $\mathcal{F}_w^\varphi = M = \mathcal{GF}_w^\varphi$, then $w \models \varphi \iff w \models \varphi[M]_\nu$.

Let $N \subseteq \nu(\varphi)$ be a set of formulas. We have:

(4) If $\mathcal{FG}_w^\varphi \subseteq N$ and $w \models \varphi$, then $w \models \varphi[N]_\mu$.
(5) If $N \subseteq \mathcal{G}_w^\varphi$ and $w \models \varphi[N]_\mu$, then $w \models \varphi$.

In particular:

(6) If $\mathcal{FG}_w^\varphi = N = \mathcal{G}_w^\varphi$, then $w \models \varphi \iff w \models \varphi[N]_\mu$.

Proof. All parts are proved by a straightforward structural induction on φ. Here, we only present two cases of the induction for (1) and (2). The rest of the proof can be found in Appendix A.

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

$$\forall i.\, ((w_i \models \varphi) \Rightarrow (w_i \models \varphi[M]_\nu)).$$

We consider one representative of the "interesting" cases and one of the "straightforward" cases:

- 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 $\varphi \in M$. We prove $w_i \models (\psi_1 \,\mathbf{U}\, \psi_2)[M]_\nu$:

$$\begin{aligned}
& w_i \models \psi_1 \,\mathbf{U}\, \psi_2 \\
\Rightarrow\ & w_i \models \psi_1 \,\mathbf{W}\, \psi_2 \\
\Longleftrightarrow\ & \forall j.\, w_{i+j} \models \psi_1 \vee \exists k \leq j.\, w_{i+k} \models \psi_2 \\
\Rightarrow\ & \forall j.\, w_{i+j} \models \psi_1[M]_\nu \vee \exists k \leq j.\, w_{i+k} \models \psi_2[M]_\nu && \text{(induction hypothesis)} \\
\Longleftrightarrow\ & w_i \models (\psi_1[M]_\nu) \,\mathbf{W}\, (\psi_2[M]_\nu) \\
\Longleftrightarrow\ & w_i \models (\psi_1 \,\mathbf{U}\, \psi_2)[M]_\nu && (\varphi \in M, \text{Definition 27})
\end{aligned}$$

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

$$\begin{aligned}
& w_i \models \psi_1 \vee \psi_2 \\
\Longleftrightarrow\ & w_i \models \psi_1 \vee w_i \models \psi_2 \\
\Rightarrow\ & w_i \models \psi_1[M]_\nu \vee w_i \models \psi_2[M]_\nu && \text{(induction hypothesis)} \\
\Longleftrightarrow\ & w_i \models (\psi_1 \vee \psi_2)[M]_\nu && \text{(Definition 27)}
\end{aligned}$$

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

$$\forall i.\, ((w_i \models \varphi[M]_\nu) \Rightarrow (w_i \models \varphi)).$$

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

$$\begin{aligned}
& w_i \models (\psi_1 \,\mathbf{U}\, \psi_2)[M]_\nu \\
\Longleftrightarrow\ & w_i \models (\psi_1[M]_\nu) \,\mathbf{W}\, (\psi_2[M]_\nu) && (\varphi \in M, \text{Definition 27}) \\
\Longleftrightarrow\ & \forall j.\, w_{i+j} \models \psi_1[M]_\nu \vee \exists k \leq j.\, w_{i+k} \models \psi_2[M]_\nu \\
\Rightarrow\ & \forall j.\, w_{i+j} \models \psi_1 \vee \exists k \leq j.\, w_{i+k} \models \psi_2 && \text{(induction hypothesis)} \\
\Longleftrightarrow\ & w_i \models \psi_1 \,\mathbf{W}\, \psi_2 \\
\Longleftrightarrow\ & w_i \models \psi_1 \,\mathbf{U}\, \psi_2 && (w_i \models \mathbf{F}\psi_2)
\end{aligned}$$

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

$$\begin{aligned}
& w_i \models (\psi_1 \vee \psi_2)[M]_\nu \\
\Longleftrightarrow\ & w_i \models \psi_1[M]_\nu \vee (w_i \models \psi_2[M]_\nu && \text{(Definition 27)} \\
\Rightarrow\ & w_i \models \psi_1 \vee w_i \models \psi_2 && \text{(induction hypothesis)} \\
\Longleftrightarrow\ & w_i \models \psi_1 \vee \psi_2
\end{aligned}$$

□

Example 31. 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 sets for M: $\emptyset$, $\{\mathbf{F}a\}$, $\{\mathbf{F}(b \wedge \mathbf{G}c)\}$, and $\{\mathbf{F}a, \mathbf{F}(b \wedge \mathbf{G}c)\}$. For $M = \emptyset$, we get $\varphi[M]_\nu \equiv \mathbf{ff}$, indicating

that if neither a nor $b \wedge \mathrm{G}c$ hold infinitely often, then φ cannot hold. For the other three possibilities (a holds infinitely often, $b \wedge \mathrm{G}c$ holds infinitely often, or both) there are words satisfying φ, like a^ω, $\{b,c\}^\omega$, and $\{a,b,c\}^\omega$, respectively.

Lemma 30 can be interpreted in terms of oracles. For example, Lemma 30.2 states that if an oracle promises us that w satisfies every property of M infinitely often, then to prove that w satisfies φ, we only need to show that it satisfies $\varphi[M]_\nu$. So, we can interpret $\varphi[M]_\nu$ as "φ after promise M."

5.6 Approximating $P_{M,N}$ with $U_{M,N}$

We prove identities (7) and (8) of the overview section. We consider Equation (7) first, i.e., we show $P_{M,N} \subseteq U_{M,N}$. Recall that $P_{M,N}$ is the set of words w satisfying $M = \mathcal{GF}^\varphi_w$ and $N = \mathcal{FG}^\varphi_w$, and $U_{M,N}$ is the language of all words satisfying $\mathbf{GF}(\psi[N]_\mu)$ for every $\psi \in M$, and $\mathbf{FG}(\psi[M]_\nu)$ for every $\psi \in N$. So it suffices to prove:

Lemma 32. *Let φ be a formula and let w be a word. If $M = \mathcal{GF}^\varphi_w$ and $N = \mathcal{FG}^\varphi_w$, then:*

$$\begin{aligned} &\forall \psi \in M. \quad w \models \mathbf{GF}(\psi[N]_\mu) \\ &\forall \psi \in N. \quad w \models \mathbf{FG}(\psi[M]_\nu). \end{aligned}$$

Proof. Let $\psi \in M = \mathcal{GF}^\varphi_w$. We have $w \models \mathbf{GF}\psi$, and so $w_i \models \psi$ for infinitely many $i \geq 0$. Since $\mathcal{FG}^\varphi_{w_i} = \mathcal{FG}^\varphi_w = N$ for every $i \geq 0$, Lemma 30.4 can be applied to w_i, $\mathcal{FG}^\varphi_{w_i}$, and ψ. This yields $w_i \models \psi[N]_\mu$ for infinitely many $i \geq 0$ and thus $w \models \mathbf{GF}(\psi[N]_\mu)$.

Let $\psi \in N = \mathcal{FG}^\varphi_w$. 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 25 the index j can be chosen so it also satisfies $M = \mathcal{GF}^\varphi_w = \mathcal{F}^\varphi_{w_{j+k}} = \mathcal{GF}^\varphi_{w_{j+k}}$ for every $k \geq 0$. Lemma 30.1 can be applied to $\mathcal{F}^\varphi_{w_{j+k}}$, w_{j+k}, and ψ. This yields $w_{j+k} \models \psi[M]_\nu$ for every $k \geq 0$ and thus $w \models \mathbf{FG}(\psi[M]_\nu)$. □

Let us now prove Equation (8), i.e., $U_{M,N} \cap \mathcal{L}(\varphi[M]_\nu)^s \subseteq \mathcal{L}(\varphi)^s$. We first observe that, by Lemma 30.2, it suffices to show that every word $w \in U_{M,N}$ satisfies $M \subseteq \mathcal{GF}^\varphi_w$. This is done in Lemma 33 below, but let us interpret this result. The lemma states that if a word w satisfies $\mathbf{GF}(\psi[N]_\mu)$ for every $\psi \in M$ and $\mathbf{FG}(\psi[M]_\nu)$ for every $\psi \in N$, then it satisfies $\mathbf{GF}\psi$ for every $\psi \in M$ and $\mathbf{FG}\psi$ for every $\psi \in N$. Recall that $\psi[M]_\nu$ can be interpreted as "ψ after promise M." So, at first sight, the lemma looks wrong: To verify the promise $M \subseteq \mathcal{GF}^\varphi_w$, we are relying on the promise $N \subseteq \mathcal{FG}^\varphi_w$, and vice versa. However, there is no circularity in this rely/guarantee reasoning. Indeed, for the promise $M \subseteq \mathcal{GF}^\varphi_w$, we only rely on the promise $N \subseteq \mathcal{FG}^\varphi_w$ "for subformulas of M," and vice versa. Since the subformula order is well founded, we eventually reach formulas ψ such that $\psi[M]_\nu = \psi$ or $\psi[N]_\mu = \psi$.

Lemma 33. *Let φ be a formula and let w be a word. For every $M \subseteq \mu(\varphi)$ and $N \subseteq \nu(\varphi)$, if*

$$\begin{aligned} &\forall \psi \in M. \quad w \models \mathbf{GF}(\psi[N]_\mu) \\ &\forall \psi \in N. \quad w \models \mathbf{FG}(\psi[M]_\nu), \end{aligned}$$

then $M \subseteq \mathcal{GF}^\varphi_w$ and $N \subseteq \mathcal{FG}^\varphi_w$.

Proof. Let $M \subseteq \mu(\varphi)$ and $N \subseteq \nu(\varphi)$. Observe that $M \cap N = \emptyset$. Let $n := |M \cup N|$. Let $\psi_1, \ldots, \psi_n$ be an enumeration of $M \cup N$ compatible with the subformula order, i.e., if ψ_i is a subformula of ψ_j, then $i \leq j$. Finally, let $(M_0, N_0), (M_1, N_1), \ldots, (M_n, N_n)$ be the unique sequence of pairs satisfying:

- $(M_0, N_0) = (\emptyset, \emptyset)$ and $(M_n, N_n) = (M, N)$.
- For every $0 < i \leq n$, if $\psi_i \in M$, then $M_i \setminus M_{i-1} = \{\psi_i\}$ and $N_i = N_{i-1}$, and if $\psi_i \in N$, then $M_i = M_{i-1}$ and $N_i \setminus N_{i-1} = \{\psi_i\}$.

We prove $M_i \subseteq \mathcal{GF}^{\varphi}_{w}$ and $N_i \subseteq \mathcal{FG}^{\varphi}_{w}$ for every $0 \leq i \leq n$ by induction on i. For $i = 0$ the result follows immediately from $M_0 = \emptyset = N_0$. For $i > 0$, we consider two cases:

- $\psi_i \in N$, i.e., $M_i = M_{i-1}$ and $N_i \setminus N_{i-1} = \{\psi_i\}$.
  By induction hypothesis and $M_i = M_{i-1}$, we have $M_i \subseteq \mathcal{GF}^{\varphi}_{w}$ and $N_{i-1} \subseteq \mathcal{FG}^{\varphi}_{w}$. We prove $\psi_i \in \mathcal{FG}^{\varphi}_{w}$, i.e., $w \models \mathbf{FG}\psi_i$, in three steps.
  — Claim 1: $\psi_i[M]_\nu = \psi_i[M_i]_\nu$.
    By the definition of $\cdot[\cdot]_\nu$, $\psi_i[M]_\nu$ is completely determined by the μ-subformulas of ψ_i that belong to M. By the definition of the sequence $(M_0, N_0), \ldots, (M_n, N_n)$, a μ-subformula of ψ_i belongs to M if and only if it belongs to M_i, and we are done.
  — Claim 2: $M_i \subseteq \mathcal{GF}^{\varphi}_{w_k}$ for every $k \geq 0$.
    Follows immediately from $M_i \subseteq \mathcal{GF}^{\varphi}_{w}$.
  — Proof of $w \models \mathbf{FG}\psi_i$.
    By the assumption of the lemma, we have $w \models \mathbf{FG}(\psi_i[M]_\nu)$, and so, by Claim 1, $w \models \mathbf{FG}(\psi_i[M_i]_\nu)$. So there exists an index j such that $w_{j+k} \models \psi_i[M_i]_\nu$ for every $k \geq 0$. By Claim 2, we further have $M_i \subseteq \mathcal{GF}^{\varphi}_{w_{j+k}}$ for every $j, k \geq 0$. So, we can apply Lemma 30.2 to M_i, w_{j+k}, and ψ_i, which yields $w_{j+k} \models \psi_i$ for every $k \geq 0$. So $w \models \mathbf{FG}\psi_i$.
- $\psi_i \in M$, i.e., $M_i \setminus M_{i-1} = \{\psi_i\}$ and $N_i = N_{i-1}$.
  By induction hypothesis, we have in this case $M_{i-1} \subseteq \mathcal{GF}^{\varphi}_{w}$ and $N_i \subseteq \mathcal{FG}^{\varphi}_{w}$. We prove $\psi_i \in \mathcal{GF}^{\varphi}_{w}$, i.e., $w \models \mathbf{GF}\psi_i$ in three steps.
  — Claim 1: $\psi_i[N]_\mu = \psi_i[N_i]_\mu$.
    The claim is proved as in the previous case.
  — Claim 2: There exists $j \geq 0$ such that $N_i \subseteq \mathcal{G}^{\varphi}_{w_k}$ for every $k \geq j$.
    Follows immediately from $N_i \subseteq \mathcal{FG}^{\varphi}_{w}$.
  — Proof of $w \models \mathbf{GF}\psi_i$.
    By the assumption of the lemma, we have $w \models \mathbf{GF}(\psi_i[N]_\mu)$. Let j be the index of Claim 2. By Claim 1, we have $w \models \mathbf{GF}(\psi_i[N_i]_\mu)$, and so there exist infinitely many $k \geq j$ such that $w_k \models \psi_i[N_i]_\mu$. By Claim 2, we further have $N_i \subseteq \mathcal{G}^{\varphi}_{w_k}$. So, we can apply Lemma 30.5 to N_i, w_k, and ψ_i, which yields $w_k \models \psi_i$ for infinitely many $k \geq j$. So $w \models \mathbf{GF}\psi_i$. □

Example 34. Let $\varphi = \mathbf{F}(a \wedge \mathbf{G}(b \vee \mathbf{F}c))$, $M = \{\varphi\}$, and $N = \{\mathbf{G}(b \vee \mathbf{F}c)\}$.

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

$$w \models \mathbf{GF}\left(\varphi[N]_\mu\right) = \mathbf{GF}(\mathbf{F}a \wedge \mathbf{tt}) \equiv \mathbf{GF}a.$$

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

$$w \models \mathbf{FG}\left((\mathbf{G}(b \vee \mathbf{F}c))[M]_\nu\right) = \mathbf{FG}(\mathbf{G}(b \vee \mathbf{ff})) \equiv \mathbf{FG}b.$$

Applying Lemma 32 to this, we then obtain that $w \models \mathbf{GF}a \wedge \mathbf{FG}b$ implies $\varphi \in \mathcal{GF}^{\varphi}_{w}$ and $\mathbf{G}(b \vee \mathbf{F}c) \in \mathcal{FG}^{\varphi}_{w}$.

5.7 The Master Theorem: Logical Characterisation of LTL

Putting together Lemmas 30, 32, and 33, we arrive at our main contribution: a decomposition of the language of an LTL formula into a Boolean combination of plain languages.

Theorem 35 (Master Theorem). *Let φ be a formula and let w be a word. Then $w \models \varphi$ if and only if there exist $M \subseteq \mu(\varphi)$ and $N \subseteq \nu(\varphi)$ satisfying:*

$$\begin{array}{ll} (1) & \exists i.\, w_i \models \mathrm{aft}(\varphi, w_{0i})[M]_\nu, \\ (2) & \forall \psi \in M. w \models \mathbf{GF}(\psi[N]_\mu), \\ (3) & \forall \psi \in N. w \models \mathbf{FG}(\psi[M]_\nu). \end{array}$$

With respect to the overview, observe that $U_{M,N}$, defined in Equation (6), is the language of words satisfying (2) and (3), and $V_{M,N}$, defined in Equation (10), is the language of words satisfying (1). If w is stable, then we can take $i = 0$, and condition (1) becomes $w \models \varphi[M]_\nu$.

Observe that $\mathrm{aft}(\varphi, w_{0i})[M]_\nu$, $\mathbf{GF}(\psi[N]_\mu)$, and $\mathbf{FG}(\psi[M]_\nu)$ are formulas of the LTL fragments νLTL, $\mathbf{GF}(\mu LTL)$, and $\mathbf{FG}(\nu LTL)$, respectively. For these fragments, we can build automata using Proposition 13. So, given a class of automata effectively closed under union and intersection, we can construct automata for all of LTL if we know how to do it for plain formulas.

Proof of the Master Theorem. ($\Rightarrow$) Assume $w \models \varphi$, and set $M := \mathcal{GF}_w^\varphi$ and $N := \mathcal{FG}_w^\varphi$. Properties (2) and (3) follow from Lemma 32. For property (1), let i be an index such that $\mathcal{F}_{w_i}^\varphi = \mathcal{GF}_{w_i}^\varphi = \mathcal{GF}_w^\varphi$; this index exists by Lemma 25. By Lemma 9, we have $w_i \models \mathrm{aft}(\varphi, w_{0i})$, and by Lemma 30.1 $w_i \models \mathrm{aft}(\varphi, w_{0i})[M]_\nu$.

($\Leftarrow$) Assume that properties (1–3) hold for sets $M \subseteq \mu(\varphi)$, $N \subseteq \nu(\varphi)$ and an index i. By Lemma 33, we have $M \subseteq \mathcal{GF}_w^\varphi$, and so $M \subseteq \mathcal{GF}_{w_i}^\varphi$. From $w_i \models \mathrm{aft}(\varphi, w_{0i})[M]_\nu$, we obtain $w_i \models \mathrm{aft}(\varphi, w_{0i})$ with Lemma 30.2. Finally, Lemma 9 yields then $w \models \varphi$. □

Example 36. Let $\varphi = \mathbf{F}(a \wedge \mathbf{G}(b \vee \mathbf{F}c))$ as in Example 34, and let $\varphi' = d\,\mathbf{U}\,(e \wedge \varphi)$. For $M = \{\varphi, \varphi'\}$, $N = \{\mathbf{G}(b \vee \mathbf{F}c)\}$, and $i = 0$ the Master Theorem yields that $w \models \varphi'$ is implied by

$$\begin{array}{ll} (1) & w \models (d\,\mathbf{U}\,(e \wedge \varphi))[M]_\nu = d\,\mathbf{W}\,(e \wedge \varphi[M]_\nu) = d\,\mathbf{W}\,(e \wedge \mathbf{tt}) \equiv d\,\mathbf{W}\,e, \\ (2) & w \models \mathbf{GF}(\varphi[N]_\mu) = \mathbf{GF}(\mathbf{F}(a \wedge \mathbf{tt})) \equiv \mathbf{GF}a, \\ & w \models \mathbf{GF}(\varphi'[N]_\mu) = \mathbf{GF}(d\,\mathbf{U}\,(e \wedge \mathbf{F}(a \wedge \mathbf{tt}))) \equiv \mathbf{GF}(e \wedge \mathbf{F}a), \text{ and} \\ (3) & w \models \mathbf{FG}((\mathbf{G}(b \vee \mathbf{F}c))[M]_\nu) = \mathbf{FG}(\mathbf{G}(b \vee \mathbf{ff})) \equiv \mathbf{FG}b. \end{array}$$

For $M' = \{\varphi\}$, $N' = \{\mathbf{G}(b \vee \mathbf{F}c)\}$, and $i' = 0$, condition (1) cannot be fulfilled, since:

$$w \not\models (d\,\mathbf{U}\,(e \wedge \varphi))[M']_\nu = \mathbf{ff}.$$

Logically this is sound, because $w \models \mathbf{ff}$ implies everything. This just shows that the set of words satisfying these conditions is empty. Let us now choose $i' = 1$, and let us assume that the first letter of the word is $\{e\}$, i.e., $w[0] = \{e\}$. Then the Master Theorem says that $w \models \varphi'$ is implied by

$$\begin{array}{ll} (1) & w \models (\mathrm{aft}(d\,\mathbf{U}\,(e \wedge \varphi), \{e\})[M']_\nu = ((\mathbf{tt} \wedge (\varphi \vee \ldots)) \vee (\mathbf{ff} \wedge \varphi'))\,[M']_\nu \equiv \mathbf{tt}, \\ (2) & w \models \mathbf{GF}(\varphi[N']_\mu) = \mathbf{GF}(\mathbf{F}(a \wedge \mathbf{tt})) \equiv \mathbf{GF}a, \\ (3) & w \models \mathbf{FG}((\mathbf{G}(b \vee \mathbf{F}c))[M']_\nu) = \mathbf{FG}(\mathbf{G}(b \vee \mathbf{ff})) \equiv \mathbf{FG}b. \end{array}$$

Thus, the Master Theorem is offering us yet another way to prove $w \models \varphi$. One that is even simpler than the first one.

The formulation of the Master Theorem is almost symmetric. Only one "impurity" breaks this symmetry: the advice function $\cdot[M]_\nu$ in condition (1). Could we also use $\cdot[N]_\mu$ at that location? Yes, the theorem remains true after substituting $\cdot[N]_\mu$ for $\cdot[M]_\nu$. So why do we choose $\cdot[M]_\nu$? The application of $\cdot[M]_\nu$ yields a formula of the fragment νLTL. The languages corresponding to this fragment are characterised by (finite) *bad prefixes*. Whenever a word does *not* satisfy $\psi[M]_\nu$, we can learn this from a finite prefix. Let us now consider $\cdot[N]_\mu$. The languages corresponding to the fragment $\cdot[N]_\mu$ are characterised by (finite) *good prefixes* and if a word satisfies $\psi[N]_\mu$, then we know this after reading a finite prefix, but in general there is no finite prefix for proving that $\psi[N]_\mu$ is *not* satisfied.

For this reason, if we choose $\psi[M]_\nu$, then we can 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 37 shows how to do this efficiently. However, this argument does not work for $\psi[N]_\mu$ 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 difficult for a deterministic automaton.

6 DRAS FOR ARBITRARY LTL FORMULAS

We use the Master Theorem (Theorem 35) to extend the translation for μLTL, νLTL, $\mathbf{GF}(\mu LTL)$, and $\mathbf{FG}(\nu LTL)$ to arbitrary LTL formulas. Since 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.

Let φ be a formula of length n. We construct a DRA for $\mathcal{L}(\varphi)$ with $2^{2^{O(n)}}$ states and at most 2^n Rabin pairs. For a fixed guess M and N, we first construct DBAs and DCAs checking (1–3), to which we refer by $\mathcal{C}^1_{\varphi,M}$, $\mathcal{B}^2_{M,N}$, and $\mathcal{C}^3_{M,N}$. These automata are then intersected resulting in a DRA $\mathcal{R}_{\varphi,M,N}$ with $2^{2^{O(n)}}$ states and one single Rabin pair. Finally, we construct a DRA $\mathcal{A}_{\mathrm{DRA}}(\varphi)$ for $\mathcal{L}(\varphi)$ by taking the union of all $\mathcal{R}_{\varphi,M,N}$. Proposition 13 shows how to build $\mathcal{B}^2_{M,N}$ and $\mathcal{C}^3_{M,N}$, but cannot be immediately applied to $\mathcal{C}^1_{\varphi,M}$. We delay the definition of $\mathcal{A}_{\mathrm{DRA}}(\varphi)$ 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 $w_i \models \mathrm{aft}(\varphi, w_{0i})[M]_\nu$ for some suffix w_i. It is intuitive that if we have an automaton identifying such an index i, then we immediately obtain that (1) of Theorem 35 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 following lemma, which allows us to postpone checking $w_i \models \mathrm{aft}(\varphi, w_{0i})[M]_\nu$ by an arbitrary number of steps and thus the automaton cannot miss the right moment.

Lemma 37. *Let φ be a formula, let w be a word. If $w \models \varphi[M]_\nu$, then $w_i \models \mathrm{aft}(\varphi, w_{0i})[M]_\nu$ for every index i.*

Proof. Assume $w \models \varphi[M]_\nu$. It suffices to prove $w_1 \models \mathrm{aft}(\varphi, w_{01})[M]_\nu$ 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 $w \models a[M]_\nu = a$, we have $a \in w_{01}$. So $\mathrm{aft}(a, w_{01})[M]_\nu = \mathbf{tt}[M]_\nu = \mathbf{tt}$, and thus $w_1 \models \mathrm{aft}(\varphi, w_{01})[M]_\nu$.
- Case $\varphi = \psi_1 \,\mathbf{U}\, \psi_2$: Since $w \models \varphi[M]_\nu$, we have $\varphi[M]_\nu \neq \mathbf{ff}$, and so $\varphi \in M$. We derive:

$$
\begin{aligned}
& w \models \varphi[M]_\nu && \\
\iff\ & w \models (\psi_1[M]_\nu)\ \mathbf{W}\ (\psi_2[M]_\nu) && (\varphi \in M, \text{Definition 27}) \\
\iff\ & w \models \psi_2[M]_\nu \vee (\psi_1[M]_\nu \wedge \mathbf{X}((\psi_1[M]_\nu)\ \mathbf{W}\ (\psi_2[M]_\nu))) && (\text{LTL semantics}) \\
\iff\ & w \models \psi_2[M]_\nu \vee (\psi_1[M]_\nu \wedge \mathbf{X}(\varphi[M]_\nu)) && (\varphi \in M, \text{Definition 27}) \\
\Rightarrow\ & w_1 \models \mathrm{aft}(\psi_2, w_{01})[M]_\nu \vee (\mathrm{aft}(\psi_1, w_{01})[M]_\nu \wedge \varphi[M]_\nu) && (\text{induction hypothesis}) \\
\iff\ & w_1 \models (\mathrm{aft}(\psi_2, w_{01}) \vee (\mathrm{aft}(\psi_1, w_{01}) \wedge \varphi))[M]_\nu && (\text{Definition 27}) \\
\iff\ & w_1 \models \mathrm{aft}(\varphi, w_{01})[M]_\nu && (\text{Definition 7})
\end{aligned}
$$

□

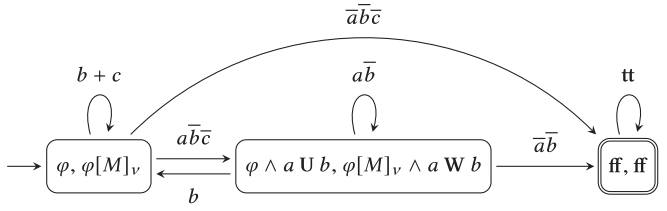

Fig. 5. DCA $C^1_{\varphi,M}$ for $\varphi = \mathbf{G}(a \,\mathbf{U}\, b \vee c)$ and $M = \{a \,\mathbf{U}\, b\}$.

Loosely speaking, $C^1_{\varphi,M}$—the automaton responsible for checking (1)—starts by checking $w \models \varphi[M]_\nu$. For this it uses the construction of Proposition 13 on the *vLTL* formula $\varphi[M]_\nu$. Intuitively, if that automaton rejects, then it rejects "after finite time." If the formula becomes equivalent to **ff** after, say, i steps, then $w \not\models \varphi[M]_\nu$, and $C^1_{\varphi,M}$ proceeds to check $w \models \mathrm{aft}(\varphi, w_{0i})[M]_\nu$. To perform this reset, $C^1_{\varphi,M}$ needs to know $\mathrm{aft}(\varphi, w_{0i})$, and so it also maintains $\mathrm{aft}(\varphi, w_{0i})$ in its state. In other words, after j steps $C^1_{\varphi,M}$ is in state:

$$\langle \mathrm{aft}(\varphi, w_{0j}), \mathrm{aft}((\mathrm{aft}\,(\varphi, w_{0i})\,[M]_\nu), w_{ij})\rangle,$$

where $i \leq j$ is the number of steps after which $C^1_{\varphi,M}$ had to reset for the last time. If the second component of the state becomes **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 $w \models \varphi_j[M]_\nu$ succeeds.

To use $\cdot[\cdot]_\nu$ for this, we first need to show that we can lift the operation to our states, which are equivalence classes and not formulas:

Lemma 38. *Let φ and ψ be two formulas. If $\varphi \sim \psi$, then $\varphi[M]_\nu \sim \psi[M]_\nu$.*

Proof. Notice that $\cdot[\cdot]_\nu$ satisfies the precondition of Lemma 6 and applying it yields the statement we wanted to prove. □

Proposition 39. *Let φ be a formula and let M be a set of formulas. Then the following DCA over the alphabet 2^{Ap} recognises exactly the words w such that $\exists i.\, w_i \models \mathrm{aft}(\varphi, w_{0i})[M]_\nu$ holds:*

$$C^1_{\varphi,M} = (Q, \delta, \langle [\varphi]_\sim, [\varphi[M]_\nu]_\sim\rangle, \mathrm{Fin}(\alpha))\,,$$

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

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

Example 40. Let $\varphi = \mathbf{G}(a \,\mathbf{U}\, b \vee c)$. Observe that φ is not in any of our "simple" fragments and that it recognises neither a safety nor a co-safety language. Further, let $M = \{a \,\mathbf{U}\, b\}$ and $M' = \emptyset$ be two guesses. Hence, $\varphi[M]_\nu = \mathbf{G}(a \,\mathbf{W}\, b \vee c)$ and $\varphi[M']_\nu = \mathbf{G}(\mathbf{ff} \vee c)$. The DCA $C^1_{\varphi,M}$ from Proposition 39 for φ and M is shown in Figure 5 and the DCA $C^1_{\varphi,M'}$ for M' is shown in Figure 6.

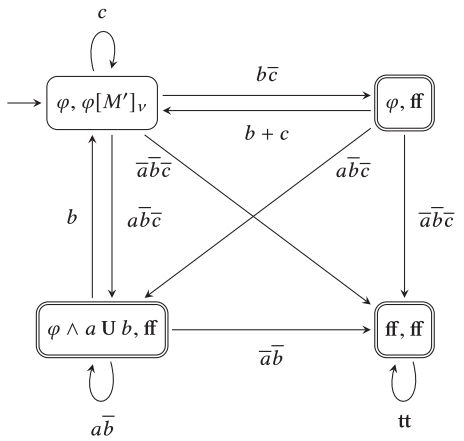

Fig. 6. DCA $C^1_{\varphi,M'}$ for $\varphi = \mathbf{G}(a\,\mathbf{U}\,b \vee c)$ and $M' = \emptyset$.

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 $M = \mathcal{GF}^{\varphi}_{w} \supset M'$. On the one hand the word w is accepted by $C^1_{\varphi,M}$, since the run alternates between the two non-rejecting states. On the other hand, we have $w_1 \models \mathrm{aft}(\varphi, w_{01})[M]_\nu \equiv \mathbf{G}(a\,\mathbf{W}\,b \vee c)$. Moving to $C^1_{\varphi,M'}$, we see that w is rejected, since the run infinitely often leaves the initial state $\langle \varphi, \varphi[M']_\nu \rangle$ and visits the rejecting state $\langle \varphi \wedge a\,\mathbf{U}\,b, \mathbf{ff} \rangle$. Consequently, we have $w_i \not\models \mathrm{aft}(\varphi, w_{0i})[M']_\nu$ for all i.

For $w' = (\overline{a}\overline{b}c)^\omega$, we have $M \supset \mathcal{GF}^{\varphi}_{w'} = M'$. The word w' is *not* rejected by $C^1_{\varphi,M}$ and we have for example $w' \models \varphi[M]_\nu = \mathbf{G}(a\,\mathbf{W}\,b \vee c)$. However, the guess M cannot be proven and thus the assumption of M being a correct guess is not satisfied. Further, w' is accepted by $C^1_{\varphi,M'}$, which loops in the initial state, and we also have $w' \models \varphi[M']_\nu \equiv \mathbf{G}c$.

Proof of Proposition 39. ($\Rightarrow$) Let w be a word such that $w_i \models \mathrm{aft}(\varphi, w_{0i})[M]_\nu$ for some i. We need to show that the run r induced by w is an accepting run on $C^1_{\varphi,M}$. Then by Lemma 37, we know that all following $j > i$ have the same property ($w_j \models \mathrm{aft}(\varphi, w_{0j})[M]_\nu$). Let r be the run on $C^1_{\varphi,M}$ corresponding to w. We prove that there is at most one $k > i$, such that $r[k] \in \alpha$ and thus $w \in \mathcal{L}(C^1_{\varphi,M})$. Assume k_1 is the first index larger than i where $r[k_1] \in \alpha$. If there is no such element, then we are immediately done. Thus, $r[k_1 + 1] = \langle \mathrm{aft}(\varphi, w_{0(k_1+2)}), (\mathrm{aft}(\varphi, w_{0(k_1+2)}))[M]_\nu \rangle$. Since $k_1 + 2 > i$, we have $w_{k_1+2} \models \mathrm{aft}(\varphi, w_{0(k_1+2)})[M]_\nu$, thus due to Lemma 12.2, we have that the second component never reaches **ff** again.

($\Leftarrow$) Assume w is accepted by $C^1_{\varphi,M}$. Then the corresponding run r visits the set of states α only finitely often. Let i be the 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 $\mathrm{aft}(\varphi, w_{0(i+1)})[M]_\nu$ at $r[i+1]$. Since this component never sees $[\mathbf{ff}]_\sim$ again after i, we can apply Lemma 12.2 to obtain $w_{i+1} \models \mathrm{aft}(\varphi, w_{0(i+1)})[M]_\nu$, which concludes the proof. □

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 13 and Proposition 39:

Theorem 41. *Let φ be a formula. We define for each $M \subseteq \mu(\varphi)$ and each $N \subseteq \nu(\varphi)$ the following DBAs and DCAs:*

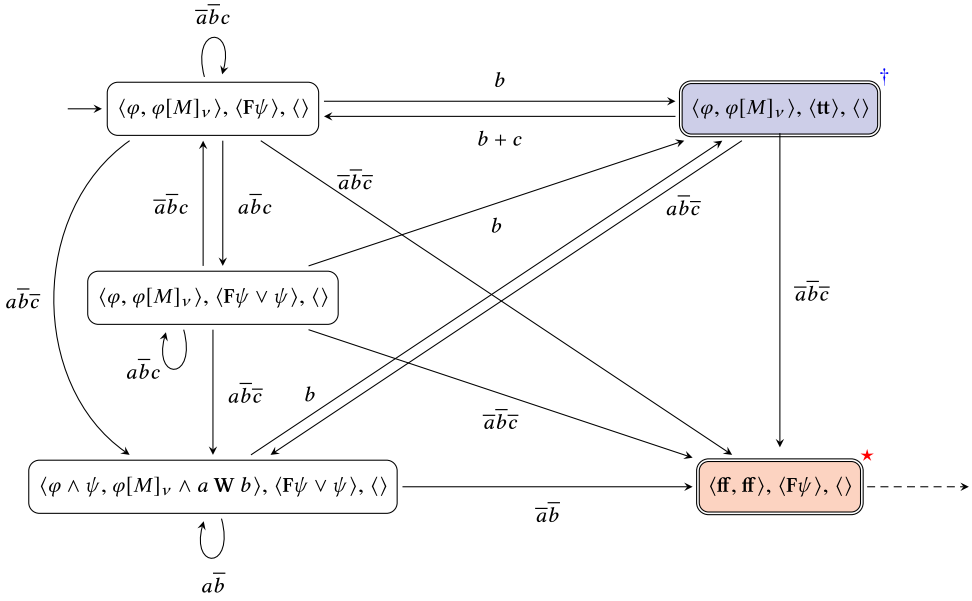

Fig. 7. DRA $\mathcal{R}_{\varphi,M,N}$ for $\varphi = \mathrm{G}(\psi \vee c)$, $\psi = a \,\mathrm{U}\, b$, $M = \{\psi\}$, and $N = \emptyset$. The DRA $\mathcal{R}_{\varphi,M,N}$ has only one Rabin pair: $\mathrm{Fin}(\star) \wedge \mathrm{Inf}(\dagger)$. Some rejecting states have been omitted and the edges leading to them are indicated by a densely dashed arrow.

- $C^1_{\varphi,M}$ *is the DCA from Proposition 39.*
- $\mathcal{B}^2_{M,N} = \bigcap_{\psi \in M} \mathcal{B}^{\psi[N]_\mu}_{\mathrm{GF}\mu}$ *is the intersection of DBAs from Proposition 13.*
- $C^3_{M,N} = \bigcap_{\psi \in N} C^{\psi[M]_\nu}_{\mathrm{FG}\nu}$ *is the intersection of DCAs from Proposition 13.*

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

$$\mathcal{R}_{\varphi,M,N} = C^1_{\varphi,M} \cap \mathcal{B}^2_{M,N} \cap C^3_{M,N}.$$

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

$$\mathcal{A}_{\mathrm{DRA}}(\varphi) = \bigcup_{\substack{M \subseteq \mu(\varphi) \\ N \subseteq \nu(\varphi)}} \mathcal{R}_{\varphi,M,N}.$$

Example 42. Let $\varphi = \mathbf{G}(a \,\mathbf{U}\, b \vee c)$ be the formula from Example 40. We build the DRA $\mathcal{R}_{\varphi,M,N}$ (Theorem 41) for $M = \{a \,\mathbf{U}\, b\}$ and $N = \emptyset$ as the intersection of the DCA $C^1_{\varphi,M}$ (Figure 5) and the DBA $\mathcal{B}^{(a\mathrm{U}b)}_{\mathrm{GF}\mu}$ (Proposition 13). The DRA is shown in Figure 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 $M = \mathcal{GF}^\varphi_w$. The word w is accepted, since the run eventually alternates between the states in the lower-left and upper-right corners of Figure 7. For $w' = (\overline{a}\overline{b}c)^\omega$, we have $M \neq \mathcal{GF}^\varphi_{w'}$. The word w' is rejected, since the run stays in the initial state and never visits an accepting state. Observe that in this case we still have $w' \models \mathbf{G}(a \,\mathbf{U}\, b \vee c)$ and the word is accepted by another DRA component. More precisely, w' is accepted by the DRA $\mathcal{R}_{\varphi,M',N'}$ with $M' = \emptyset$ and $N' = \emptyset$.

The reader might have noticed that there are distinct states, e.g., $\mathrm{F}\psi$ and $\mathrm{F}\psi \vee \psi$, that are language-equivalent and there exists a simple syntactic procedure to show these formulas are

equivalent. This can be integrated into the presented framework and only needs technical changes as described in Reference [69].

Proof of Theorem 41. Let φ be a formula, let $\mathcal{A}_{\mathrm{DRA}}(\varphi)$ be the corresponding automaton, and let w be a word.

($\Rightarrow$) Assume w is in $\mathcal{L}(\varphi)$. By Theorem 35 there exist M and N such that (1–3) hold. We show that $w \in \mathcal{L}(\mathcal{A}_{\mathrm{DRA}}(\varphi))$ by proving that the run r for w on $\mathcal{R}_{\varphi,M,N}$ is accepting. This run is accepting if there exist accepting runs for w on $C^1_{\varphi,M}$, $\mathcal{B}^2_{M,N}$, and $C^3_{M,N}$. To show this, we simply need to apply Proposition 39 to (1) and Proposition 13 to (2–3) and we are done.

($\Leftarrow$) Assume w is accepted by $\mathcal{A}_{\mathrm{DRA}}(\varphi)$. Thus, there exists an accepting run r for w on $\mathcal{R}_{\varphi,M,N}$ for some $M \subseteq \mu(\varphi)$ and $N \subseteq \nu(\varphi)$. Thus, $C^1_{\varphi,M}$, $\mathcal{B}^2_{M,N}$, and $C^3_{M,N}$ accept w and applying Proposition 39 and Proposition 13 yields (1–3) for M and N. Hence, the right-hand side of Theorem 35 is fulfilled and we follow that w is in $\mathcal{L}(\varphi)$. □

6.1 Complexity Analysis

Let φ be a formula of length n. The size of the automata from Proposition 13 crucially depends on the properties of propositional equivalence, because it determines how $\mathrm{Reach}(\varphi)_{/\sim}$ is partitioned into equivalence classes. In Lemma 10, we already showed that 2^{2^n} is an upper bound of the cardinality of $\mathrm{Reach}(\varphi)_{/\sim}$. The automata $\mathcal{B}^{\varphi}_{\mu}$, $\mathcal{B}^{\varphi}_{\nu}$, C^{φ}_{μ}, and C^{φ}_{ν} have at most 2^{2^n} states and, by the same argument, $\mathcal{B}^{\varphi}_{\mathrm{GF}\mu}$ and $C^{\varphi}_{\mathrm{FG}\nu}$ have at most $2^{2^{n+1}}$ states. This is already asymptotically optimal, as the smallest automata are still doubly exponential in the worst-case [1, 43].

Interestingly, not only $\mathrm{Reach}(\varphi)_{/\sim}$ but also $\bigcup_{\psi \in \mathrm{Reach}(\varphi)} \mathrm{Reach}(\psi[M]_\nu)_{/\sim}$ has at most 2^{2^n} elements. Indeed, this set is a subset of all (positive) Boolean functions over at most n variables, since $\cdot[\cdot]_\nu$ only replaces existing temporal subformulas by other subformulas of the same (or smaller) size or the constant $\mathbf{ff}$. It follows that the automaton $C_{\varphi,M}$ has at most $(2^{2^n})^2 = 2^{2^{n+1}}$ states.

Let us now bound the number of states of the whole automaton. We have seen before that $C^1_{\varphi,M}$ has at most $2^{2^{n+1}}$ states. By Proposition 13, $\mathcal{L}(\mathbf{GF}(\psi[N]_\mu)$ 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 recognised by a DBA with $k \cdot \prod_{j=1}^{k} s_j$ states. Since $|M| \leq n$, the intersection of the DBAs $\mathcal{B}^2_{M,N}$ for the formulas $\mathbf{GF}(\psi[N]_\mu)$ 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 13 $\mathcal{L}(\mathbf{FG}(\psi[M]_\mu)$ 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 DCA with $\prod_{j=1}^{k} s_j$ states. Since $|N| \leq n$, the intersection of the DCAs $C^3_{M,N}$ for the formulas $\mathbf{FG}(\psi[M]_\nu)$ 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 $\mathcal{R}_{\varphi,M,N}$ 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 $\mathcal{A}_{\mathrm{DRA}}(\varphi)$ 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.

Remark 43. While the achieved result matches the theoretical worst-case complexity, it is easy to refine the construction to achieve a practically scalable translation. For instance, one can determine based on syntactic criteria if a particular pair of sets (M, N) is redundant. The corresponding redundant $\mathcal{R}_{\varphi,M,N}$ can then be removed from $\mathcal{A}_{\mathrm{DRA}}(\varphi)$ and thus decreasing its size. Additionally, standard-style optimisations such as dropping redundant Rabin pairs, see, e.g., Reference [27], or using specialised intersection constructions for $\mathcal{B}^2_{M,N}$ can be used. For an extensive list of optimisations and an in-depth discussion of their effect, we refer the interested reader to the Reference [69].

7 LDBA CONSTRUCTION

7.1 LDBAs for Arbitrary LTL Formulas

We present a translation of LTL into limit-deterministic Büchi automaton (LDBA), a special class of nondeterministic Büchi automata. 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 and is deterministic. If a run enters the accepting component via a so-called "jump", it cannot leave this component anymore.

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 35, 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 *vLTL*. Using this initial idea, we rephrase Theorem 35 and break the symmetry of the Master Theorem 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 44 (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 $M \subseteq \mu(\varphi)$, $N \subseteq \nu(\varphi)$, and $i \geq 0$ satisfying:*

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

Proof. Clearly, the existence of an index i satisfying (1′–3′) implies that conditions (1–3) of Theorem 35 hold. For the other direction, assume conditions (1–3) hold. By Lemma 37 the index i stemming from condition (1) can be chosen arbitrarily large. Furthermore, since $w \models \bigwedge_{\psi \in M} \mathbf{FG}(\psi[M]_\nu)$, we can choose i so it also satisfies $w_i \models \bigwedge_{\psi \in M} \mathbf{G}(\psi[M]_\nu)$. □

The LDBA construction tracks in the initial component $\mathrm{aft}(\varphi, w_{0i})$, i.e., after reading a finite word w_{0i} the initial component is in state $[\mathrm{aft}(\varphi, w_{0i})]_\sim$. The states $[\mathrm{aft}(\varphi, w_{0i})]_\sim$ are then connected to the accepting component by jumps that guess sets M and N and the stabilisation point i. The jump leads to the corresponding initial state of the intersection automaton $(\mathcal{B}_{M,N})$ of three DBAs, which are in charge of checking (1′), (2′), and (3′), and we refer to these automata by $\mathcal{B}^1_M$, $\mathcal{B}^2_{M,N}$, and $\mathcal{B}^3_{M,N}$, respectively.

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

$$\chi[M]_\nu \qquad \bigwedge_{\psi \in M} \mathbf{GF}(\psi[N]_\mu) \qquad \bigwedge_{\psi \in N} \mathbf{G}(\psi[M]_\nu).$$

These formulas belong to *vLTL*, $\mathbf{GF}(\mu LTL)$, and *vLTL*, respectively, and so we can obtain DBAs for them following the recipes of Proposition 13. Note that for all formulas χ and χ' the DBAs $\mathcal{B}_{\chi,M,N}$ and $\mathcal{B}_{\chi',M,N}$ have the same transition relation on common states and the same accepting

states. We take advantage of this and combine all $\mathcal{B}_{\chi,M,N}$ into a single structure that we call $\mathcal{B}_{M,N}$. Formally, we construct a *semi-deterministic*[8] Büchi automaton that is a Büchi automaton with a deterministic transition relation, but with a set of initial states. The set of initial states $Q_{0,M,N}$ then contains for each $[\chi]_\sim \in \mathrm{Reach}(\varphi)_{/\sim}$ the initial state $([\chi[M]_\nu]_\sim, q_0', q_0'')$. Summarising, we obtain:

Theorem 45. *Let φ be a formula. We define for each $M \subseteq \mu(\varphi)$ and each $N \subseteq \nu(\varphi)$ the following:*

- $\mathcal{B}^1_M = \bigcup_{[\chi]_\sim \in \mathrm{Reach}(\varphi)_{/\sim}} \mathcal{B}_\nu^{\chi[M]_\nu}$ *is the (semi-deterministic) union of DBAs from Proposition 13.*
- $\mathcal{B}^2_{M,N} = \bigcap_{\psi \in M} \mathcal{B}^{\psi[N]_\mu}_{\mathbf{GF}\mu}$ *is the intersection of DBAs from Proposition 13.*
- $\mathcal{B}^3_{M,N}$ *is the DBA for* $\bigwedge_{\psi \in N} \mathbf{G}(\psi[M]_\nu)$ *from Proposition 13.*

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

$$\mathcal{B}_{M,N} = (Q_{M,N}, \delta_{M,N}, Q_{0,M,N}, \mathrm{Inf}(\alpha_{M,N})).$$

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

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

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

- $Q = \mathrm{Reach}(\varphi)_{/\sim} \cup \bigcup\{Q_{M,N} : M \subseteq \mu(\varphi), N \subseteq \nu(\varphi)\}$. *That is, a state is either an equivalence class $[\psi]_\sim$ or a product state $([\psi]_\sim, q', q'') \in \mathcal{B}_{M,N}$ from one of the accepting components.*
- $\delta = \mathrm{aft}_\sim \cup \delta_\curvearrowright \cup \bigcup\{\delta_{M,N} : M \subseteq \mu(\varphi), N \subseteq \nu(\varphi)\}$. *That is, a transition is either within the initial component ($\mathrm{aft}_\sim$), within an accepting component ($\delta_{M,N}$), or it is a jump ($\delta_\curvearrowright$) from the initial to an accepting component. Let $\delta_\curvearrowright$ be defined for each state $[\psi]_\sim \in \mathrm{Reach}(\varphi)_{/\sim}$ and each letter $\sigma \in 2^{Ap}$ as:*

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

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

Note that the automata $\mathcal{B}^1_M$ and $\mathcal{B}^3_{M,N}$ 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 46. Let $\varphi = \mathbf{F}a \vee \mathbf{FG}b$. The LDBA $\mathcal{A}_{\mathsf{LDBA}}(\varphi)$ constructed from Theorem 45 is depicted in Figure 8. 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., $M = \{\mathbf{F}a\}, N = \{\mathbf{G}b\}$ or $M' = \{\mathbf{FG}b\}, N' = \{\}$, and some non-accepting states. We mark these omissions by dashed outgoing transitions.

This example illustrates already a potential optimisation: $\mathcal{B}^1_M$ and $\mathcal{B}^3_{M,N}$ can be combined into a single component, since both are in charge of accepting formulas from *vLTL*.

[8]The term appears in Reference [81] 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.

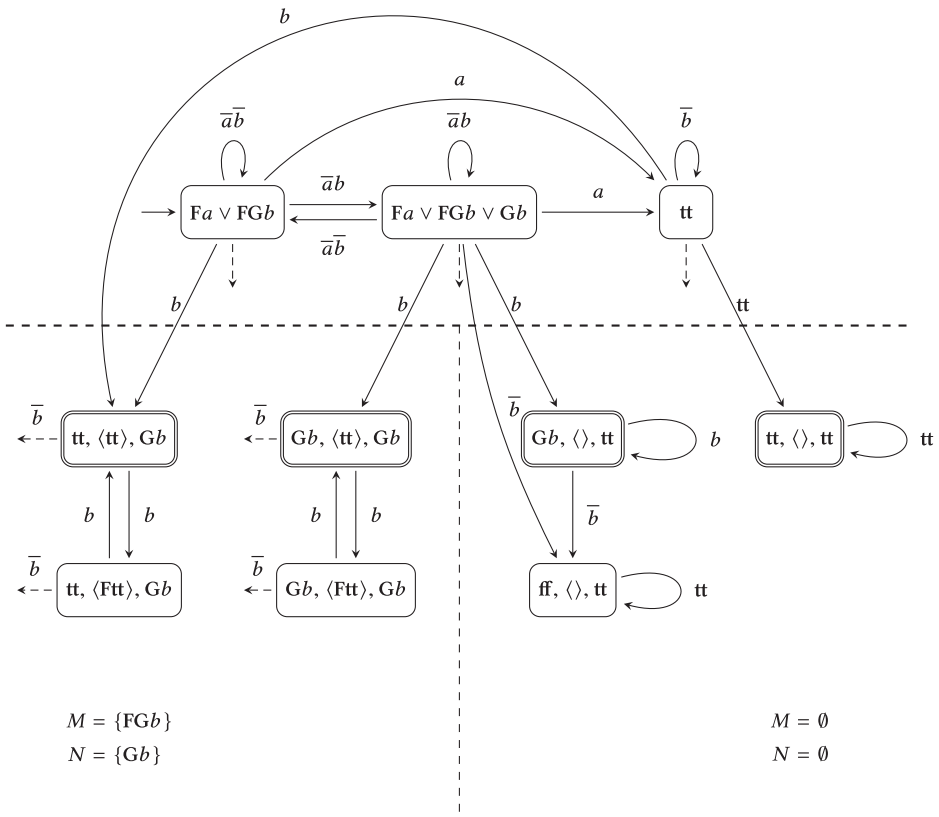

Fig. 8. $\mathcal{A}_{\mathsf{LDBA}}(\varphi)$ for $\varphi = \mathrm{F}a \vee \mathrm{FG}b$. Omitted states and transitions are indicated by dashed arrows.

Proof of Theorem 45. Let φ be a formula, let $\mathcal{A}_{\mathsf{LDBA}}(\varphi)$ be the corresponding automaton, and let w be a word. Further, we denote by $q_{\chi',M,N} \in Q_{0,M,N}$ the initial state of $\mathcal{B}_{M,N}$ with $[\chi']_\sim = [\chi[M]_\nu]_\sim$ in the first component for every reachable state $[\chi]_\sim \in \mathrm{Reach}(\varphi)_{/\sim}$ and for all sets of formulas $M \subseteq \mu(\varphi)$ and $N \subseteq \nu(\varphi)$. We let $\mathcal{L}(q_{\chi',M,N})$ denote the language accepted from $q_{\chi',M,N}$ on the automaton $\mathcal{B}_{M,N}$. Using Proposition 13, we can characterise the language $\mathcal{L}(q_{\chi',M,N})$ in terms of LTL:

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

($\Rightarrow$) Assume w is in $\mathcal{L}(\varphi)$. By Corollary 44 there exist i, M, and N 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 $\mathrm{aft}_\sim([\varphi]_\sim, w_{0i}) = [\mathrm{aft}(\varphi, w_{0i})]_\sim = [\chi]_\sim$ for some χ. The run then branches off to the accepting component $\mathcal{B}_{M,N}$. Notice that (1′–3′) exactly matches the right-hand side of our characterisation of $\mathcal{L}(q_{\chi',M,N})$ and thus, we have $w_i \in \mathcal{L}(q_{\chi',M,N})$. Thus, there exists an accepting run r' for w_i on $\mathcal{B}_{M,N}$ starting in $q_{\chi',M,N}$. Observe that we can reach from $[\chi]_\sim$ the same states as $q_{\chi',M,N}$, since $\delta_\frown([\chi]_\sim, w[i]) \cap Q_{M,N} = \delta_{M,N}(q_{\chi',M,N}, w[i])$. Thus, r simply follows the accepting run r' from this point on and we are done.

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

$$r'[j] = \begin{cases} q_{\chi',M,N} & \text{if } j = 0 \\ r[i+j] & \text{otherwise.} \end{cases}$$

Thus, $w \in \mathcal{L}(q_{\chi',M,N})$ and using our characterisation of this language, we end up with (1′–3′) of Corollary 44. Consequently, $w \models \varphi$ and we are done. □

7.2 Complexity Analysis

Let φ be a formula of length n. Since $\mathcal{A}_{\mathrm{LDBA}}(\varphi)$ is a union and intersection of several components, the number of states can be bounded by the size of the components:

$$|\mathrm{Reach}(\varphi)_{/\sim}| + \sum_{\substack{M \subseteq \mu(\varphi) \\ N \subseteq \nu(\varphi)}} (|Q^1_M| \cdot |Q^2_{M,N}| \cdot |Q^3_{M,N}|).$$

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

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

Recall that the lower bound for the blowup of a translation from LTL to LDBAs is double exponential [72].

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 these LDBAs usable for quantitative (and not only qualitative) probabilistic model checking. We give further details for this in Section 9.

8 NBA CONSTRUCTION

We proceed as in the DRA case: We first define constructions for LTL fragments by adapting the function aft to the nondeterministic setting (Section 8.1) and then assemble them into a translation for arbitrary LTL formulas (Section 8.2) using Corollary 44.

8.1 NBAs for μLTL, νLTL, GF(μLTL), and FG(νLTL)

Let φ be a formula in one of the fragments. Loosely speaking, the states of the deterministic automaton for φ are subformulas of φ, and the transition function is determined by aft: There is a transition $\psi \xrightarrow{\sigma} \psi'$ iff $\mathrm{aft}(\psi, \sigma) = \psi'$. To get nondeterministic automata, we put $\mathrm{aft}(\psi, \sigma)$ into disjunctive normal form (DNF) and proceed as follows: If the DNF of $\mathrm{aft}(\psi, \sigma)$ is $\psi_1 \vee \cdots \vee \psi_n$, then the automaton has transitions $\psi \xrightarrow{\sigma} \psi_1, \ldots, \psi \xrightarrow{\sigma} \psi_n$. We introduce a function $\mathrm{aft}_\vee$ such that,

loosely speaking, $\mathrm{aft}_\vee(\psi,\sigma) = \{\psi_1,\dots,\psi_n\}$. So the NBAs we construct have a transition $\psi \xrightarrow{\sigma} \psi'$ if $\psi' \in \mathrm{aft}_\vee(\psi,\sigma)$.

Transforming the above into a correct and formally defined construction requires some care. Section 8.1.1 introduces some notations, Section 8.1.2 defines the function $\mathrm{aft}_\vee$, and Section 8.1.3 defines the NBAs for our four fragments of LTL.

8.1.1 Disjunctive Normal Form. We represent formulas in DNF as sets of sets of literals. For example, $(a \wedge b) \vee (\neg a \wedge c)$ is represented as $\{\{a,b\},\{\neg a,c\}\}$. We use $X, Y, \dots$ to denote finite sets of sets and use the following notation:

- $X \otimes Y := \{A \cup B : A \in X, B \in Y\}$
- $\bigotimes\{X_1, X_2, \dots X_n\} := \{A_1 \cup A_2 \cup \dots \cup A_n : A_1 \in X_1, A_2 \in X_2, \dots A_n \in X_n\}$.
- $\mathit{min}(X) := \{A \in X : \forall B \in X.\ B \not\subset A\}$
- $X \cup_{\mathit{min}} Y := \mathit{min}(X \cup Y)$, $X \otimes_{\mathit{min}} Y := \mathit{min}(X \otimes Y)$, and $\bigotimes_{\mathit{min}} \mathcal{X} := \mathit{min}(\bigotimes \mathcal{X})$.

The following identities are useful:

$$\bigotimes\nolimits_{\mathit{min}} \emptyset = \{\emptyset\} \qquad \bigotimes\nolimits_{\mathit{min}} \{X\} = \mathit{min}(X) \qquad \bigotimes\nolimits_{\mathit{min}} (\mathcal{X} \cup \mathcal{Y}) = (\bigotimes\nolimits_{\mathit{min}} \mathcal{X}) \otimes_{\mathit{min}} (\bigotimes\nolimits_{\mathit{min}} \mathcal{Y}).$$

We define the minimal DNF $\mathrm{dnf}(\varphi)$ of an LTL formula φ and show that it satisfies the following property: The assignments that propositionally satisfy φ are the sets of literals of $\mathrm{dnf}(\varphi)$.

Proposition 47 (Minimal Disjunctive Normal Form). *Let φ be a formula. We then recursively define the minimal disjunctive normal form $\mathrm{dnf}(\varphi)$ as follows:*

$$\begin{array}{llll} \mathrm{dnf}(\mathbf{tt}) & = \{\emptyset\} & \mathrm{dnf}(\varphi \wedge \psi) & = \mathrm{dnf}(\varphi) \otimes_{\mathit{min}} \mathrm{dnf}(\psi) \\ \mathrm{dnf}(\mathbf{ff}) & = \emptyset & \mathrm{dnf}(\varphi \vee \psi) & = \mathrm{dnf}(\varphi) \cup_{\mathit{min}} \mathrm{dnf}(\psi) \\ \mathrm{dnf}(a) & = \{\{a\}\} & \mathrm{dnf}(\mathbf{X}\varphi) & = \{\{\mathbf{X}\varphi\}\} \\ \mathrm{dnf}(\neg a) & = \{\{\neg a\}\} & \mathrm{dnf}(\varphi \mathbin{op} \psi) & = \{\{\varphi \mathbin{op} \psi\}\} \quad \textit{for } op \in \{\mathbf{U}, \mathbf{M}, \mathbf{R}, \mathbf{W}\}. \end{array}$$

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

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

Proof sketch. This is proven by a straightforward structural induction on φ. □

Example 48. Let $\varphi = a \vee \overbrace{((a \wedge \mathbf{X}b) \vee \mathbf{F}(b \vee c))}^{\psi}$. We then compute $\mathrm{dnf}(a) = \{\{a\}\}$, $\mathrm{dnf}(a \wedge \mathbf{X}b) = \{\{a, \mathbf{X}b\}\}$, and $\mathrm{dnf}(\mathbf{F}(b \vee c)) = \{\{\mathbf{F}(b \vee c)\}\}$. Then $\mathrm{dnf}(\psi) = \{\{a, \mathbf{X}b\}, \{\mathbf{F}(b \vee c)\}\}$ and finally:

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

8.1.2 The Disjunctive "after"-Function. We define $\mathrm{aft}_\vee$, the disjunctive version of aft.

Definition 49 (Disjunctive aft*).* Let Ψ be a set of formulas and let σ be a letter. We define

$$\mathrm{aft}_\vee(\Psi,\sigma) := \bigotimes_{\mathit{min}} \{\mathrm{dnf}(\mathrm{aft}(\psi,\sigma)) : \psi \in \Psi\}.$$

We extend the definition to words as follows:

$$\begin{aligned} \mathrm{aft}_\vee(\Psi, \epsilon) &:= \{\Psi\} \\ \mathrm{aft}_\vee(\Psi, w\sigma) &:= \bigcup \{\mathrm{aft}_\vee(\Psi', \sigma) : \Psi' \in \mathrm{aft}_\vee(\Psi, w)\}. \end{aligned}$$

Finally, we define the reachability set of a formula φ by

$$\mathrm{aft}_\vee(\varphi, w) := \bigcup\{\mathrm{aft}_\vee(\Psi, w) : \Psi \in \mathrm{dnf}(\varphi)\}$$
$$\mathrm{Reach}_\vee(\varphi) := \bigcup\{\mathrm{aft}_\vee(\varphi, w) : w \in (2^{Ap})^*\}.$$

Notice that we use $\bigcup$ in the definition of $\mathrm{aft}_\vee(\Psi, w\sigma)$ and do not use $\bigcup_{min}$. Moreover, notice that $\mathrm{aft}_\vee(\varphi, \epsilon) = \mathrm{dnf}(\varphi)$.

Example 50. Let $\varphi = \mathbf{X}a \vee \mathbf{X}(a \wedge b)$. We compute $\mathrm{aft}_\vee(\varphi, \{\})$ and $\mathrm{aft}_\vee(\varphi, \{\}\{b\})$:

$$\begin{aligned}\mathrm{aft}_\vee(\varphi, \{\}) &= \bigcup\{\mathrm{aft}_\vee(\Psi, \{\}) : \Psi \in \mathrm{dnf}(\varphi)\}\\ &= \bigcup\{\mathrm{aft}_\vee(\{\mathbf{X}a\}, \{\}), \mathrm{aft}_\vee(\{\mathbf{X}(a \wedge b)\}, \{\})\}\\ &= \bigcup\{\mathrm{dnf}(a), \mathrm{dnf}(a \wedge b)\}\\ &= \bigcup\{\{\{a\}\}, \{\{a, b\}\}\} = \{\{a\}, \{a, b\}\}.\end{aligned}$$

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

The reachability set of φ is $\mathrm{Reach}_\vee(\varphi) = \{\{\mathbf{X}a\}, \{\mathbf{X}(a \wedge b)\}, \{a, b\}, \{a\}, \{\}\}$.

The following two lemmas prove elementary properties of $\mathrm{aft}_\vee$. The proofs are given in Appendix A.

Lemma 51. *Let φ be a formula, let w be a finite word, and let $\mathcal{I}$ be a propositional assignment. Then:*

$$\mathcal{I} \models_p \mathrm{aft}(\varphi, w) \iff \exists \Psi \subseteq \mathcal{I}.\ \Psi \in \mathrm{aft}_\vee(\varphi, w).$$

Lemma 52. *Let φ be a formula, let w be a finite word, let $M \subseteq \mu(\varphi)$, and let $\mathcal{I} \subseteq \mathrm{sf}(\varphi)$. Then:*

(1) $\mathrm{aft}_\vee(\varphi, w) \subseteq 2^{\mathrm{sf}(\varphi)}$,
(2) $\emptyset \in \mathrm{aft}_\vee(\varphi, w) \iff \mathrm{aft}(\varphi, w) \sim \mathbf{tt}$,
(3) $\mathrm{aft}_\vee(\varphi, w) = \emptyset \iff \mathrm{aft}(\varphi, w) \sim \mathbf{ff}$,
(4) $\mathcal{L}(\mathrm{aft}(\varphi, w)) = \bigcup\{\bigcap_{\psi \in \Psi} \mathcal{L}(\psi) : \Psi \in \mathrm{aft}_\vee(\varphi, w)\}$,
(5) Let $\Psi[M]_\nu := \{\psi[M]_\nu : \psi \in \Psi\}$. *Then:*

$$\mathcal{I} \models_p \mathrm{aft}(\varphi, w)[M]_\nu \iff \exists \Psi.\ \Psi[M]_\nu \subseteq \mathcal{I} \wedge \Psi \in \mathrm{aft}_\vee(\varphi, w).$$

8.1.3 Automata Constructions. We construct NBAs for the LTL fragments of Section 2.2.2.

Proposition 53. *Let $\varphi \in \mu LTL$.*

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

$$\mathcal{B}^\varphi_\mu = (\mathrm{Reach}_\vee(\varphi), \mathrm{aft}_\vee, \mathrm{dnf}(\varphi), \mathrm{Inf}(\{\emptyset\})).$$

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

$$\mathcal{B}^{\varphi}_{\mathbf{GF}\mu} = \left(\mathrm{Reach}_{\vee}(\mathbf{F}\varphi), \mathrm{aft}^{\mathbf{F}\varphi}_{\vee}, \{\{\mathbf{F}\varphi\}\}, \mathrm{Inf}(\{\emptyset\})\right)$$

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

Let $\varphi \in vLTL$.

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

$$\mathcal{B}^{\varphi}_{\nu} = (\mathrm{Reach}_{\vee}(\varphi), \mathrm{aft}_{\vee}, \mathrm{dnf}(\varphi), \mathrm{Inf}(\mathrm{Reach}_{\vee}(\varphi))).$$

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

$$\mathcal{B}^{\varphi}_{\mathbf{FG}\nu} = \left(\mathrm{Reach}_{\vee}(\mathbf{G}\varphi) \cup \{\{\mathbf{FG}\varphi\}\}, \mathrm{aft}^{\mathbf{G}\varphi}_{\vee}, \{\{\mathbf{FG}\varphi\}\}, \mathrm{Inf}(\mathrm{Reach}_{\vee}(\mathbf{G}\varphi))\right)$$

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

Example 54. Let $\varphi = a \wedge \mathbf{X}(b \vee \mathbf{F}c)$, the formula for which a DBA was given in Example 14. The NBA $\mathcal{B}^{\varphi}_{\mathbf{GF}\mu}$ for φ is shown in Figure 9. Compared to the DBA of Example 14, the NBA has a simpler structure, although in this case the same number of states.

Example 55. Let $\varphi_n = \bigvee_{i=1}^{n} \mathbf{F}(a_i \wedge \mathbf{XF}b_i)$ a family of LTL formulas. In Example 54, we showed that the NBA has a simpler structure, but the number of states stayed the same. In this example, we look at the family φ_n, which yields a polynomial-sized nondeterministic automaton for φ_n (Figure 10), while we obtain an exponential-sized deterministic automaton (Figure 11).

8.2 NBAs for Arbitrary LTL Formulas

Equipped with NBA constructions for LTL fragments, we go back to the construction in Theorem 45 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 56. *Let φ be a formula. We define for each $M \subseteq \mu(\varphi)$ and each $N \subseteq \nu(\varphi)$ the following:*

- $\mathcal{B}^1_M = \bigcup_{\Psi \in \mathrm{Reach}_{\vee}(\varphi)} \mathcal{B}^{\wedge \Psi[M]_{\nu}}_{\nu}$ *is the union of NBAs from Proposition 53.*
- $\mathcal{B}^2_{M,N} = \bigcap_{\psi \in M} \mathcal{B}^{\psi[N]_{\mu}}_{\mathbf{GF}\mu}$ *is the intersection of NBAs from Proposition 53.*
- $\mathcal{B}^3_{M,N}$ *is the NBA for $\bigwedge_{\psi \in N} \mathbf{G}(\psi[M]_{\nu})$ from Proposition 53.*

Let $\mathcal{B}_{M,N}$ be the intersection of the NBAs $\mathcal{B}^1_M$, $\mathcal{B}^2_{M,N}$, and $\mathcal{B}^3_{M,N}$:

$$\mathcal{B}_{M,N} = (Q_{M,N}, \delta_{M,N}, Q_{0,M,N}, \mathrm{Inf}(\alpha_{M,N})).$$

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

$$\mathcal{A}_{\mathrm{NBA}}(\varphi) = (Q, \delta, \mathrm{dnf}(\varphi), \mathrm{Inf}(\alpha)).$$

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

- $Q = \mathrm{Reach}_{\vee}(\varphi) \cup \bigcup\{Q_{M,N} : M \subseteq \mu(\varphi), N \subseteq \nu(\varphi)\}$. *That is, a state is either a clause Ψ or a tuple of clauses $(\Psi, \Psi', \Psi'') \in \mathcal{B}_{M,N}$ from one of the accepting components.*
- $\delta = \mathrm{aft}_{\vee} \cup \delta_{\curvearrowright} \cup \bigcup\{\delta_{M,N} : M \subseteq \mu(\varphi), N \subseteq \nu(\varphi)\}$. *That is, a transition is either within the initial component ($\mathrm{aft}_{\vee}$), within an accepting component ($\delta_{M,N}$), or it is a jump ($\delta_{\curvearrowright}$) from*

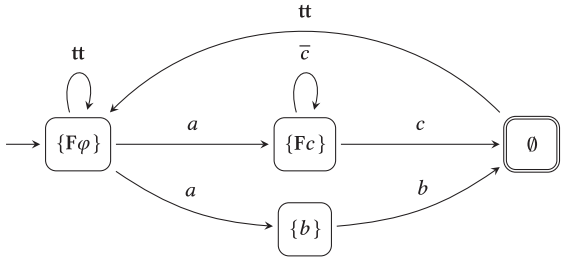

Fig. 9. NBA $\mathcal{B}^{\varphi}_{\mathrm{GF}\mu}$ for $\varphi = a \wedge \mathbf{X}(b \vee \mathbf{F}c)$. Notice the different self-loops on $\{\mathbf{F}\varphi\}$ and $\{\mathbf{F}c\}$. In the first state the automaton guesses nondeterministically when to track φ, and in the second state the construction already resolves the nondeterminism.

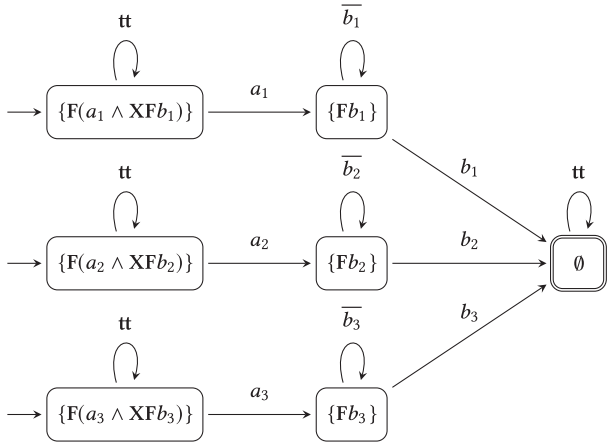

Fig. 10. NBA $\mathcal{B}^{\varphi}_{\mu}$ for $\varphi_3 = \bigvee_{i=1}^{3} \psi_i$ with $\psi_i = \mathbf{F}(a_i \wedge \mathbf{X}\mathbf{F}b_i)$. Notice that this NBA repeats for each ψ_i the same structure and has in total $2n + 1 = 7$ states.

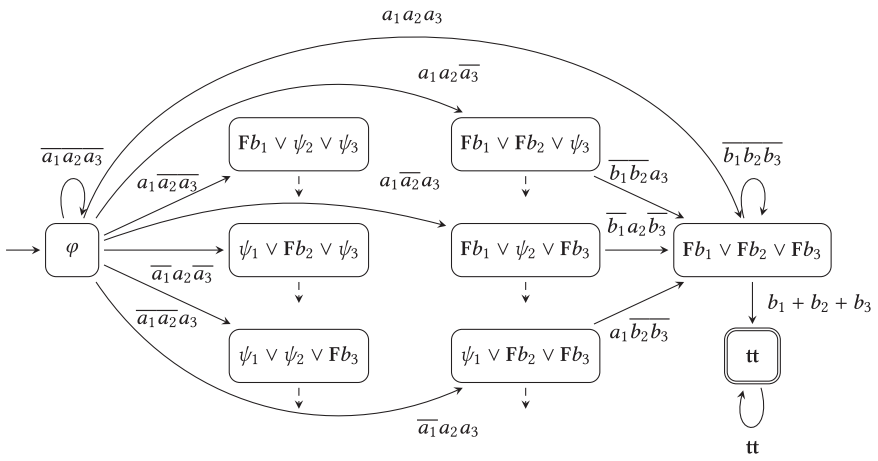

Fig. 11. DBA $\mathcal{B}^{\varphi}_{\mu}$ for $\varphi_3 = \bigvee_{i=1}^{3} \psi_i$ with $\psi_i = \mathbf{F}(a_i \wedge \mathbf{X}\mathbf{F}b_i)$ from Proposition 13 with simplified state labels. Omitted transitions are indicated by dashed arrows. This automaton has $2^n + 1 = 9$ states.

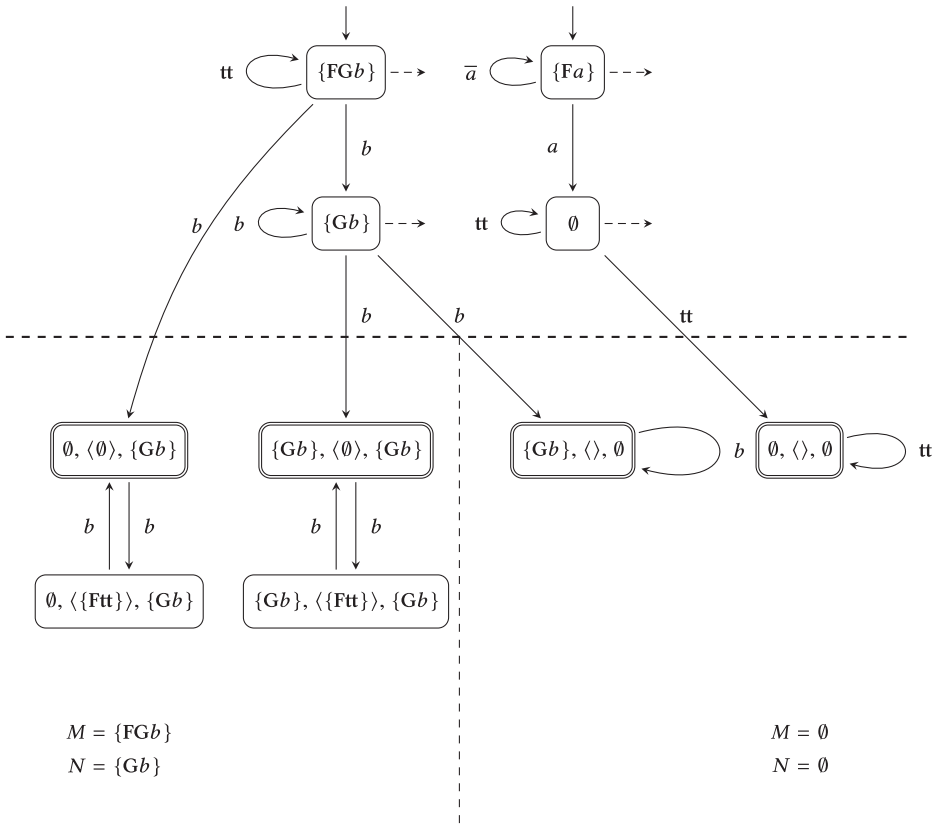

Fig. 12. $\mathcal{A}_{\mathsf{NBA}}(\varphi)$ for $\varphi = \mathbf{F}a \vee \mathbf{FG}b$. Omitted states and transitions are indicated by dashed arrows.

initial to accepting component. Let $\delta_{\curvearrowright}$ be defined for each state $\Psi \in \mathrm{Reach}_{\vee}(\varphi)$ and each letter $\sigma \in 2^{Ap}$ as:

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

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

Note that the automata $\mathcal{B}^1_M$ and $\mathcal{B}^3_{M,N}$ 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. Before we move on to the proof of the theorem, we illustrate the construction in the following example:

Example 57. Let us revisit the formula $\varphi = \mathbf{F}a \vee \mathbf{FG}b$ previously used in Example 46. The NBA $\mathcal{A}_{\mathsf{NBA}}(\varphi)$ constructed from Theorem 56 is depicted in Figure 12. Compared to the LDBA sketched in Figure 8 the transition structure is considerably simpler.

Proof of Theorem 56. 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 $Q_{\Psi',M,N} \subseteq Q_{0,M,N}$ the initial states of $\mathcal{B}_{M,N}$ with $\Psi' = \Psi[M]_{\nu}$ in the first component for every reachable state $\Psi \in \mathrm{Reach}_{\vee}(\varphi)$ and for all sets of formulas $M \subseteq \mu(\varphi)$ and $N \subseteq \nu(\varphi)$. We denote by $\mathcal{L}(Q_{\Psi',M,N})$ the language accepted from the states $Q_{\Psi',M,N}$ on the automaton $\mathcal{B}_{M,N}$ and using Proposition 53, we can characterise the language $\mathcal{L}(Q_{\Psi',M,N})$

in terms of LTL:

$$\forall v.\ v \in \mathcal{L}(Q_{\Psi',M,N}) \iff \forall \psi \in \Psi'.\ v \models \psi \\ \wedge \forall \psi \in M.\ v \models \mathbf{GF}(\psi[N]_\mu) \\ \wedge \forall \psi \in N.\ v \models \mathbf{G}(\psi[M]_\nu)$$

($\Rightarrow$) Assume $w \models \varphi$. By Corollary 44 there exists i, M, and N 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 $\mathcal{I} := \{\psi \in \mathrm{sf}(\mathrm{aft}(\varphi, w_{0i})[M]_\nu) : w_i \models \psi\}$ be a propositional assignment. From (1′) and Lemma 4, we follow $\mathcal{I} \models_p \mathrm{aft}(\varphi, w_{0i})[M]_\nu$ and thus, we can apply Lemma 52 to obtain a clause $\Psi \in \mathrm{aft}_\vee(\varphi, w_{0i})$ such that $\Psi[M]_\nu \subseteq \mathcal{I}$. Thus, the run follows for the first i steps a matching path in the initial component to arrive at Ψ. Assume we have $w_i \in \mathcal{L}(Q_{\Psi',M,N})$, then the run r branches off from the clause Ψ via $\delta_\frown$ to the accepting component $\mathcal{B}_{M,N}$ and follows the accepting run belonging to w_i.

Thus, it remains to show that right-hand side of the characterisation of $\mathcal{L}(Q_{\Psi',M,N})$ 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 $\Psi' = \Psi[M]_\nu \subseteq \mathcal{I}$ 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 which the run transitions via $\delta_\frown$ to $\mathcal{B}_{M,N}$ for some $\Psi \in \mathrm{aft}_\vee(\varphi, w_{0i})$, $M \subseteq \mu(\varphi)$, and $N \subseteq \nu(\varphi)$. This has to happen, since every accepting state is in some accepting component $\mathcal{B}_{M,N}$. Because the run r is accepting, w_i is in $\mathcal{L}(Q_{\Psi',M,N})$ for $\Psi' = \Psi[M]_\nu$ and we obtain the following from the LTL characterisation:

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

From the second and third fact, we can immediately follow (2′) and (3′) of Corollary 44 and for showing $w \models \varphi$, we only need to prove the remaining (1′): $w_i \models \mathrm{aft}(\varphi, w_{0i})[M]_\nu$. Let now $\mathcal{I} := \{\psi \in \mathrm{sf}(\mathrm{aft}(\varphi, w_{0i})[M]_\nu) : w_i \models \psi\}$ be a propositional assignment. Observe that from the first fact, we can follow $\Psi[M]_\nu \subseteq \mathcal{I}$. Then, we apply Lemma 52 and get $\mathcal{I} \models_p \mathrm{aft}(\varphi, w_{0i})[M]_\nu$. Using Lemma 4, we derive (1′) and we are done. □

8.3 Complexity Analysis

The elements of $\mathrm{Reach}_\vee(\varphi)$ are *sets* of temporal subformulas of φ, i.e., $\mathrm{Reach}_\vee(\varphi) \subseteq 2^{\mathrm{sf}(\varphi)}$. Since the number of temporal subformulas is bounded by the length of the formula, we immediately obtain $|\mathrm{Reach}_\vee(\varphi)| \leq 2^{|\mathrm{sf}(\varphi)|} \leq 2^n$ 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 $2^{|\mathrm{sf}(\varphi)|}$, the NBAs for $\mathbf{GF}(\mu LTL)$ have at most $2^{|\mathrm{sf}(\varphi)|+1}$, and the NBAs for $\mathbf{FG}(\nu LTL)$ have at most $2^{|\mathrm{sf}(\varphi)|+1} + 1$ 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., Reference [7].

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

$$|\mathrm{Reach}_\vee(\varphi)| + \sum_{\substack{M \subseteq \mu(\varphi) \\ N \subseteq \nu(\varphi)}} (|Q^1_M| \cdot |Q^2_{M,N}| \cdot |Q^3_{M,N}|).$$

We bound the size of each of these components by an exponential function. $\mathcal{B}^1_M$ 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 temporal subformulas of that formula is at most $|\mathrm{sf}(\varphi)| \leq n$ and so Q^1_M has at most 2^n elements. $\mathcal{B}^2_{M,N}$ is an intersection of $|M|$-many automata with at most 2^{n+1} states. We bound

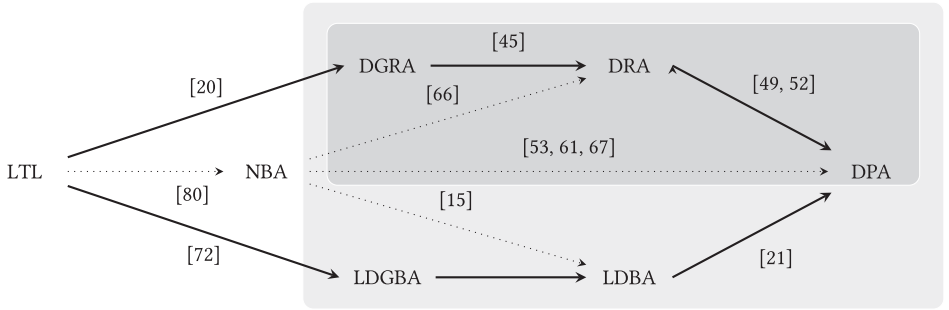

Fig. 13. Selection of LTL translations to different types of automata, excluding the translations presented in this article. Only syntactically defined classes are depicted; the semantically defined unambiguous, good-for-games, and good-for-MDPs automata are omitted. Classical translations are depicted as dotted arrows, and recent semantic approaches as thick arrows. The latter are implemented in `Rabinizer`; translation of NBAs to DRAs is implemented in `ltl2dstar` [40], to LDBAs in `Seminator` [9], and to DPAs in `Spot` [18] and `nbadet` [54]. The light-grey area indicates the automata suitable for probabilistic model checking, and the dark-grey area the ones suitable for the traditional automata-theoretic approach to synthesis.

$|M|$ 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. $\mathcal{B}^3_{M,N}$ is constructed from the formula $\bigwedge_{\psi \in N} \mathbf{G}(\psi[M]_\nu)$. This formula has at most $|\mathrm{sf}(\varphi)| + |\nu(\varphi)|$ temporal subformulas and thus the automaton has at most $2^{|\mathrm{sf}(\varphi)|+|\nu(\varphi)|} \leq 2^{2n}$ states. Going back to our initial bound and inserting the upper bounds for the components, we obtain:

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

This construction is refined in Reference [69] by replacing $\mathcal{B}^2_{M,N}$ by a component of size $n \cdot 2^{n+1}$, which reduces the upper bound to $O(2^{5n}) \subseteq 2^{O(n)}$.

9 APPLICATIONS

The three classes of ω-automata discussed here (NBAs, LDBAs, and DRAs) have a wide range of applications in verification and synthesis. In particular, NBAs are the standard for verification of (possibly concurrent) programs, and DRAs are the traditional choice for verification of and controller synthesis for systems with probabilistic behaviour, see, e.g., References [7, 14]. Here, we describe also the less-known use of LDBAs in both verification and synthesis.

First, the LTL synthesis problem [65] consists of determining whether there is a system that for all input streams produces output streams satisfying a given LTL formula, and, if that is the case, constructing such a system. Although the traditional automata-theoretic approach suggests to construct a DPA and further analyse it, this approach was not much in use due to bad scalability. The main bottleneck of the classic LTL→NBA→DPA translation path (see Figure 13) was the underlying determinisation in the second step. The situation changed when alternative paths to obtain DPAs from LTL appeared that side-stepped the determinisation of general NBA and that alleviated the scalability issue. First, References [20, 22] enabled the first step of the path LTL→D(G)RA→DPA, with the latter delivered by the index-appearance record [52], improved and tailored in Reference [49]. Second, a path LTL→LDBA→DPA was enabled by an efficient generation of LDBAs [72] together with a transformation of LDBAs to DPAs [21]. The latter path turned out to practically perform mostly even better than the former. As a result, the tool `Strix` [55, 57], based on the latter construction, has recently won all LTL tracks of the synthesis competition SyntComp [36]. Besides, the preserved semantic labelling of the states of the automata

allows for heuristics guiding the exploration of the on-the-fly generated automaton [55] together with efficient deployment of learning-based algorithms and lifelong learning paradigms in LTL synthesis [42]. Such efforts have a great impact on the practical performance of solutions to this 2-EXPTIME-complete problem.

Second, the problem of model checking probabilistic systems against an LTL specification [7] is to determine the probability that an LTL formula φ holds on the infinite paths generated by a given Markov chain $\mathcal{M}$, written $\Pr^{\mathcal{M}}(\varphi)$. More generally, for a Markov decision process $\mathcal{M}$, the problem is to determine the maximal (or minimal) probability that φ is satisfied, written $\sup_{\mathfrak{S}} \Pr^{\mathcal{M}_{\mathfrak{S}}}(\varphi)$, 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 $\mathcal{M}$. The automata-theoretic approach again prescribes creating a product of the system and the automaton for φ. However, as opposed to non-probabilistic model checking, the automaton $\mathcal{A}$ cannot be generally used if it is nondeterministic. Intuitively, resolving nondeterminism of the automaton may depend on the yet unknown, probabilistically given future.

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 of the two. While DRAs are typically used [44], algorithms using LDBAs are known [15, 78]. The second variant is *quantitative* probabilistic model checking, where one is interested in the exact satisfaction probability. Here, in contrast, LDBAs cannot be used in general. Recently it has been shown that LDBAs can be used if they satisfy certain conditions [33, 72]. These conditions are further discussed in Reference [34], which introduces *good-for-MDPs* automata. In particular, the LDBAs of Reference [72] or of this work satisfy these conditions after slightly adapting the product construction.

The key insight is that almost all words generated by $\mathcal{M}_{\mathfrak{S}}$ are ν- and μ-stable once a "bottom strongly connected component" is reached and one can resolve the LDBA nondeterminism upon that occasion. For a detailed proof, see Reference [69]. The advantages of using LDBAs compared to DRAs are the smaller size and that it is sufficient to decompose the product into "maximal end-components" only once. This improves the overall efficiency [73] compared to DRAs and often also to generalised DRAs [13, 45].

10 EXPERIMENTAL EVALUATION

We provide experimental evidence that the uniform translation techniques derived from the Master Theorem are not obtained at the expense of poor performance in practice. We focus on the representative case of deterministic Rabin automata and do not evaluate the translations to nondeterministic or limit-deterministic Büchi automata.

10.1 Method

Metric. We compare the size (number of states, number of Rabin pairs) of the automata produced by the different approaches. We do not include any resource consumption analysis, i.e., measurements of computation time or allocated memory. First, these values are very sensitive to implementation. Second, and more importantly, if a product construction is used, they are also typically negligible compared to the total resource consumption.

For each formula set $\{\varphi_1, \ldots, \varphi_n\}$ and for each translation approach, we report the geometric average $\sqrt[n]{\prod_{i=1}^{n} a_i}$ of the sizes $\{a_1, \ldots, a_n\}$ of the produced automata. Because of the identity

$$\sqrt[n]{\prod_{i=1}^{n} \frac{a_i}{b_i}} = \frac{\sqrt[n]{\prod_{i=1}^{n} a_i}}{\sqrt[n]{\prod_{i=1}^{n} b_i}}$$

Table 2. Parametrised Formulas Set

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

the geometric average of the size ratios $\{a_1/b_1, \dots, a_n/b_n\}$ for any two approaches with sizes $\{a_1, \dots, a_n\}$ and $\{b_1, \dots, b_n\}$ is given by the ratio of their geometric averages.

Input Formula Sets. We base the evaluation on two sets of formulas. The first set consists of the well-known "Dwyer"-patterns, which collects 55 LTL formulas specifying common properties [19]. The second set is obtained by instantiating the 11 parametrised formulas from Table 2. These families are either taken from References [29, 59, 77] or are simple combinations of **U**, **GF**, and **FG** formulas. The second set of formulas is useful to isolate and analyse strong and weak points of the compared translations. We do not report on randomly generated formulas, because in our experience formulas from real-world examples usually have a high degree of structure.

The formula sets are obtained by executing `genltl`[9] 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 remove (close to) duplicate entries: first, we bring formulas into negation normal form; second, we apply a standard set of LTL simplification rules [6, 24, 59, 68, 75] allowing us to turn off or at least restrict the LTL simplifier in the evaluation; third, we normalise the atomic propositions and remove formulas that are equal modulo renaming of atomic propositions.

As a consequence, the number of formulas we consider is less than the number of formulas of the corresponding original publications. For example, Reference [19] lists 55 formulas, but we remove six entries: e.g., 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 the two formula sets.

Output ω-Automata Class. For the sake of simplicity in this article, we have only presented constructions for NBAs, LDBAs, and DRAs with acceptance conditions defined on states. However, in practice acceptance conditions defined on transitions as well as generalised acceptance conditions, e.g., the generalised Büchi or Rabin condition, are preferred, because they yield automata with fewer states. Thus, when possible, we select deterministic generalised Rabin automata with acceptance conditions defined on transitions. We highlight this distinction by an *s* to denote state-acceptance (sDRA, sDGRA) and a *t* to denote transition-acceptance (tDRA, tDGRA).

Compared Translators. It is not the goal of this experimental evaluation to compare all known LTL→tDGRA-translators. We compare our approach with three translators of different types: a *historic* translator developed before the work on direct translations is begun; an actively maintained *state-of-the-art* translator that incorporates (most of) the advances made so far; and a direct translator.

[9] `genltl` is a component of `Spot` [18] to generate LTL formulas from existing patterns. We use the version `genltl (spot) 2.7.2`.

- `ltl2dstar` ([40], `0.5.4`). This tool uses `ltl2ba` to translate LTL to NBAs and then implements a classic *Safra-based* determinisation procedure. Safra-based translations have been the predominant state-of-the-art approaches before direct translations to deterministic automata have been investigated [45] and thus give a historic perspective. This configuration yields only sDRA, a subclass of tDGRA, which cannot in general yield as small automata.
- `ltl2tgba` ([18], `Spot 2.8.1`). The tool has been enriched with a determinisation construction and employs a *portfolio* approach to construct small automata, in our configuration tDGRA. It is to be noted that, in the meantime, constructions proposed here and the predecessor article [23] and related work such as Reference [59] have been adopted. Thus, one can argue that it is the state-of-the-art portfolio translator.
- `ltl2dgra` (`asymmetric`, [48], `Owl 20.06`).[9] This direct translation to DRAs and DGRAs has been described in Reference [20] and revised and corrected in Reference [22]. It uses all available optimisations, including the usual reduction rules for Rabin pairs, e.g., References [22, 27]. This approach is the most similar to ours, but treats the least- and greatest-fixed-point operators asymmetrically, focusing on the latter, and does not feature the double exponential upper bound.

Our own translator is:

- `ltl2dgra` (`symmetric`, `Owl 20.06`).[9] The tool implements the DRA construction of this article that treats both types of the operators symmetrically. We use a straightforward modification of the described DRA construction that yields tDGRA. We refer the interested reader to Reference [69] for an explicit description of this modification as well as for a list of applied optimisations. Most of these optimisations exploit the modular and semantic structure provided by the Master Theorem. Such optimisations are for example syntactic criteria for skipping sets M and N of Theorem 35 to remove redundant components, specialised intersection constructions for $\mathcal{B}^2_{M,N}$ and $C^3_{M,N}$, or better alternatives to the propositional equivalence relation $\sim$.

We include neither `ltl3dra` [5] nor `ltl3ba` [6] in combination with `ltl2dstar` in the comparison, since we consider them subsumed by the actively developed translator `ltl2tgba`.

The complete experimental setup, including the formula sets, the translators, and the scripts generating the tables, is available from Reference [70].

10.2 Results

The measured automata sizes for the LTL formulas are listed in Table 3. An entry $n(m)$ indicates that the automaton has n states and m Rabin pairs. We refer by φ to formulas from the "Dwyer" set, and by χ to formulas from the parametrised set. We write $\overline{\varphi}$ instead of $\neg\varphi$. The rows of the table are sorted by decreasing ratio between the largest and the smallest automaton. More precisely, we compute $\frac{max+1}{min+1}$ for each row, where *min* refers to the number of states of the smallest automaton and *max* refers to the number of states of the largest automaton. We then sort rows of the table according to this value.

10.3 Discussion

The results indicate the following:

- The new constructions show a clear improvement over the historic state before the semantic translations. The average ratios of `ltl2dgra (asymmetric)`,`ltl2dgra (symmetric)`, and

[9] The source-code of both tools is located in the repository of Reference [47].

Table 3. The Left Table Displays the Results for the "Dwyer" Set from Table 4 and The Right Table for the Parametrised Formulas from Table 2

LTL	ltl2dstar (historic)	ltl2dgra (asymmetric)	ltl2dgra (symmetric, this article)	ltl2tgba (portfolio)
φ_{39}	54,674 (11)	54 (8)	39 (4)	17
φ_{49}	26,030 (9)	22 (3)	51 (8)	12 (2)
φ_{44}	14,157 (9)	20 (3)	31 (7)	10 (2)
$\overline{\varphi_{49}}$	2,554 (11)	22 (8)	19 (3)	15 (2)
$\overline{\varphi_{44}}$	1,193 (11)	18 (7)	14 (3)	9 (2)
φ_{38}	964 (6)	20 (2)	9 (3)	20
φ_{48}	145 (5)	6 (2)	7 (3)	4
φ_{43}	137 (5)	6 (2)	7 (3)	4
φ_{28}	111 (4)	8 (2)	11 (3)	4
φ_{37}	61 (4)	8 (4)	6 (2)	5
φ_{14}	17 (3)	11 (3)	67 (4)	6
φ_{34}	32 (2)	8 (6)	13 (2)	3
φ_{35}	38 (3)	4 (3)	5 (2)	4
φ_{47}	25 (2)	4 (2)	3 (2)	3
φ_{29}	25 (3)	4 (2)	4 (2)	3
φ_{42}	24 (2)	5 (2)	3 (2)	3
φ_{33}	29 (2)	20 (5)	6 (2)	4
φ_{40}	20 (2)	4	3 (2)	6
φ_{45}	21 (2)	4	5 (2)	6
$\overline{\varphi_{39}}$	9	29 (6)	10 (2)	6
$\overline{\varphi_{40}}$	14 (3)	3 (2)	5	4
φ_{23}	12 (3)	4 (2)	3 (3)	3
$\overline{\varphi_{45}}$	15 (3)	5 (4)	5	4
φ_{13}	19 (3)	22 (2)	7 (2)	7
φ_{36}	16 (2)	6	6	6
$\frac{1}{n}\Sigma$	1,027.54	6.50	6.35	4.49
σ	6,204.59	7.27	9.22	2.94
$\sqrt[n]{\Pi}$	10.21	4.75	4.47	3.91
med.	5.0	4.0	4.0	4.0

LTL	ltl2dstar (historic)	ltl2dgra (asymmetric)	ltl2dgra (symmetric, this article)	ltl2tgba (portfolio)
$\chi_{9,3}$	255,274 (13)	1 (6)	1 (6)	1 (8)
$\chi_{8,4}$	12,278 (8)	1 (17)	15 (2)	1 (5)
$\overline{\chi_{8,4}}$	6,522 (7)	1 (5)	145 (2)	1 (10)
$\chi_{9,2}$	795 (7)	1 (3)	1 (3)	1 (4)
$\chi_{8,3}$	624 (6)	1 (10)	7 (2)	1 (4)
$\overline{\chi_{8,3}}$	424 (5)	1 (4)	45 (2)	1 (8)
$\overline{\chi_{8,2}}$	44 (3)	1 (3)	13 (2)	1 (6)
$\chi_{8,2}$	43 (4)	1 (5)	3 (2)	1 (3)
$\chi_{11,3}$	194 (9)	19 (4)	19 (4)	8 (4)
$\overline{\chi_{10,4}}$	319 (6)	65	31	16
$\overline{\chi_{9,4}}$	33 (4)	1 (7)	1 (4)	1 (4)
$\chi_{7,4}$	32 (5)	1 (5)	1 (5)	1 (5)
$\overline{\chi_{6,5}}$	32 (5)	1 (5)	1 (5)	1 (5)
$\chi_{7,3}$	16 (4)	1 (4)	1 (4)	1 (4)
$\overline{\chi_{6,4}}$	16 (4)	1 (4)	1 (4)	1 (4)
$\overline{\chi_{7,4}}$	15	1	1	`err`
$\overline{\chi_{9,3}}$	15 (3)	1 (5)	1 (3)	1 (3)
$\chi_{6,5}$	14	1	1	1
$\overline{\chi_{7,3}}$	11	1	1	`err`
$\chi_{6,4}$	11	1	1	1
$\chi_{6,3}$	8	1	1	1
$\chi_{7,2}$	8 (3)	1 (3)	1 (3)	1 (3)
$\overline{\chi_{6,3}}$	8 (3)	1 (3)	1 (3)	1 (3)
$\overline{\chi_{7,2}}$	7	1	1	`err`
$\overline{\chi_{9,2}}$	7 (2)	1 (3)	1 (2)	1 (2)
$\frac{1}{n}\Sigma$	4,265.75	8.44	11.56	5.92
σ	31,421.86	13.48	23.49	8.19
$\sqrt[n]{\Pi}$	19.72	3.35	4.14	3.01
med.	13.0	2.5	4.0	3.0

The tables list number of states, followed by the number of Rabin pairs (if larger than 1). The results for the remaining 73 and 41 formulas, respectively, are located in Appendix B. We write $\frac{1}{n}\Sigma$, σ, $\sqrt[n]{\Pi}$, and med., for the average, the standard deviation, the geometric average, and the median, respectively, for the number of states. We denote by `err` an error thrown by the tool.

Table 4. Formulas Referenced by Table 3

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_{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_{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}	$\mathbf{G}(a \vee b\ \mathbf{W}\ (c \vee (b \wedge d \wedge \mathbf{X}(b\ \mathbf{U}\ e))))$
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_{40}	$\mathbf{G}(a \vee \mathbf{F}(b \wedge \mathbf{X}\mathbf{F}c))$
φ_{42}	$\mathbf{G}(a \vee \mathbf{G}(b \vee (c \wedge \mathbf{X}\mathbf{F}d)))$
φ_{43}	$\mathbf{G}(a \vee \mathbf{G}\overline{b} \vee (c \vee \overline{b}\ \mathbf{U}\ (\overline{b} \wedge d \wedge \mathbf{X}(\overline{b}\ \mathbf{U}\ e)))\ \mathbf{U}\ b)$
φ_{44}	$\mathbf{G}(a \vee (b \vee \overline{c}\ \mathbf{U}\ (\overline{c} \wedge d \wedge \mathbf{X}(\overline{c}\ \mathbf{U}\ e)))\ \mathbf{U}\ (c \vee \mathbf{G}(b \vee (d \wedge \mathbf{X}\mathbf{F}e))))$
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}	$\mathbf{G}(a \vee (b \vee \overline{c}\ \mathbf{U}\ (\overline{c} \wedge d \wedge e \wedge \mathbf{X}((\overline{c} \wedge e)\ \mathbf{U}\ f)))\ \mathbf{U}\ (c \vee \mathbf{G}(b \vee (d \wedge e \wedge \mathbf{X}(e\ \mathbf{U}\ f)))))$

`ltl2tgba` to `ltl2dstar` are 46%, 43%, and 38% for the "Dwyer" set, and 17%, 21%, and 15% for the parametrised set.

- Our uniform approach is at least as good as the DRA-specific asymmetric approach in ~85% of the cases, with average ratio being 94% and 124%.
- Our approach is competitive with the "current best of" portfolio with average ratio of 114% and 138%. This is encouraging, since the amount of compositional optimisation in our implementation is still quite restricted and offers further possibilities for improvement.

A closer look at the results reveals that most formulas are not too complicated, and all approaches yield small automata of similar size. More concretely, in the "Dwyer" set, we only measured major differences for 25 formulas, and even less for the parametrised set. Major differences to the "historic" Safra-based approach seem to appear for formulas with more complex infinitary behaviour caused by deep nesting of **U** and **R**, e.g., φ_{39}, φ_{44}, and φ_{49}, and Boolean combinations of $\mathbf{FG}a$ and $\mathbf{GF}b$, e.g., $\chi_{8,n}$ and $\chi_{9,n}$.

The collected data demonstrate 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 systematic penalty in practice. 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.

11 CONCLUDING REMARKS

The Master Theorem we presented provides a decomposition of LTL formulas from which one can derive LTL translations in a straightforward way. This result builds upon work from a

series of publications [22, 45, 46, 72]. In particular, the idea that a word eventually stabilises with respect to a formula and the idea of inductively checking a complex LTL expression by delegating to auxiliary automata are already outlined there. Other translations such as the translation to LDBAs of References [38, 39] or the obligation sets of References [50, 51] follow similar ideas.

The Master Theorem is made possible by the addition of the operators **W** and **M** to the core LTL syntax, which makes it *complete* in the sense that for every modal operator there exists a greatest-fixed-point and a least-fixed-point variant. The essential novelty is that the mappings $\cdot[\cdot]_\mu$ and $\cdot[\cdot]_\nu$ exploit the existence of these variants and that their application to any formula φ yields a simpler formula, but *not in the sense one might expect*. In particular, $\varphi[N]_\mu$ might be *stronger* than φ. For example, the information that 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 $\varphi[M]_\nu$ and $\varphi[N]_\mu$ 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 References [45, 46], but could be applied essentially only to **F** and **G** operators. Hence, the result presented here successfully finishes the journey embarked upon in Reference [45]: A single theorem provides an arguably elegant (unified, symmetric, syntax-independent, not overly complex) and efficient (asymptotically optimal, practically relevant, direct) translation of LTL to ω-automata of your choice. Moreover, based on the Master Theorem one can derive an efficient, syntactic, and exponential normalisation procedure for LTL [71] that limits the nesting of greatest-fixed-point and least-fixed-pointer operators yielding a normal form described in Reference [12].

The practical relevance of these constructions and their precursors has been demonstrated by tools, too, as the reduced size of the automata plays the crucial role in speeding up LTL verification and synthesis. The tool `Rabinizer` [48] implements all these constructions. Its inception [27, 46] with the generalised Rabin condition [45] and integration [13, 41] into the probabilistic model checker `PRISM` [44] has led to the development of the standard HOA format [4] for ω-automata, covering less standard conditions, and to the extension of `PRISM` allowing for external LTL-to-HOA translators. Support for such constructions is then accessible in the reusable library `Owl` [47]; it distils the key functionalities of `Rabinizer`, which is a tool for end-users of LTL, into reusable modules and adds further support for developers. Further, `MoChiBa` [73] has extended `PRISM` to use also LDBAs. Besides, `Strix` [55] uses the translations to deterministic automata for synthesis of reactive systems from LTL specifications. In particular, to obtain a deterministic parity automaton, it makes use of a compositional construction that uses the simpler LTL fragments translations whenever possible, and only falling back to the more general procedure [21] for small parts of the formula.

Open Questions and Future Work. We have focused here on three common classes of ω-automata, namely, NBAs, LDBAs, and DRAs, and did not investigate other classes. It remains open whether we can obtain a direct translation to deterministic *parity* automata that is better than chaining existing constructions. Note that both Safra-based [53, 62, 67] as well as non-Safra-based [49, 52] techniques involve a "linearisation" of the acceptance condition using appearance records [32]. It is not clear whether LTL induces any special structure on them that could be exploited.

Finally, the presented constructions have a highly regular product structure, compared to Safra's determinisation, allowing for a symbolic implementation.

APPENDICES

A OMITTED PROOFS

A.1 Proof of Lemma 6

We need an auxiliary lemma.

Lemma 58. *Let f be a function on formulas such that $f(\mathbf{tt}) = \mathbf{tt}$, $f(\mathbf{ff}) = \mathbf{ff}$, $f(\chi_1 \wedge \chi_2) = f(\chi_1) \wedge f(\chi_2)$, and $f(\chi_1 \vee \chi_2) = f(\chi_1) \vee f(\chi_2)$ for all formulas χ_1 and χ_2. For every formula φ and every set of formulas $\mathcal{I}$:*

$$\mathcal{I} \models_p f(\varphi) \iff \{\chi \in \mathrm{sf}(\varphi) : \mathcal{I} \models_p f(\chi)\} \models_p \varphi.$$

Proof. We proceed by a structural induction on φ. We only consider two representative cases.

- Case $\varphi = \mathbf{X}\psi$:

$$\begin{aligned} \mathcal{I} \models_p f(\mathbf{X}\psi) &\iff \mathbf{X}\psi \in \{\chi \in \mathrm{sf}(\mathbf{X}\psi) : \mathcal{I} \models_p f(\chi)\} \\ &\iff \{\chi \in \mathrm{sf}(\mathbf{X}\psi) : \mathcal{I} \models_p f(\chi)\} \models_p \mathbf{X}\psi \end{aligned}$$

- Case $\varphi = \psi_1 \wedge \psi_2$:

$$\begin{aligned} & \mathcal{I} \models_p f(\psi_1 \wedge \psi_2) \\ \iff & \mathcal{I} \models_p f(\psi_1) \wedge \mathcal{I} \models_p f(\psi_2) \\ \iff & \{\chi \in \mathrm{sf}(\psi_1) : \mathcal{I} \models_p f(\chi)\} \models_p \psi_1 \\ & \wedge \{\chi \in \mathrm{sf}(\psi_2) : \mathcal{I} \models_p f(\chi)\} \models_p \psi_2 \qquad \text{(induction hypothesis)} \\ \iff & \{\chi \in \mathrm{sf}(\psi_1 \wedge \psi_2) : \mathcal{I} \models_p f(\chi)\} \models_p \psi_1 \wedge \psi_2 \end{aligned}$$

□

Lemma 6. *Let f be a function on formulas such that $f(\mathbf{tt}) = \mathbf{tt}$, $f(\mathbf{ff}) = \mathbf{ff}$, and $f(\chi_1 \wedge \chi_2) = f(\chi_1) \wedge f(\chi_2)$, $f(\chi_1 \vee \chi_2) = f(\chi_1) \vee f(\chi_2)$ for all formulas χ_1 and χ_2. For every pair of formulas φ and ψ, if $\varphi \sim \psi$, then $f(\varphi) \sim f(\psi)$.*

Proof. By symmetry it suffices to show $\mathcal{I} \models_p f(\varphi) \Rightarrow \mathcal{I} \models_p f(\psi)$ for every assignment $\mathcal{I}$. We derive this as follows:

$$\begin{aligned} \mathcal{I} \models_p f(\varphi) &\iff \{\chi \in \mathrm{sf}(\varphi) : \mathcal{I} \models_p f(\chi)\} \models_p \varphi && \text{(Lemma 58)} \\ &\iff \{\chi \in \mathrm{sf}(\varphi) : \mathcal{I} \models_p f(\chi)\} \models_p \psi && (\varphi \sim \psi) \\ &\Rightarrow \{\chi \in \mathrm{sf}(\varphi) \cap \mathrm{sf}(\psi) : \mathcal{I} \models_p f(\chi)\} \models_p \psi && \text{(restriction to } \mathrm{sf}(\psi)) \\ &\Rightarrow \{\chi \in \mathrm{sf}(\psi) : \mathcal{I} \models_p f(\chi)\} \models_p \psi && \text{(monotonicity of } \models_p) \\ &\iff \mathcal{I} \models_p f(\psi) && \text{(Lemma 58)} \end{aligned}$$

□

A.2 Proof of Lemma 30

Lemma 30. *Let φ be a formula and let w be a word. Let $M \subseteq \mu(\varphi)$ be a set of formulas. We have:*

(1) If $\mathcal{F}_w^\varphi \subseteq M$ and $w \models \varphi$, then $w \models \varphi[M]_\nu$.
(2) If $M \subseteq \mathcal{GF}_w^\varphi$ and $w \models \varphi[M]_\nu$, then $w \models \varphi$.

In particular:

(3) If $\mathcal{F}_w^\varphi = M = \mathcal{GF}_w^\varphi$, then $w \models \varphi \iff w \models \varphi[M]_\nu$.

Let $N \subseteq \nu(\varphi)$ be a set of formulas. We have:

(4) If $\mathcal{FG}_w^\varphi \subseteq N$ and $w \models \varphi$, then $w \models \varphi[N]_\mu$.
(5) If $N \subseteq \mathcal{G}_w^\varphi$ and $w \models \varphi[N]_\mu$, then $w \models \varphi$.

In particular:

(6) If $\mathcal{FG}^{\varphi}_{w} = N = \mathcal{G}^{\varphi}_{w}$, *then* $w \models \varphi \iff w \models \varphi[N]_\mu$.

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

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

$$\forall i.\, (\, (w_i \models \varphi) \Rightarrow (w_i \models \varphi[N]_\mu)\,).$$

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

$$\begin{aligned}
& w_i \models \psi_1 \,\mathbf{W}\, \psi_2 \\
\iff\ & w_i \models \psi_1 \,\mathbf{U}\, \psi_2 && (w_i \not\models \mathbf{G}\psi_1) \\
\iff\ & \exists j.\, w_{i+j} \models \psi_2 \wedge \forall k < j.\, w_{i+k} \models \psi_1 \\
\Rightarrow\ & \exists j.\, w_{i+j} \models \psi_2[N]_\mu \wedge \forall k < j.\, w_{i+k} \models \psi_1[N]_\mu && \text{(induction hypothesis)} \\
\iff\ & w_i \models (\psi_1[N]_\mu) \,\mathbf{U}\, (\psi_2[N]_\mu) \\
\iff\ & w_i \models (\psi_1 \,\mathbf{W}\, \psi_2)[N]_\mu && (\varphi \notin N, \text{Definition 28})
\end{aligned}$$

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

$$\begin{aligned}
& w_i \models \psi_1 \vee \psi_2 \\
\iff\ & w_i \models \psi_1 \vee w_i \models \psi_2 \\
\Rightarrow\ & w_i \models \psi_1[N]_\mu \vee w_i \models \psi_2[N]_\mu && \text{(induction hypothesis)} \\
\iff\ & w_i \models (\psi_1 \vee \psi_2)[N]_\mu && \text{(Definition 28)}
\end{aligned}$$

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

$$\forall i.\, (\, (w_i \models \varphi[N]_\mu) \Rightarrow (w_i \models \varphi)\,).$$

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

$$\begin{aligned}
& w_i \models (\psi_1 \,\mathbf{W}\, \psi_2)[N]_\mu \\
\iff\ & w_i \models (\psi_1[N]_\mu) \,\mathbf{U}\, (\psi_2[N]_\mu) && (\varphi \notin N, \text{Definition 28}) \\
\iff\ & \exists j.\, w_{i+j} \models \psi_2[N]_\mu \wedge \forall k < j.\, w_{i+k} \models \psi_1[N]_\mu \\
\Rightarrow\ & \exists j.\, w_{i+j} \models \psi_2 \wedge \forall k < j.\, w_{i+k} \models \psi_1 && \text{(induction hypothesis)} \\
\iff\ & w_i \models \psi_1 \,\mathbf{U}\, \psi_2 \\
\Rightarrow\ & w_i \models \psi_1 \,\mathbf{W}\, \psi_2
\end{aligned}$$

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

$$\begin{aligned}
& w_i \models (\psi_1 \vee \psi_2)[N]_\mu \\
\iff\ & w_i \models \psi_1[N]_\mu \vee w_i \models \psi_2[N]_\mu && \text{(Definition 28)} \\
\Rightarrow\ & w_i \models \psi_1 \vee w_i \models \psi_2 && \text{(induction hypothesis)} \\
\iff\ & w_i \models \psi_1 \vee \psi_2
\end{aligned}$$

□

A.3 Proof of Proposition 13

Proposition 13. *Let $\varphi \in \mu LTL$.*

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

$$\mathcal{B}^{\varphi}_{\mu} = (\mathrm{Reach}(\varphi)_{/\sim}, \mathrm{aft}_{\sim}, [\varphi]_{\sim}, \mathrm{Inf}(\{[\mathbf{tt}]_{\sim}\})).$$

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

$$C^{\varphi}_{\mu} = (\mathrm{Reach}(\varphi)_{/\sim}, \mathrm{aft}_{\sim}, [\varphi]_{\sim}, \mathrm{Fin}(\overline{\{[\mathbf{tt}]_{\sim}\}})).$$

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

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

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

Let $\varphi \in \nu LTL$.

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

$$C^{\varphi}_{\nu} = (\mathrm{Reach}(\varphi)_{/\sim}, \mathrm{aft}_{\sim}, [\varphi]_{\sim}, \mathrm{Fin}(\{[\mathbf{ff}]_{\sim}\})).$$

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

$$\mathcal{B}^{\varphi}_{\nu} = (\mathrm{Reach}(\varphi)_{/\sim}, \mathrm{aft}_{\sim}, [\varphi]_{\sim}, \mathrm{Inf}(\overline{\{[\mathbf{ff}]_{\sim}\}})).$$

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

$$C^{\varphi}_{\mathrm{FG}\nu} = (\mathrm{Reach}(\mathbf{G}\varphi)_{/\sim}, \mathrm{aft}^{\mathrm{G}\varphi}_{\sim}, [\mathbf{G}\varphi]_{\sim}, \mathrm{Fin}(\{[\mathbf{ff}]_{\sim}\}))$$

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

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

($\Rightarrow_1$) Assume $w \models \varphi$. Using Lemma 12, we can find an index i such that $\mathrm{aft}(\varphi, w_{0i}) \sim \mathbf{tt}$. Hence, the run r with $r[i] = [\mathrm{aft}(\varphi, w_{0i})]_{\sim}$ 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 $[\mathbf{tt}]_{\sim} = \mathrm{aft}_{\sim}([\varphi]_{\sim}, w_{0i}) = [\mathrm{aft}(\varphi, w_{0i})]_{\sim}$. Thus, applying Lemma 12 to $\mathrm{aft}(\varphi, w_{0i}) \sim \mathbf{tt}$ yields $w \models \varphi$ and we are done.

(2) Observe that $\mathcal{B}^{\varphi}_{\mu}$ and C^{φ}_{μ} 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 C^{φ}_{μ} and vice versa. Thus, $\mathcal{L}(\mathcal{B}^{\varphi}_{\mu}) = \mathcal{L}(C^{\varphi}_{\mu})$ and (2) is an intermediate consequence of (1).

($\Rightarrow_3$) Assume $w \models \mathbf{GF}\varphi$. This is equivalent to $\forall i.\, w_i \models \mathbf{F}\varphi$. Then, we obtain by Lemma 12: $\forall i.\, \exists j.\, \mathrm{aft}(\mathbf{F}\varphi, w_{ij}) \sim \mathbf{tt}$. Since $\mathcal{B}^{\varphi}_{\mathrm{GF}\mu}$ 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 $\mathrm{aft}(\mathbf{F}\varphi, w_{0j}) \sim \mathbf{tt}$. Observe that this is the first point where $\mathrm{aft}_{\sim}$ and $\mathrm{aft}^{\mathrm{F}\varphi}_{\sim}$ 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 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 $\mathcal{L}(\mathcal{B}^{\varphi}_{\mathrm{GF}\mu})$.

($\Leftarrow_3$) Assume w is accepted by $\mathcal{B}^{\varphi}_{\mathrm{GF}\mu}$. Then there exists an accepting run r for w and this run visits the state $[\mathbf{tt}]_\sim$ infinitely often. Provided we have shown $\forall i.\ \exists j \geq i.\ \exists k.\ \mathrm{aft}(\mathbf{F}\varphi, w_{jk}) \sim \mathbf{tt}$, we apply Lemma 12 to obtain $\forall i.\ \exists j \geq i.\ w_j \models \mathbf{F}\varphi$, which is equivalent to $w \models \mathbf{GF}\varphi$, which we wanted to show. We now focus on the remaining goal $\forall i.\ \exists j \geq i.\ \exists k.\ \mathrm{aft}(\mathbf{F}\varphi, w_{jk}) \sim \mathbf{tt}$: 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 $\mathrm{aft}(\varphi, w_{jk}) \sim \mathbf{tt}$ and fill the remaining gap.

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

A.4 Proofs of Lemma 51, Lemma 52, and Proposition 53

We first show how to obtain a satisfying propositional assignment for φ from a satisfying assignment for $\mathrm{aft}(\varphi, \sigma)$. Second, we establish Lemma 51 for a single letter σ in Lemma 59 and then generalise it to arbitrary finite words.

Lemma 59. *Let Φ be a set of formulas, let σ be a letter, and let $\mathcal{I}$ be a propositional assignment. Then:*

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

Proof. Before we begin with the proof, we make the following technical observation for $\otimes_{min}$: Let $\mathcal{X}$ and the elements of $\mathcal{X}$ be finite sets. Then for every set A and X, we have:

$$A \in \bigotimes_{min} \mathcal{X} \wedge X \in \mathcal{X} \Rightarrow \exists B \in X.\ B \subseteq A.$$

($\Leftarrow$) Assume there exists a subset $\Psi \subseteq \mathcal{I}$ of the propositional assignment $\mathcal{I}$ with $\Psi \in \mathrm{aft}_\vee(\Phi, \sigma)$. Further, let $\psi \in \Phi$ be an arbitrary element of the set Φ. We unfold the definition of $\mathrm{aft}_\vee$ and see that by the previous observation for $\otimes_{min}$ that there exists some $\Psi' \subseteq \Psi$ with $\Psi' \in \mathrm{dnf}(\mathrm{aft}(\psi, \sigma))$ that contributed to Ψ for $\psi \in \Phi$. Since Ψ' is an element of the DNF, we have $\Psi' \models_p \mathrm{aft}(\psi, \sigma)$ (Proposition 47) and due to the monotonicity of $\models_p$, we also have $\mathcal{I} \models_p \mathrm{aft}(\psi, \sigma)$.

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

- Base $|\Phi| = 0$. Then the right-hand side evaluates to $\mathrm{aft}_\vee(\Phi, \sigma) = \{\emptyset\}$. 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 $\mathcal{I} \models_p \mathrm{aft}(\psi, \sigma)$ for all formulas $\psi \in \Phi$. First, this also holds for the subset Φ' and applying the induction hypothesis to Φ' yields $\exists \Psi' \subseteq \mathcal{I}.\ \Psi' \in \mathrm{aft}_\vee(\Phi', \sigma)$. Let now Ψ' be such a set of formulas. Second, with Proposition 47 applied to ψ'', we obtain a set $\Psi'' \subseteq \mathcal{I}$ with $\Psi'' \in \mathrm{dnf}(\mathrm{aft}(\psi'', \sigma))$. Taking these two steps together, we have $\Psi' \cup \Psi'' \in (\mathrm{aft}_\vee(\Phi', \sigma)) \otimes (\mathrm{aft}_\vee(\{\psi''\}, \sigma))$. Hence, there exists some subset $\Psi \subseteq \Psi' \cup \Psi''$ that is contained in $(\mathrm{aft}_\vee(\Phi', \sigma)) \otimes_{min} (\mathrm{aft}_\vee(\{\psi''\}, \sigma)) = \mathrm{aft}_\vee(\Phi', \sigma)$ and we are done. □

Lemma 51. *Let φ be a formula, let w be a finite word, and let $\mathcal{I}$ be a propositional assignment. Then:*

$$\mathcal{I} \models_p \mathrm{aft}(\varphi, w) \iff \exists \Psi \subseteq \mathcal{I}.\ \Psi \in \mathrm{aft}_\vee(\varphi, w).$$

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

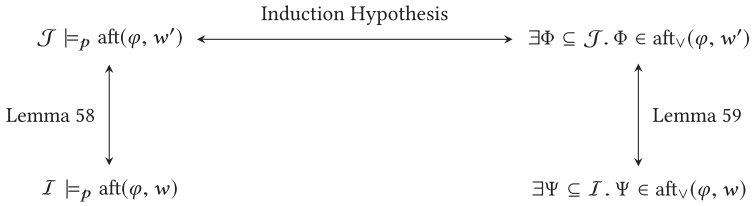

Fig. 14. Induction step proof structure of Lemma 51.

- Base $w = \epsilon$. Simplifying the statement, we see that we need to prove: $\mathcal{I} \models_p \varphi \iff \exists \Psi \subseteq \mathcal{I}.\, \Psi \in \mathrm{dnf}(\varphi)$. This immediately follows from Proposition 47.
- Step $w = w'\sigma$. We prove both directions separately and the proofs follow the structure sketched in Figure 14. For both directions, we define the propositional assignment $\mathcal{J} = \{\psi : \mathcal{I} \models_p \mathrm{aft}(\psi, \sigma)\}$.
  ($\Rightarrow$) Assume $\mathcal{I} \models_p \mathrm{aft}(\varphi, w)$ holds. Applying Lemma 58 gives us $\mathcal{J} \models_p \mathrm{aft}(\varphi, w')$ and with the induction hypothesis, we obtain a set Φ such that $\Phi \subseteq \mathcal{J}$ and $\Phi \in \mathrm{aft}_\vee(\varphi, w')$. By definition of $\mathcal{J}$, we have $\mathcal{I} \models_p \mathrm{aft}(\psi, \sigma)$ for all $\psi \in \mathcal{J}$ and thus also all $\psi \in \Phi$. Hence, we can apply Lemma 59 and have $\exists \Psi \subseteq \mathcal{I}.\, \Psi \in \mathrm{aft}_\vee(\Phi, \sigma)$. Since $\Phi \in \mathrm{aft}_\vee(\varphi, w')$, we have $\mathrm{aft}_\vee(\Phi, \sigma) \subseteq \mathrm{aft}_\vee(\varphi, w'\sigma)$ and are done.
  ($\Leftarrow$) Let Ψ and $\mathcal{I}$ be sets such that $\Psi \subseteq \mathcal{I}$ and $\Psi \in \mathrm{aft}_\vee(\varphi, w)$. Further let Φ be a set such that $\Phi \in \mathrm{aft}_\vee(\varphi, w')$ and $\Psi \in \mathrm{aft}_\vee(\Phi, \sigma)$. Assuming we proved $\Phi \subseteq \mathcal{J}$, we can simply apply the induction hypothesis and Lemma 58 to obtain at $\mathcal{I} \models_p \mathrm{aft}(\varphi, w)$. Thus, it remains to show: $\Phi \subseteq \mathcal{J}$. Let ψ an arbitrary formula from the set Φ. To show the inclusion, we need to establish $\mathcal{I} \models_p \mathrm{aft}(\psi, \sigma)$, but this exactly follows from Lemma 59 and we are done. □

Lemma 52. *Let φ be a formula, let w be a finite word, let $M \subseteq \mu(\varphi)$, and let $\mathcal{I} \subseteq \mathrm{sf}(\varphi)$. Then:*

(1) $\mathrm{aft}_\vee(\varphi, w) \subseteq 2^{\mathrm{sf}(\varphi)}$,
(2) $\emptyset \in \mathrm{aft}_\vee(\varphi, w) \iff \mathrm{aft}(\varphi, w) \sim \mathbf{tt}$,
(3) $\mathrm{aft}_\vee(\varphi, w) = \emptyset \iff \mathrm{aft}(\varphi, w) \sim \mathbf{ff}$,
(4) $\mathcal{L}(\mathrm{aft}(\varphi, w)) = \bigcup\{\bigcap_{\psi \in \Psi} \mathcal{L}(\psi) : \Psi \in \mathrm{aft}_\vee(\varphi, w)\}$,
(5) Let $\Psi[M]_\nu := \{\psi[M]_\nu : \psi \in \Psi\}$. *Then:*

$$\mathcal{I} \models_p \mathrm{aft}(\varphi, w)[M]_\nu \iff \exists \Psi.\, \Psi[M]_\nu \subseteq \mathcal{I} \wedge \Psi \in \mathrm{aft}_\vee(\varphi, w).$$

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:

$$\begin{aligned} \mathrm{aft}(\varphi, w) \sim \mathbf{tt} &\iff \emptyset \models_p \mathrm{aft}(\varphi, w) \\ &\iff \emptyset \in \mathrm{aft}_\vee(\varphi, w) \quad \text{(Lemma 51)}. \end{aligned}$$

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

$$\begin{aligned} \mathrm{aft}(\varphi, w) \sim \mathbf{ff} &\iff \mathcal{U} \not\models_p \mathrm{aft}(\varphi, w) && (\models_p \text{ is monotone}) \\ &\iff \forall \Psi \subseteq \mathcal{U}.\, \Psi \notin \mathrm{aft}_\vee(\varphi, w) && \text{(Lemma 51)} \\ &\iff \forall \Psi.\, \Psi \notin \mathrm{aft}_\vee(\varphi, w) && \text{(every set of formulas is a subset of } \mathcal{U}) \\ &\iff \mathrm{aft}_\vee(\varphi, w) = \emptyset. \end{aligned}$$

(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:

$$\begin{aligned}
w' \in \mathcal{L}(\mathrm{aft}(\varphi, w)) &\iff w' \models \mathrm{aft}(\varphi, w) && \text{(Lemma 9)}\\
&\iff \mathcal{I} \models_p \mathrm{aft}(\varphi, w) && \text{(Lemma 4)}\\
&\iff \exists \Psi.\, \Psi \in \mathrm{aft}_\vee(\varphi, w) \wedge \Psi \subseteq \mathcal{I} && \text{(Lemma 51)}\\
&\iff \exists \Psi.\, \Psi \in \mathrm{aft}_\vee(\varphi, w) \wedge \forall \psi \in \Psi.\, w' \models \psi\\
&\iff w' \in \bigcup \left\{ \bigcap_{\psi \in \Psi} \mathcal{L}(\psi) : \Psi \in \mathrm{aft}_\vee(\varphi, w) \right\}.
\end{aligned}$$

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

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

Proof of the claim: We proceed by a structural induction on χ. As before we only consider a subset of the cases: one representative of the "interesting" cases and two of the "straightforward" cases:

- Case $\chi = \chi_1 \,\mathbf{U}\, \chi_2$: We proceed by a further case distinction. Assume $\chi_1 \,\mathbf{U}\, \chi_2 \notin M$. In this case, we have on the left side of the equivalence: $\mathcal{I} \not\models_p \chi[M]_\nu = \mathbf{ff}$. For the right side observe that by the assumption $\mathcal{I} \subseteq \mathrm{sf}(\varphi)$ of the lemma the set $\mathcal{I}$ does not contain $\mathbf{ff}$. Thus, $\chi \notin \{\psi : \psi[M]_\nu \in \mathcal{I}\}$ and we follow $\{\psi : \psi[M]_\nu \in \mathcal{I}\} \not\models_p \chi$. Let us now assume $\chi_1 \,\mathbf{U}\, \chi_2 \in M$. We then derive:

$$\begin{aligned}
&\mathcal{I} \models_p \chi[M]_\nu\\
\iff &\mathcal{I} \models_p (\chi_1[M]_\nu) \,\mathbf{W}\, (\chi_2[M]_\nu)\\
\iff &(\chi_1[M]_\nu) \,\mathbf{W}\, (\chi_2[M]_\nu) \in \mathcal{I}\\
\iff &\chi_1 \,\mathbf{U}\, \chi_2 \in \{\psi : \psi[M]_\nu \in \mathcal{I}\}\\
\iff &\{\psi : \psi[M]_\nu \in \mathcal{I}\} \models_p \chi.
\end{aligned}$$

- Case $\chi = \chi_1 \,\mathbf{W}\, \chi_2$:

$$\begin{aligned}
&\mathcal{I} \models_p \chi[M]_\nu\\
\iff &\mathcal{I} \models_p (\chi_1[M]_\nu) \,\mathbf{W}\, (\chi_2[M]_\nu)\\
\iff &(\chi_1[M]_\nu) \,\mathbf{W}\, (\chi_2[M]_\nu) \in \mathcal{I}\\
\iff &\chi_1 \,\mathbf{W}\, \chi_2 \in \{\psi : \psi[M]_\nu \in \mathcal{I}\}\\
\iff &\{\psi : \psi[M]_\nu \in \mathcal{I}\} \models_p \chi.
\end{aligned}$$

- Case $\chi = \chi_1 \wedge \chi_2$:

$$\begin{aligned}
&\mathcal{I} \models_p \chi[M]_\nu\\
\iff &\mathcal{I} \models_p \chi_1[M]_\nu \wedge \chi_2[M]_\nu\\
\iff &\{\psi : \psi[M]_\nu \in \mathcal{I}\} \models_p \chi_1 \wedge \{\psi : \psi[M]_\nu \in \mathcal{I}\} \models_p \chi_2 && \text{(induction hypothesis)}\\
\iff &\{\psi : \psi[M]_\nu \in \mathcal{I}\} \models_p \chi.
\end{aligned}$$

($\Rightarrow_5$) Assume $\mathcal{I} \models_p \mathrm{aft}(\varphi, w)[M]_\nu$. Hence, the assignment $\mathcal{J} = \{\psi : \psi[M]_\nu \in \mathcal{I}\}$ is a satisfying assignment of $\mathrm{aft}(\varphi, w)$, i.e., $\mathcal{J} \models_p \mathrm{aft}(\varphi, w)$, which follows from the previous claim. By Lemma 51 there exists some Ψ such that $\Psi \subseteq \mathcal{J}$ and $\Psi \in \mathrm{aft}_\vee(\varphi, w)$. Thus, $\Psi[M]_\nu \subseteq \mathcal{I}$ and we are done.

($\Leftarrow_5$) Assume $\Psi[M]_\nu \subseteq \mathcal{I}$ and $\Psi \in \mathrm{aft}_\vee(\varphi, w)$. Further, let $\mathcal{J} = \{\psi : \psi[M]_\nu \in \mathcal{I}\}$. Then $\Psi \subseteq \mathcal{J}$ and we can apply the other direction of Lemma 51 to get $\mathcal{J} \models_p \mathrm{aft}(\varphi, w)$. Applying the claim, we obtain $\mathcal{I} \models_p \mathrm{aft}(\varphi, w)[M]_\nu$ and we are done. □

Proposition 53. *Let $\varphi \in \mu LTL$.*

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

$$\mathcal{B}_\mu^\varphi = (\,\mathrm{Reach}_\vee(\varphi), \mathrm{aft}_\vee, \mathrm{dnf}(\varphi), \mathrm{Inf}(\{\emptyset\})\,).$$

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

$$\mathcal{B}_{\mathbf{GF}\mu}^\varphi = \left(\,\mathrm{Reach}_\vee(\mathbf{F}\varphi), \mathrm{aft}_\vee^{\mathbf{F}\varphi}, \{\{\mathbf{F}\varphi\}\}, \mathrm{Inf}(\{\emptyset\})\,\right)$$

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

Let $\varphi \in \nu LTL$.

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

$$\mathcal{B}_\nu^\varphi = (\,\mathrm{Reach}_\vee(\varphi), \mathrm{aft}_\vee, \mathrm{dnf}(\varphi), \mathrm{Inf}(\mathrm{Reach}_\vee(\varphi))\,).$$

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

$$\mathcal{B}_{\mathbf{FG}\nu}^\varphi = \left(\,\mathrm{Reach}_\vee(\mathbf{G}\varphi) \cup \{\{\mathbf{FG}\varphi\}\}, \mathrm{aft}_\vee^{\mathbf{G}\varphi}, \{\{\mathbf{FG}\varphi\}\}, \mathrm{Inf}(\mathrm{Reach}_\vee(\mathbf{G}\varphi))\,\right)$$

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

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

($\Rightarrow_1$) Assume $w \models \varphi$. Using Lemma 12 with $\sim$, we have $\mathrm{aft}(\varphi, w_{0i}) \sim \mathbf{tt}$ for some i and also $\emptyset \in \mathrm{aft}_\vee(\varphi, w_{0i})$ using Lemma 52. Hence, there exists some 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 $\mathcal{B}_\mu^\varphi$.

($\Leftarrow_1$) Assume w is accepted by $\mathcal{B}_\mu^\varphi$. Then there exists an accepting run r for w and this run visits the state $\emptyset$ infinitely often. Let i be the index such that $\emptyset \in \mathrm{aft}_\vee(\varphi, w_{0i})$. Applying Lemma 52, we have $\mathrm{aft}(\varphi, w_{0i}) \sim \mathbf{tt}$ and with Lemma 12, 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 to $\forall i.\exists j.\, \mathrm{aft}(\mathbf{F}\varphi, w_{ij}) \sim \mathbf{tt}$ according to Lemma 12. By applying Lemma 51, we obtain $\forall i.\exists j.\, \emptyset \in \mathrm{aft}_\vee(\mathbf{F}\varphi, w_{ij})$. 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 $\emptyset \in \mathrm{aft}_\vee(\mathbf{F}\varphi, w_{0j})$. Then the run follows this path to reach $\emptyset$ for the segment w_{0j} and thus $r[j-1] = \emptyset$. After visiting the accepting state the run returns to the initial state and we repeat this from $r[j] = \{\mathbf{F}\varphi\}$. Applying this construction again and again, we obtain an accepting run and thus the word w is accepted by $\mathcal{B}_{\mathbf{GF}\mu}^\varphi$.

($\Leftarrow_2$) Assume w is accepted by $\mathcal{B}_{\mathbf{GF}\mu}^\varphi$. Then there exists an accepting run r for w and this run visits the state $\emptyset$ infinitely often. Provided we have shown $\forall i.\exists j \geq i.\exists k.\, \mathrm{aft}(\mathbf{F}\varphi, w_{jk}) \sim \mathbf{tt}$, we apply Lemma 12 to obtain $\forall i.\, \exists j \geq i.\, w_j \models \mathbf{F}\varphi$, which is equivalent to $w \models \mathbf{GF}\varphi$, which we wanted to show. We now focus on the remaining goal $\forall i.\exists j \geq i.\exists k.\, \mathrm{aft}(\mathbf{F}\varphi, w_{jk}) \sim \mathbf{tt}$: Let i be an arbitrary index. Since r is an accepting run, there exist infinitely many j's, such that $r[j-1] = \emptyset$ 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 j and k. Thus, $\emptyset \in \mathrm{aft}_\vee(\varphi, w_{jk})$ and with Lemma 52, 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) $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{B}_\nu^\varphi)$ and (4) $\mathcal{L}(\mathbf{FG}\varphi) = \mathcal{L}(\mathcal{B}_{\mathbf{FG}\nu}^\varphi)$. 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 $\mathcal{B}_{\mathbf{FG}\nu}^\varphi$ adds to

the structure of $\mathcal{B}_{\nu}^{\mathrm{G}\varphi}$ just one additional, non-accepting state with a self-loop and a transition to $\{\mathrm{G}\varphi\}$.

($\Rightarrow_4$) Assume $w \models \mathbf{FG}\varphi$. Then there exists some i such that $w_i \models \mathrm{G}\varphi$. An accepting run just stays in $\mathbf{FG}\varphi$ up to i and then moves to $\mathrm{G}\varphi$. 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 $\{\mathrm{G}\varphi\}$. By (3), we know that then $w_i \models \mathrm{G}\varphi$ and hence $w \models \mathbf{FG}\varphi$. □

B OMITTED TABLES

Table 5. Remaining Formulas from the "Dwyer" Set from Table 3

LTL	ltl2dstar (historic)	ltl2dgra (asymmetric)	ltl2dgra (symmetric, this article)	ltl2tgba (portfolio)
$\overline{\varphi_{13}}$	8	20	8	8
$\overline{\varphi_{14}}$	7	16 (3)	7	7
φ_9	5	2	2 (2)	2
φ_{25}	7 (2)	3	4	3
φ_{27}	9 (2)	4	5	4
φ_{31}	8 (3)	5	5	4
φ_4	6 (2)	4 (2)	3 (2)	3
φ_{18}	6 (2)	4 (2)	4 (2)	3
φ_{30}	6 (2)	3 (2)	3	3
φ_{24}	6	6 (3)	3 (2)	3
φ_{15}	4	2	2	2
φ_6	4	2	2	2
φ_{20}	4	2	2 (2)	2
φ_3	4	2	2	2
φ_5	4	2 (2)	2	2
φ_8	4	2 (2)	2	2
φ_{19}	4	2 (2)	2	2
φ_{32}	7 (2)	4 (2)	5	4
φ_1	2	1	1	1
φ_{16}	5 (2)	3	3	3
$\overline{\varphi_{17}}$	5	3	3 (2)	3
φ_{22}	5	4 (2)	3 (2)	3
$\overline{\varphi_{28}}$	5	8	5	5
$\overline{\varphi_{43}}$	5	8	5	5
$\overline{\varphi_{48}}$	5	8	5	5
φ_{26}	6 (2)	4	4	4
$\overline{\varphi_{35}}$	6	4	4	4
$\overline{\varphi_{41}}$	5	6	4	4
$\overline{\varphi_{34}}$	4	6	6	4
$\overline{\varphi_{46}}$	5	6	4	4
$\overline{\varphi_{15}}$	3	2	2	2
$\overline{\varphi_6}$	3	2	2	2
$\overline{\varphi_7}$	3	2	2	2
$\overline{\varphi_{36}}$	7	5	5	5
$\overline{\varphi_{37}}$	7	5	5	5
$\overline{\varphi_2}$	4	3	3	3
φ_2	4 (2)	3	3	3

LTL	ltl2dstar (historic)	ltl2dgra (asymmetric)	ltl2dgra (symmetric, this article)	ltl2tgba (portfolio)
$\overline{\varphi_{16}}$	4	3	3	3
$\overline{\varphi_{25}}$	4	3	3	3
$\overline{\varphi_{21}}$	4	3	3	3
$\overline{\varphi_{30}}$	4	3	3	3
φ_{21}	4 (2)	3	3	3
$\overline{\varphi_{11}}$	8	9	7	7
$\overline{\varphi_{26}}$	5	5	4	4
$\overline{\varphi_{31}}$	5	5	5	4
$\overline{\varphi_{27}}$	5	4	4	4
φ_{41}	5 (2)	4	4	4
$\overline{\varphi_{42}}$	5 (2)	4	5	4
$\overline{\varphi_{47}}$	5 (2)	4	5	4
$\overline{\varphi_{32}}$	5	4	4	4
φ_{46}	5 (2)	4	4	4
φ_{10}	6	5 (2)	5	5
$\overline{\varphi_{33}}$	5	6	6	5
φ_{12}	7	6 (2)	6	6
φ_{11}	8 (2)	7	7	7
$\overline{\varphi_{38}}$	9	8	8	8
$\overline{\varphi_1}$	2	2	2	2
$\overline{\varphi_3}$	3	3	3	3
$\overline{\varphi_{20}}$	2	2	2	2
$\overline{\varphi_5}$	3	3	3	3
$\overline{\varphi_4}$	4	4	4	4
φ_7	3 (2)	3	3	3
φ_{17}	4 (3)	4	4	4
$\overline{\varphi_8}$	3	3	3	3
$\overline{\varphi_{19}}$	3	3	3	3
$\overline{\varphi_9}$	3 (2)	3	3	3
$\overline{\varphi_{18}}$	4	4	4	4
$\overline{\varphi_{22}}$	3	3	3	3
$\overline{\varphi_{10}}$	6	6	6	6
$\overline{\varphi_{24}}$	4 (2)	4	4	4
$\overline{\varphi_{23}}$	4	4	4	4
$\overline{\varphi_{29}}$	4	4	4	4
$\overline{\varphi_{12}}$	7	7	7	7

Table 6. Pruned and Normalised Formulas from the "Dwyer"-set [19]

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_4	$\mathbf{G}(a \vee b \vee \mathbf{G}\overline{b} \vee c\,\mathbf{U}\,b)$
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}	$\overline{a}\,\mathbf{W}\,(a \wedge \overline{b}\,\mathbf{W}\,(b\,\mathbf{W}\,(\overline{b}\,\mathbf{W}\,(b\,\mathbf{W}\,(\mathbf{G}\overline{b})))))$
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_{15}	$a\,\mathbf{W}\,b$
φ_{16}	$\mathbf{G}\overline{a} \vee b\,\mathbf{U}\,(a \vee c)$
φ_{17}	$\mathbf{G}\overline{a} \vee \mathbf{F}(a \wedge b\,\mathbf{W}\,c)$
φ_{18}	$\mathbf{G}(a \vee b \vee \mathbf{G}\overline{b} \vee c\,\mathbf{U}\,(b \vee d))$
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_{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_{25}	$a\,\mathbf{W}\,(a \wedge b \wedge \mathbf{X}(a\,\mathbf{U}\,c))$
φ_{26}	$\mathbf{G}\overline{a} \vee b\,\mathbf{U}\,(a \vee (b \wedge c \wedge \mathbf{X}(b\,\mathbf{U}\,d)))$
φ_{27}	$a\,\mathbf{W}\,(a \vee b\,\mathbf{W}\,(b \wedge c \wedge \mathbf{X}(b\,\mathbf{U}\,d)))$
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}	$\mathbf{G}(a \vee b\,\mathbf{W}\,(c \vee (b \wedge d \wedge \mathbf{X}(b\,\mathbf{U}\,e))))$
φ_{30}	$a\,\mathbf{U}\,b \vee \mathbf{G}(a \vee \mathbf{XG}c)$
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}	$\overline{a}\,\mathbf{W}\,(a \wedge (b\,\mathbf{U}\,c \vee \mathbf{G}(b \vee \mathbf{XG}d)))$
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}	$\mathbf{G}(a \vee \mathbf{G}(b \vee \mathbf{XG}c) \vee (b \vee d \vee \mathbf{X}(d\,\mathbf{R}\,(c \vee d)))\,\mathbf{U}\,(d \vee e))$
φ_{35}	$\mathbf{G}(a \vee \mathbf{XG}\overline{b} \vee \mathbf{XF}(b \wedge \mathbf{F}c))$
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}	$\mathbf{G}(a \vee \mathbf{G}(b \vee \mathbf{XG}\overline{c} \vee \mathbf{XF}(c \wedge \mathbf{F}d)))$
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_{40}	$\mathbf{G}(a \vee \mathbf{F}(b \wedge \mathbf{XF}c))$
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_{42}	$\mathbf{G}(a \vee \mathbf{G}(b \vee (c \wedge \mathbf{XF}d)))$
φ_{43}	$\mathbf{G}(a \vee \mathbf{G}\overline{b} \vee (c \vee \overline{b}\,\mathbf{U}\,(\overline{b} \wedge d \wedge \mathbf{X}(\overline{b}\,\mathbf{U}\,e)))\,\mathbf{U}\,b)$
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_{46}	$\mathbf{G}\overline{a} \vee (b \vee \overline{a}\,\mathbf{U}\,(\overline{a} \wedge c \wedge d \wedge \mathbf{X}((\overline{a} \wedge d)\,\mathbf{U}\,e)))\,\mathbf{U}\,a$
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 7. Remaining Formulas from the Parametrised Set from Table 3

LTL	ltl2dstar (historic)	ltl2dgra (asymmetric)	ltl2dgra (symmetric, this article)	ltl2tgba (portfolio)
φ_{30}	33	65	42	16
φ_{32}	18 (4)	5 (3)	5 (3)	4 (3)
$\overline{\varphi_{29}}$	28 (4)	25	15	8
$\overline{\varphi_{28}}$	15 (3)	9	7	4
φ_{29}	17	25	16	8
φ_{9}	33 (2)	15	11 (2)	12
φ_{12}	56 (3)	42	115	42
φ_{31}	7 (2)	2 (2)	2 (2)	2 (2)
$\overline{\varphi_{9}}$	28 (4)	26 (4)	13	11
$\overline{\varphi_{32}}$	13	5	6	5
φ_{8}	10	5	5 (2)	4
φ_{7}	5	2	2 (2)	2
$\overline{\varphi_{6}}$	7	11	5	5
$\overline{\varphi_{31}}$	5	2	2	2
φ_{28}	9	9	6	4
φ_{11}	24 (3)	18	36	18
$\overline{\varphi_{33}}$	28	19	14	19
$\overline{\varphi_{4}}$	5	3 (2)	3	3
φ_{13}	2	1	1	1
φ_{14}	2	1	1	1
φ_{15}	2	1	1	1
$\overline{\varphi_{13}}$	2	1	1	1
$\overline{\varphi_{14}}$	2	1	1	1
$\overline{\varphi_{15}}$	2	1	1	1
$\overline{\varphi_{1}}$	6	4 (2)	4	4
$\overline{\varphi_{5}}$	6	5 (2)	4	4
φ_{10}	8 (2)	8	11	8
φ_{4}	4	3	3	3
$\overline{\varphi_{2}}$	10	8	8	8
φ_{1}	5	4	4	4
φ_{5}	5	4	4	4
φ_{6}	6	5	5	5
$\overline{\varphi_{8}}$	6 (2)	6 (3)	5	5
$\overline{\varphi_{10}}$	8 (2)	7	7	7
$\overline{\varphi_{3}}$	18	16 (2)	16	16
φ_{2}	9	8	8	8
φ_{3}	17	16	16	16
$\overline{\varphi_{11}}$	18 (3)	17	17	17
$\overline{\varphi_{12}}$	42 (4)	41	41	41
$\overline{\varphi_{7}}$	3 (2)	3	3	3
φ_{27}	`err`	1 (9)	1 (9)	1 (16)

Table 8. Pruned and Normalised Formulas from the Parametrised Set

φ_1	$(a \,\mathbf{U}\, b) \,\mathbf{U}\, c$
φ_2	$((a \,\mathbf{U}\, b) \,\mathbf{U}\, c) \,\mathbf{U}\, d$
φ_3	$(((a \,\mathbf{U}\, b) \,\mathbf{U}\, c) \,\mathbf{U}\, d) \,\mathbf{U}\, e$
φ_4	$a \,\mathbf{U}\, (b \,\mathbf{U}\, c)$
φ_5	$a \,\mathbf{U}\, (b \,\mathbf{U}\, (c \,\mathbf{U}\, d))$
φ_6	$a \,\mathbf{U}\, (b \,\mathbf{U}\, (c \,\mathbf{U}\, (d \,\mathbf{U}\, e)))$
φ_7	$\mathbf{G}(\overline{a} \vee a \,\mathbf{U}\, b)$
φ_8	$\mathbf{G}(\overline{a} \vee a \,\mathbf{U}\, (b \wedge b \,\mathbf{U}\, c))$
φ_9	$\mathbf{G}(\overline{a} \vee a \,\mathbf{U}\, (b \wedge b \,\mathbf{U}\, (c \wedge c \,\mathbf{U}\, d)))$
φ_{10}	$(\mathbf{F}a \vee \mathbf{G}b) \wedge (\mathbf{F}b \vee \mathbf{G}c)$
φ_{11}	$(\mathbf{F}a \vee \mathbf{G}b) \wedge (\mathbf{F}b \vee \mathbf{G}c) \wedge (\mathbf{F}c \vee \mathbf{G}d)$
φ_{12}	$(\mathbf{F}a \vee \mathbf{G}b) \wedge (\mathbf{F}b \vee \mathbf{G}c) \wedge (\mathbf{F}c \vee \mathbf{G}d) \wedge (\mathbf{F}d \vee \mathbf{G}e)$
φ_{13}	$\mathbf{FG}a \wedge \mathbf{FG}b \wedge \mathbf{FG}c$
φ_{14}	$\mathbf{FG}a \wedge \mathbf{FG}b \wedge \mathbf{FG}c \wedge \mathbf{FG}d$
φ_{15}	$\mathbf{FG}a \wedge \mathbf{FG}b \wedge \mathbf{FG}c \wedge \mathbf{FG}d \wedge \mathbf{FG}e$
φ_{16}	$\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{GF}c$
φ_{17}	$\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{GF}c \wedge \mathbf{GF}d$
φ_{18}	$\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{GF}c \wedge \mathbf{GF}d \wedge \mathbf{GF}e$
φ_{19}	$\mathbf{FG}a \vee \mathbf{FG}b \vee \mathbf{GF}c$
φ_{20}	$\mathbf{FG}a \vee \mathbf{FG}b \vee \mathbf{FG}c \vee \mathbf{GF}d$
φ_{21}	$\mathbf{FG}a \vee \mathbf{FG}b \vee \mathbf{FG}c \vee \mathbf{FG}d \vee \mathbf{GF}e$
φ_{22}	$(\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{GF}c) \vee (\mathbf{FG}\overline{c} \wedge (\mathbf{FG}\overline{a} \vee \mathbf{FG}\overline{b}))$
φ_{23}	$(\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{GF}c \wedge \mathbf{GF}d) \vee (\mathbf{FG}\overline{c} \wedge (\mathbf{FG}\overline{a} \vee \mathbf{FG}\overline{b} \vee \mathbf{FG}\overline{d}))$
φ_{24}	$(\mathbf{GF}a \wedge \mathbf{GF}b \wedge \mathbf{GF}c \wedge \mathbf{GF}d \wedge \mathbf{GF}e) \vee (\mathbf{FG}\overline{c} \wedge (\mathbf{FG}\overline{a} \vee \mathbf{FG}\overline{b} \vee \mathbf{FG}\overline{d} \vee \mathbf{FG}\overline{e}))$
φ_{25}	$(\mathbf{FG}a \vee \mathbf{GF}b) \wedge (\mathbf{FG}c \vee \mathbf{GF}a)$
φ_{26}	$(\mathbf{FG}a \vee \mathbf{GF}b) \wedge (\mathbf{FG}c \vee \mathbf{GF}a) \wedge (\mathbf{FG}d \vee \mathbf{GF}c)$
φ_{27}	$(\mathbf{FG}a \vee \mathbf{GF}b) \wedge (\mathbf{FG}c \vee \mathbf{GF}a) \wedge (\mathbf{FG}d \vee \mathbf{GF}c) \wedge (\mathbf{FG}e \vee \mathbf{GF}d)$
φ_{28}	$\mathbf{GF}(a \wedge \mathbf{XX}a) \vee \mathbf{GF}(\overline{a} \wedge \mathbf{XX}\overline{a})$
φ_{29}	$\mathbf{GF}(a \wedge \mathbf{XXX}a) \vee \mathbf{GF}(\overline{a} \wedge \mathbf{XXX}\overline{a})$
φ_{30}	$\mathbf{GF}(a \wedge \mathbf{XXXX}a) \vee \mathbf{GF}(\overline{a} \wedge \mathbf{XXXX}\overline{a})$
φ_{31}	$\mathbf{FG}(a \vee b) \vee \mathbf{FG}(\overline{a} \vee \mathbf{X}b)$
φ_{32}	$\mathbf{FG}(a \vee b) \vee \mathbf{FG}(\overline{a} \vee \mathbf{X}b) \vee \mathbf{FG}(a \vee \mathbf{XX}b)$
φ_{33}	$\mathbf{FG}(a \vee b) \vee \mathbf{FG}(\overline{a} \vee \mathbf{X}b) \vee \mathbf{FG}(a \vee \mathbf{XX}b) \vee \mathbf{FG}(\overline{a} \vee \mathbf{XXX}b)$

ACKNOWLEDGMENTS

The authors want to thank (in alphabetical order) Alexandre Duret-Lutz, Orna Kupferman, Benedikt Seidl, the participants of the 2nd Jerusalem Winter School in Computer Science and Engineering on Formal Verification, and the anonymous referees for their helpful comments and remarks.

REFERENCES

[1] Rajeev Alur and Salvatore La Torre. 2004. Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Log.* 5, 1 (2004), 1–25. DOI: https://doi.org/10.1145/963927.963928

[2] Dana Angluin and Dana Fisman. 2020. Regular ω-languages with an informative right congruence. *Inf. Comput.* (2020), 104598. DOI: https://doi.org/10.1016/j.ic.2020.104598

[3] Valentin M. Antimirov. 1996. Partial derivatives of regular expressions and finite automaton constructions. *Theor. Comput. Sci.* 155, 2 (1996), 291–319. DOI: https://doi.org/10.1016/0304-3975(95)00182-4

[4] Tomáš Babiak, František Blahoudek, Alexandre Duret-Lutz, Joachim Klein, Jan Křetínský, David Müller, David Parker, and Jan Strejček. 2015. The Hanoi omega-automata format. In *Proceedings of the International Conference on Computer-Aided Verification*. 479–486. DOI: https://doi.org/10.1007/978-3-319-21690-4_31

[5] Tomáš Babiak, Frantisek Blahoudek, Mojmír Křetínský, and Jan Strejcek. 2013. Effective translation of LTL to deterministic Rabin automata: Beyond the (F, G)-fragment. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis*. 24–39. DOI: https://doi.org/10.1007/978-3-319-02444-8_4

[6] Tomáš Babiak, Mojmír Křetínský, Vojtech Rehák, and Jan Strejcek. 2012. LTL to Büchi automata translation: Fast and more deterministic. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. 95–109. DOI: https://doi.org/10.1007/978-3-642-28756-5_8

[7] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking*. The MIT Press.

[8] Christel Baier, Stefan Kiefer, Joachim Klein, Sascha Klüppelholz, David Müller, and James Worrell. 2016. Markov chains and unambiguous Büchi automata. In *Proceedings of the International Conference on Computer-Aided Verification*. 23–42. DOI: https://doi.org/10.1007/978-3-319-41528-4_2

[9] Frantisek Blahoudek, Alexandre Duret-Lutz, Mikulás Klokocka, Mojmír Křetínský, and Jan Strejcek. 2017. Seminator: A tool for semi-determinization of omega-automata. In *Proceedings of the International Conferences on Logic for Programming, Artificial Intelligence and Reasoning*. 356–367. http://www.easychair.org/publications/paper/340360.

[10] Julian Brunner, Benedikt Seidl, and Salomon Sickert. 2019. A verified and compositional translation of LTL to deterministic Rabin automata. In *Proceedings of the 10th International Conference on Interactive Theorem Proving, (ITP'19) (LIPIcs)*, John Harrison, John O'Leary, and Andrew Tolmach (Eds.), Vol. 141. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 11:1–11:19. DOI: https://doi.org/10.4230/LIPIcs.ITP.2019.11

[11] Janusz A. Brzozowski. 1964. Derivatives of regular expressions. *J. ACM* 11, 4 (1964), 481–494. DOI: https://doi.org/10.1145/321239.321249

[12] Edward Y. Chang, Zohar Manna, and Amir Pnueli. 1992. Characterization of temporal property classes. In *Proceedings of the 19th International Colloquium on Automata, Languages and Programming (ICALP'92) (Lecture Notes in Computer Science)*, Werner Kuich (Ed.), Vol. 623. Springer, 474–486. DOI: https://doi.org/10.1007/3-540-55719-9_97

[13] Krishnendu Chatterjee, Andreas Gaiser, and Jan Křetínský. 2013. Automata with generalized Rabin pairs for probabilistic model checking and LTL synthesis. In *Proceedings of the International Conference on Computer-Aided Verification*. 559–575. DOI: https://doi.org/10.1007/978-3-642-39799-8_37

[14] Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem (Eds.). 2018. *Handbook of Model Checking*. Springer. DOI: https://doi.org/10.1007/978-3-319-10575-8

[15] Costas Courcoubetis and Mihalis Yannakakis. 1995. The complexity of probabilistic verification. *J. ACM* 42, 4 (1995), 857–907. DOI: https://doi.org/10.1145/210332.210339

[16] Jean-Michel Couvreur. 1999. On-the-fly verification of linear temporal logic. In *Proceedings of the World Congress on Formal Methods*. 253–271.

[17] Marco Daniele, Fausto Giunchiglia, and Moshe Y. Vardi. 1999. Improved automata generation for linear temporal logic. In *Proceedings of the International Conference on Computer-Aided Verification*. 249–260.

[18] Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Etienne Renault, and Laurent Xu. 2016. Spot 2.0 - A framework for LTL and ω-automata manipulation. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis*. 122–129. DOI: https://doi.org/10.1007/978-3-319-46520-3_8

[19] Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. 1998. Property specification patterns for finite-state verification. In *Proceedings of the ACM SIGSOFT Workshop on Formal Methods in Software Practice*. 7–15. DOI: https://doi.org/10.1145/298595.298598

[20] Javier Esparza and Jan Křetínský. 2014. From LTL to deterministic automata: A safraless compositional approach. In *Proceedings of the International Conference on Computer-Aided Verification*. 192–208. DOI: https://doi.org/10.1007/978-3-319-08867-9_13

[21] Javier Esparza, Jan Křetínský, Jean-François Raskin, and Salomon Sickert. 2017. From LTL and limit-deterministic Büchi automata to deterministic parity automata. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. 426–442. DOI: https://doi.org/10.1007/978-3-662-54577-5_25

[22] Javier Esparza, Jan Křetínský, and Salomon Sickert. 2016. From LTL to deterministic automata—A safraless compositional approach. *Form. Meth. Syst. Des.* 49, 3 (2016), 219–271. DOI: https://doi.org/10.1007/s10703-016-0259-2

[23] Javier Esparza, Jan Křetínský, and Salomon Sickert. 2018. One theorem to rule them all: A unified translation of LTL into ω-automata. In *Proceedings of the ACM/IEEE Symposium on Logic in Computer Science*. 384–393. DOI: https://doi.org/10.1145/3209108.3209161

[24] Kousha Etessami and Gerard J. Holzmann. 2000. Optimizing Büchi automata. In *Proceedings of the International Conference on Concurrency Theory*. 153–167. DOI: https://doi.org/10.1007/3-540-44618-4_13

[25] Carsten Fritz. 2003. Constructing Büchi automata from linear temporal logic using simulation relations for alternating Büchi automata. In *Proceedings of the International Conference on Implementation and Application of Automata*. 35–48.

[26] Dov Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. 1980. On the temporal analysis of fairness. In *Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'80)*. ACM, New York, NY, 163–173. DOI: https://doi.org/10.1145/567446.567462

[27] Andreas Gaiser, Jan Křetínský, and Javier Esparza. 2012. Rabinizer: Small deterministic automata for LTL(F,G). In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis*. 72–76. DOI: https://doi.org/10.1007/978-3-642-33386-6_7

[28] Paul Gastin and Denis Oddoux. 2001. Fast LTL to Büchi automata translation. In *Proceedings of the International Conference on Computer-Aided Verification*. 53–65. DOI: https://doi.org/10.1007/3-540-44585-4_6

[29] Jaco Geldenhuys and Henri Hansen. 2006. Larger automata and less work for LTL model checking. In *Proceedings of the 13th International SPIN Workshop on Model Checking Software*. 53–70. DOI: https://doi.org/10.1007/11691617_4

[30] Rob Gerth, Doron A. Peled, Moshe Y. Vardi, and Pierre Wolper. 1995. Simple on-the-fly automatic verification of linear temporal logic. In *Proceedings of the 15th IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification*. 3–18.

[31] Dimitra Giannakopoulou and Flavio Lerda. 2002. From states to transitions: Improving translation of LTL formulae to Büchi automata. In *Proceedings of the International Conference on Formal Techniques for Distributed Objects, Components and Systems*. 308–326. DOI: https://doi.org/10.1007/3-540-36135-9_20

[32] Yuri Gurevich and Leo Harrington. 1982. Trees, automata, and games. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber (Eds.). ACM, 60–65. DOI: https://doi.org/10.1145/800070.802177

[33] Ernst Moritz Hahn, Guangyuan Li, Sven Schewe, Andrea Turrini, and Lijun Zhang. 2015. Lazy probabilistic model checking without determinisation. In *Proceedings of the International Conference on Concurrency Theory (LIPIcs)*, Vol. 42. 354–367. DOI: https://doi.org/10.4230/LIPIcs.CONCUR.2015.354

[34] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. 2020. Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In *Proceedings of the 26th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Held as Part of the European Joint Conferences on Theory and Practice of Software. (Lecture Notes in Computer Science)*, Armin Biere and David Parker (Eds.), Vol. 12078. Springer, 306–323. DOI: https://doi.org/10.1007/978-3-030-45190-5_17

[35] Thomas A. Henzinger and Nir Piterman. 2006. Solving games without determinization. In *Proceedings of the Conference on Computer Science Logic (Lecture Notes in Computer Science)*, Vol. 4207. Springer, 395–410. DOI: https://doi.org/10.1007/11874683_26

[36] Swen Jacobs, Roderick Bloem, Maximilien Colange, Peter Faymonville, Bernd Finkbeiner, Ayrat Khalimov, Felix Klein, Michael Luttenberger, Philipp J. Meyer, Thibaud Michaud, Mouhammad Sakr, Salomon Sickert, Leander Tentrup, and Adam Walker. 2019. In *Proceedings of the 5th Reactive Synthesis Competition (SYNTCOMP'18): Benchmarks, Participants & Results. CoRR* abs/1904.07736 (2019).

[37] Simon Jantsch, David Müller, Christel Baier, and Joachim Klein. 2019. From LTL to unambiguous Büchi automata via disambiguation of alternating automata. In *Proceedings of the 3rd World Congress on Formal Methods—The Next 30 Years (Lecture Notes in Computer Science)*, Maurice H. ter Beek, Annabelle McIver, and José N. Oliveira (Eds.), Vol. 11800. Springer, 262–279. DOI: https://doi.org/10.1007/978-3-030-30942-8_17

[38] Dileep Kini and Mahesh Viswanathan. 2015. Limit deterministic and probabilistic automata for LTL \ GU. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. 628–642. DOI: https://doi.org/10.1007/978-3-662-46681-0_57

[39] Dileep Kini and Mahesh Viswanathan. 2017. Optimal translation of LTL to limit deterministic automata. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. 113–129. DOI: https://doi.org/10.1007/978-3-662-54580-5_7

[40] Joachim Klein and Christel Baier. 2006. Experiments with deterministic *omega*-automata for formulas of linear temporal logic. *Theor. Comput. Sci.* 363, 2 (2006), 182–195. DOI: https://doi.org/10.1016/j.tcs.2006.07.022

[41] Zuzana Komárková and Jan Křetínský. 2014. Rabinizer 3: Safraless translation of LTL to small deterministic automata. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis (Lecture Notes in Computer Science)*, Vol. 8837. 235–241. DOI: https://doi.org/10.1007/978-3-319-11936-6_17

[42] Jan Kretínský, Alexander Manta, and Tobias Meggendorfer. 2019. Semantic labelling and learning for parity game solving in LTL synthesis. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis (Lecture Notes in Computer Science)*, Vol. 11781. Springer, 404–422.

[43] Orna Kupferman and Moshe Y. Vardi. 1998. Freedom, weakness, and determinism: From linear-time to branching-time. In *Proceedings of the ACM/IEEE Symposium on Logic in Computer Science*. 81–92. DOI: https://doi.org/10.1109/LICS.1998.705645

[44] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *Proceedings of the International Conference on Computer-Aided Verification (Lecture Notes in Computer Science)*, Vol. 6806. 585–591. DOI: https://doi.org/10.1007/978-3-642-22110-1_47

[45] Jan Křetínský and Javier Esparza. 2012. Deterministic automata for the (F,G)-fragment of LTL. In *Proceedings of the International Conference on Computer-Aided Verification*. 7–22. DOI: https://doi.org/10.1007/978-3-642-31424-7_7

[46] Jan Křetínský and Ruslán Ledesma-Garza. 2013. Rabinizer 2: Small deterministic automata for LTL \ GU. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis*. 446–450. DOI: https://doi.org/10.1007/978-3-319-02444-8_32

[47] Jan Křetínský, Tobias Meggendorfer, and Salomon Sickert. 2018. Owl: A library for ω-words, automata, and LTL. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis*. 543–550. DOI: https://doi.org/10.1007/978-3-030-01090-4_34

[48] Jan Křetínský, Tobias Meggendorfer, Salomon Sickert, and Christopher Ziegler. 2018. Rabinizer 4: From LTL to your favourite deterministic automaton. In *Proceedings of the International Conference on Computer-Aided Verification*. 567–577. DOI: https://doi.org/10.1007/978-3-319-96145-3_30

[49] Jan Křetínský, Tobias Meggendorfer, Clara Waldmann, and Maximilian Weininger. 2017. Index appearance record for transforming Rabin automata into parity automata. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. 443–460. DOI: https://doi.org/10.1007/978-3-662-54577-5_26

[50] Jianwen Li, Geguang Pu, Lijun Zhang, Zheng Wang, Jifeng He, and Kim Guldstrand Larsen. 2013. 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*, Zhiming Liu, Jim Woodcock, and Huibiao Zhu (Eds.). Lecture Notes in Computer Science, Vol. 8051. Springer, 256–270. DOI: https://doi.org/10.1007/978-3-642-39698-4_16

[51] Jianwen Li, Lijun Zhang, Shufang Zhu, Geguang Pu, Moshe Y. Vardi, and Jifeng He. 2018. An explicit transition system construction approach to LTL satisfiability checking. *Form. Asp. Comput.* 30, 2 (2018), 193–217.

[52] Christoph Löding. 1999. *Methods for the Transformation of Automata: Complexity and Connection to Second Order Logic*. Master's thesis. Institute of Computer Science and Applied Mathematics, Christian-Albrechts-University of Kiel, Germany.

[53] Christof Löding and Anton Pirogov. 2019. Determinization of Büchi automata: Unifying the approaches of Safra and Muller-Schupp. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming, (LIPIcs)*, Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.), Vol. 132. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 120:1–120:13. DOI: https://doi.org/10.4230/LIPIcs.ICALP.2019.120

[54] Christof Löding and Anton Pirogov. 2019. New optimizations and heuristics for determinization of Büchi automata. In *Proceedings of the 17th International Symposium on Automated Technology for Verification and Analysis, (Lecture Notes in Computer Science)*, Yu-Fang Chen, Chih-Hong Cheng, and Javier Esparza (Eds.), Vol. 11781. Springer, 317–333. DOI: https://doi.org/10.1007/978-3-030-31784-3_18

[55] Michael Luttenberger, Philipp J. Meyer, and Salomon Sickert. 2020. Practical synthesis of reactive systems from LTL specifications via parity games. *Acta Inform.* 57, 1-2 (2020), 3–36. DOI: https://doi.org/10.1007/s00236-019-00349-3

[56] Zohar Manna and Amir Pnueli. 1990. A hierarchy of temporal properties. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*. 377–410. DOI: https://doi.org/10.1145/93385.93442

[57] Philipp J. Meyer, Salomon Sickert, and Michael Luttenberger. 2018. Strix: Explicit reactive synthesis strikes back! In *Proceedings of the International Conference on Computer-Aided Verification*. 578–586. DOI: https://doi.org/10.1007/978-3-319-96145-3_31

[58] Satoru Miyano and Takeshi Hayashi. 1984. Alternating finite automata on omega-words. *Theor. Comput. Sci.* 32 (1984), 321–330. DOI: https://doi.org/10.1016/0304-3975(84)90049-5

[59] David Müller and Salomon Sickert. 2017. LTL to deterministic Emerson-Lei automata. In *Proceedings of the International Symposium on Games, Automata, Logics, and Formal Verification*. 180–194. DOI: https://doi.org/10.4204/EPTCS.256.13

[60] David E. Muller, Ahmed Saoudi, and Paul E. Schupp. 1988. Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In *Proceedings of the ACM/IEEE Symposium on Logic in Computer Science*. 422–427. DOI: https://doi.org/10.1109/LICS.1988.5139

[61] Nir Piterman. 2006. From nondeterministic Buchi and Streett automata to deterministic parity automata. In *Proceedings of the ACM/IEEE Symposium on Logic in Computer Science*. 255–264. DOI: https://doi.org/10.1109/LICS.2006.28

[62] Nir Piterman. 2007. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Log. Meth. Comput. Sci.* 3, 3 (2007). DOI: https://doi.org/10.2168/LMCS-3(3:5)2007

[63] Amir Pnueli. 1977. The temporal logic of programs. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*. 46–57. DOI: https://doi.org/10.1109/SFCS.1977.32

[64] Amir Pnueli. 1981. The temporal semantics of concurrent programs. *Theor. Comput. Sci.* 13 (1981), 45–60. DOI: https://doi.org/10.1016/0304-3975(81)90110-9

[65] Amir Pnueli and Roni Rosner. 1989. On the synthesis of a reactive module. In *Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. 179–190. DOI: https://doi.org/10.1145/75277.75293

[66] Shmuel Safra. 1988. On the complexity of omega-automata. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*. 319–327. DOI: https://doi.org/10.1109/SFCS.1988.21948

[67] Sven Schewe. 2009. Tighter bounds for the determinisation of Büchi automata. In *Proceedings of the International Conference on Foundations of Software Science and Computer Structures (Lecture Notes in Computer Science)*, Vol. 5504. 167–181. DOI: https://doi.org/10.1007/978-3-642-00596-1_13

[68] Salomon Sickert. 2016. Linear temporal logic. *Arch. Form. Proofs* 2016. Retrieved from https://www.isa-afp.org/entries/LTL.shtml.

[69] Salomon Sickert. 2019. *A Unified Translation of Linear Temporal Logic to ω-Automata*. Ph.D. Dissertation. Technical University of Munich, Germany. Retrieved from http://nbn-resolving.de/urn:nbn:de:bvb:91-diss-20190801-1484932-1-4.

[70] Salomon Sickert. 2020. A Unified Translation of Linear Temporal Logic to ω-Automata: Supplemental Material for the Experimental Evaluation. DOI: https://doi.org/10.5281/zenodo.3972121

[71] Salomon Sickert and Javier Esparza. 2020. An efficient normalisation procedure for linear temporal logic and very weak alternating automata. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller (Eds.). ACM, 831–844. DOI: https://doi.org/10.1145/3373718.3394743

[72] Salomon Sickert, Javier Esparza, Stefan Jaax, and Jan Křetínský. 2016. Limit-deterministic Büchi automata for linear temporal logic. In *Proceedings of the International Conference on Computer-Aided Verification*. 312–332. DOI: https://doi.org/10.1007/978-3-319-41540-6_17

[73] Salomon Sickert and Jan Křetínský. 2016. MoChiBA: Probabilistic LTL model checking using limit-deterministic Büchi automata. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis*. 130–137. DOI: https://doi.org/10.1007/978-3-319-46520-3_9

[74] A. Prasad Sistla. 1994. Safety, liveness and fairness in temporal logic. *Formal Asp. Comput.* 6, 5 (1994), 495–512. DOI: https://doi.org/10.1007/BF01211865

[75] Fabio Somenzi and Roderick Bloem. 2000. Efficient Büchi automata from LTL formulae. In *Proceedings of the International Conference on Computer-Aided Verification*. 248–263. DOI: https://doi.org/10.1007/10722167_21

[76] Martin Sulzmann and Peter Thiemann. 2018. LTL semantic tableaux and alternating ω-automata via linear factors. In *Proceedings of the International Colloquium on Theoretical Aspects of Computing*. 11–34. DOI: https://doi.org/10.1007/978-3-030-02508-3_2

[77] Deian Tabakov, Kristin Y. Rozier, and Moshe Y. Vardi. 2012. Optimized temporal monitors for SystemC. *Form. Meth. Syst. Des.* 41, 3 (2012), 236–268. DOI: https://doi.org/10.1007/s10703-011-0139-8

[78] Moshe Y. Vardi. 1985. Automatic verification of probabilistic concurrent finite-state programs. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*. 327–338. DOI: https://doi.org/10.1109/SFCS.1985.12

[79] Moshe Y. Vardi. 1994. Nontraditional applications of automata theory. In *Proceedings of the International Symposium on Theoretical Aspects of Computer Software*. 575–597. DOI: https://doi.org/10.1007/3-540-57887-0_116

[80] Moshe Y. Vardi and Pierre Wolper. 1986. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings of the Symposium on Logic in Computer Science*. IEEE Computer Society, 332–344.

[81] Moshe Y. Vardi and Pierre Wolper. 1986. Automata-theoretic techniques for modal logics of programs. *J. Comput. Syst. Sci.* 32, 2 (1986), 183–221. DOI: https://doi.org/10.1016/0022-0000(86)90026-7

Received January 2020; revised August 2020; accepted August 2020