An Efficient Normalisation Procedure for Linear Temporal Logic and Very Weak Alternating Automata

Salomon Sickert
Technische Universität München
Germany
s.sickert@tum.de

Javier Esparza
Technische Universität München
Germany
esparza@in.tum.de

Abstract
In the mid 80s, Lichtenstein, Pnueli, and Zuck proved a classical theorem stating that every formula of Past LTL (the extension of LTL with past operators) is equivalent to a formula of the form $\bigwedge_{i=1}^{n} GF\varphi_i \lor FG\psi_i$, where $\varphi_i$ and $\psi_i$ contain only past operators. Some years later, Chang, Manna, and Pnueli built on this result to derive a similar normal form for LTL. Both normalisation procedures have a non-elementary worst-case blow-up, and follow an involved path from formulas to counter-free automata to star-free regular expressions and back to formulas. We improve on both points. We present a direct and purely syntactic normalisation procedure for LTL yielding a normal form, comparable to the one by Chang, Manna, and Pnueli, that has only a single exponential blow-up. As an application, we derive a simple algorithm to translate LTL into deterministic Rabin automata. The algorithm normalises the formula, translates it into a special very weak alternating automaton, and applies a simple determination procedure, valid only for these special automata.

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

Keywords: Linear Temporal Logic, Normal Form, Weak Alternating Automata, Deterministic Automata

1 Introduction
In seminal work carried out in the middle 80s, Lichtenstein, Pnueli, and Zuck investigated Past Linear Temporal Logic (Past LTL), a temporal logic with future and past operators. They proved the classical result stating that every formula is equivalent to another one of the form

$$\bigwedge_{i=1}^{n} GF\varphi_i \lor FG\psi_i$$

(1)

where $\varphi_i$ and $\psi_i$ only contain past operators [8, 25]. Shortly after, Manna and Pnueli introduced the safety-progress hierarchy, containing six classes of properties (Figure 1a), and presented a logical characterisation of each class in terms of syntactic fragments of Past LTL [11, 12]. The class of reactivity properties, placed at the top of the hierarchy, contains all Past LTL properties, and its syntactic characterisation, given by (1), is the class of reactivity formulas.

In the early 90s, LTL (which only has future operators, but is known to be as expressive as Past LTL), became the logic of choice for most model-checking applications. At that time Chang, Manna, and Pnueli showed that the classes of the safety-progress hierarchy also admit syntactic characterisations in terms of LTL fragments [3]. In particular, they proved that every LTL formula is equivalent to another one in which every path of the syntax tree alternates at most once between the “least-fixed-point” operators $U$ and $M$ and the “greatest-fixed-point” operators $W$ and $R$. In the notation introduced in [2], which mimics the definition of the $\Sigma_i$, $\Pi_i$, and $\Delta_i$ classes of the arithmetical and polynomial hierarchies, they proved that every LTL formula is equivalent to a $\Delta_2$-formula.

While these normal forms have had large conceptual impact in model checking, automatic synthesis, and deductive verification (see e.g. [17] for a recent survey), the normalisation procedures proving that they are indeed normal forms have had none. In particular, contrary to the case of propositional or first-order logic, they have not been implemented in tools. The reason is that they are not direct, have high complexity, and their correctness proofs are involved. Let us elaborate on this. In [25], Zuck gives a detailed description of the normalisation procedure of [8]. First, Zuck translates the initial Past LTL formula into a counter-free semi-automaton, then applies the Krohn-Rhodes decomposition and other results to translate the automaton into a
star-free regular expression, and finally translates this expression into a reactivity formula with a non-elementary blow-up. In [11, 12] the procedure is not even presented, the reader is referred to [25] and/or to previous results. The normalisation procedure of [3] for LTL calls the translation procedure of [8, 25] for Past LTL as a subroutine, and so it is not any simpler. Finally, while Maler and Pnueli present in [10] an improved translation of star-free regular languages to Past LTL, their work still leads to a triple exponential normalisation procedure for Past LTL. Further, it is not clear to us if this translation can also be used to obtain $\Delta_2$-formulas.

In this paper we present a novel normalisation procedure that translates any LTL formula into an equivalent $\Delta_2$-formula. Our procedure is:

- **Direct.** It does not require any detour through automata or regular expressions.
- **Syntax-guided.** It consists of a few syntactic rewrite rules—not unlike the rules for putting a boolean formula in conjunctive or disjunctive normal form—that can be described in less than a page.
- **Single exponential.** The length of the $\Delta_2$-formula is at most exponential in the length of the original formula, a dramatic improvement on the previous non-elementary and triple exponential bounds.

The correctness proof of the procedure consists of a few lemmas, all of them with routine proofs by structural induction. It is presented in Sections 4 to 6, modulo the omission of some straightforward induction cases. To make this paper self-contained, the proofs of three lemmas taken from [5, 21] are reproduced in the appendix of the extended version of this paper [23]. We have mechanised the complete correctness proof in Isabelle/HOL [15], building upon previous work [1, 19, 20]. The formalised proof consists of roughly 1000 lines, from which one can extract a formally verified normalisation procedure consisting of ca. 200 lines of Standard ML code, excluding standard definitions added by the code-generation. Both the formalisation and instructions for extracting code are located in [22].

In the second part of the paper (Sections 7 and 8) we use the new normalisation procedure to derive a simple translation of LTL into deterministic Rabin automata (DRW). First, we show that every formula of $\Delta_2$ can be translated into a very weak alternating Büchi automaton (A1W) in which every path has at most one alternation between accepting and non-accepting states. Further, we provide a simple determination procedure for these automata, based on a breakpoint construction. The LTL-to-DRW translation normalises the formula, transforms it into an A1W with at most one alternation, and determinises this intermediate automaton.

Due to space constraints we do not provide an overview of LTL-to-DRW translations and refer the reader to [21, Ch. 1]. Furthermore, we only provide a preliminary experimental evaluation of the proposed translations and leave a detailed analysis as future work.

## 2 Preliminaries

Let $\Sigma$ be a finite alphabet. A *word* $w$ over $\Sigma$ is an infinite sequence of letters $a_0a_1a_2\ldots$ with $a_i \in \Sigma$ for all $i \geq 0$, and a language is a set of words. A *finite word* is a finite sequence of letters. As usual, the set of all words (finite words) is denoted $\Sigma^\omega (\Sigma^*)$. We let $w[i]$ (starting at $i = 0$) denote the $i$-th letter of a word $w$. The finite infix $w[i]w[i+1]\ldots w[j-1]$ is abbreviated with $w_{ij}$ and the infinite suffix $w[i]w[i+1]\ldots$ with $w_i$. We denote the infinite repetition of a finite word $\sigma_1\ldots\sigma_n$ by $(\sigma_1\ldots\sigma_n)^\omega = \sigma_1\ldots\sigma_n\sigma_1\ldots\sigma_n\ldots$

**Definition 1.** LTL formulas over a set $\mathcal{A}$ of atomic propositions are constructed by the following syntax:

$$\varphi ::= \top \mid \bot \mid a \mid \neg a \mid \varphi \land \psi \mid \varphi \lor \psi \mid X\varphi \mid \varphi U \varphi \mid \varphi W \varphi \mid \varphi R \varphi \mid \varphi M \varphi$$

where $a \in \mathcal{A}$ is an atomic proposition and $X, U, W, R,$ and $M$ are the next, (strong) until, weak until, (weak) release, and strong release operators, respectively.

The inclusion of both the strong and weak until operators as well as the negation normal form are essential to our approach. The operators $R$ and $M$, however, are added to ensure that every formula of length $n$ in the standard syntax, with negation but only the until operator, is equivalent to a formula of length $O(n)$ in our syntax. They could be removed, if we accept an exponential blow-up incurred by expressing $R$ with $W$. The semantics is defined as usual:

**Definition 2.** Let $w$ be a word over the alphabet $2^\varphi$ and let $\varphi$ be a formula. The satisfaction relation $w \models \varphi$ is inductively defined as the smallest relation satisfying:

$$w \models \top \quad w \models a \iff a \in w[0] \quad w \models \neg a \iff a \notin w[0] \quad w \models \varphi \land \psi \iff w \models \varphi \land w \models \psi \quad w \models X\varphi \iff w_1 \models \varphi$$

We let $L(\varphi) := \{ w \in (2^\varphi)^\omega : w \models \varphi \}$ denote the language of $\varphi$. We overload the definition of $\models$ and write $\varphi \models \psi$ as a shorthand for $L(\varphi) \subseteq L(\psi)$.

We use the standard abbreviations $F \varphi ::= \top U \varphi$ (eventually) and $G \varphi ::= \bot R \varphi$ (always). Finally, we introduce the
3 The Safety-Progress Hierarchy

We recall the hierarchy of temporal properties studied by Manna and Pnueli [11] following the formulation of Černá and Pelánek [2]. In the ensuing sections we describe structures that have a direct correspondence to this hierarchy and in this sense the hierarchy provides a map to navigate the results of this paper.

Definition 4 ([2, 11]). Let \( P \subseteq \Sigma^* \) be a property over \( \Sigma \).

- \( P \) is a safety property if there exists a language of finite words \( L \subseteq \Sigma^* \) such that for every \( w \in P \) all finite prefixes of \( w \) belong to \( L \).
- \( P \) is a guarantee property if there exists a language of finite words \( L \subseteq \Sigma^* \) such that for every \( w \in P \) there exists a finite prefix of \( w \) which belongs to \( L \).
- \( P \) is an obligation property if it can be expressed as a positive boolean combination of safety and guarantee properties.
- \( P \) is a recurrence property if there exists a language of finite words \( L \subseteq \Sigma^* \) such that for every \( w \in P \) infinitely many prefixes of \( w \) belong to \( L \).
- \( P \) is a persistence property if there exists a language of finite words \( L \subseteq \Sigma^* \) such that for every \( w \in P \) all but finitely many prefixes of \( w \) belong to \( L \).
- \( P \) is a reactivity property if \( P \) can be expressed as a positive boolean combination of recurrence and persistence properties.

The inclusions between these classes are shown in Figure 1a. Chang, Manna, and Pnueli give in [3] a syntactic characterisation of the classes of the safety-progress hierarchy in terms of fragments of LTL. The following is a corollary of the proof of [3, Thm. 8]:

Definition 5 (Adapted from [2]). We define the following classes of LTL formulas:

- The class \( \Sigma_0 = \Pi_0 = \Delta_0 \) is the least set containing all atomic propositions and their negations, and is closed under the application of conjunction and disjunction.
- The class \( \Sigma_{i+1} \) is the least set containing \( \Pi_1 \) and is closed under the application of conjunction, disjunction, and the \( X \), \( U \), and \( M \) operators.
- The class \( \Pi_{i+1} \) is the least set containing \( \Sigma_1 \) and is closed under the application of conjunction, disjunction, and the \( X \), \( R \), and \( W \) operators.
- The class \( \Delta_{i+1} \) is the least set containing \( \Sigma_{i+1} \) and \( \Pi_{i+1} \) and is closed under the application of conjunction and disjunction.

\begin{align*}
\text{reactivity} \subset \text{persistence} \subset \text{obligation} \subset \text{guarantee} & \subset \text{reedom} \subset \text{recurrence} \subset \text{invariance} \\
\Delta_2 & \subset \Pi_2 \subset \Delta_1 \subset \Pi_1 \subset \Delta_0 \subset \Pi_0 \subset \Delta_0
\end{align*}

(a) Safety-progress hierarchy [11] (b) Syntactic-future hierarchy

Figure 1. Both hierarchies, side-by-side, indicating the correspondence of Theorem 6

Theorem 6 (Adapted from [2]). A property that is specifiable in LTL is a guarantee (safety, obligation, persistence, recurrence, reactivity, respectively) property if and only if it is specifiable by a formula from the class \( \Sigma_1 \), \( \Pi_1 \), \( \Delta_1 \), respectively.

4 Overview of the Normalisation Result

Fix an LTL formula \( \varphi \) over a set of atomic propositions \( AP \). Our new normal form is based on two notions:

- A partition of the universe \( U := (2^{|AP|})^\omega \) of all words into equivalence classes of words that, loosely speaking, exhibit the same “limit-behaviour” with respect to \( \varphi \).
- The notion of stable word with respect to \( \varphi \).

A partition of \( U \). Let \( \mu(\varphi) \) and \( \nu(\varphi) \) be the sets containing the subformulas of \( \varphi \) of the form \( \psi_1 \circ \psi_2 \) for \( \varphi \in \{ U, M \} \) and \( \varphi \in \{ W, R \} \), respectively. Given a word \( w \), define:

\[
\mathcal{GF}_w^\varphi := \{ \psi : \psi \in \mu(\varphi) \land w \models GF \psi \} \\
\mathcal{FG}_w^\varphi := \{ \psi : \psi \in \nu(\varphi) \land w \models FG \psi \}
\]

(To simplify the notation, when \( \varphi \) is clear from the context we simply write \( \mathcal{GT}_w \) and \( \mathcal{FG}_w \).) Two words \( w, v \) have the same limit-behaviour w.r.t. \( \varphi \) if \( \mathcal{GT}_w = \mathcal{GT}_v \) and \( \mathcal{FG}_w = \mathcal{FG}_v \). Having the same limit-behaviour is an equivalence relation, which induces the partition \( P = \{ P_{M,N} \subseteq U : M \subseteq \mu(\varphi), N \subseteq \nu(\varphi) \} \) given by:

\[
P_{M,N} := \{ w \in U : M = \mathcal{GT}_w \land N = \mathcal{FG}_w \}
\]

Example 7. Let \( \varphi = Ga \lor bUc \). We have \( \mu(\varphi) = \{ bUc \} \) and \( \nu(\varphi) = \{ Ga \} \). The partition \( P \) has four equivalence classes:

- \( P_{0,0} \) contains all words such that \( bUc \) holds only finitely often and \( Ga \) fails infinitely often (which in this case implies that \( Ga \) never holds), e.g. \( \{ b \}^{<\omega} \) or \( \{ c \} \{ b \}^\omega \).
- \( P_{0,\{Ga\}} \) contains all words such that \( bUc \) holds infinitely often and \( Ga \) fails infinitely often, e.g. \( \{ a \}^{<\omega} \) or \( \{ c \} \{ a \}^\omega \).
- \( P_{1,\{bUc\},0} \) contains all words such that \( bUc \) holds infinitely often and \( Ga \) fails infinitely often, e.g. \( \{ a \} \{ c \}^{<\omega} \) or \( \{ a \} \{ c \}^\omega \).
Further, \( \varphi[M]_{1}^{\Pi} \) and \( \varphi[N]_{1}^{\Sigma} \) are obtained from \( \varphi, M, \) and \( N \) by means of a simple, linear-time syntactic substitution procedure. Observe that the right-hand side is a formula of \( \Delta_{2} \), and that we write \( \equiv^{S_{\varphi}} \), i.e., the equivalence is only valid within the universe of stable words. In this paper we lift this restriction. In Section 6 we define a formula \( \varphi[M]_{1}^{\Sigma} \in \Sigma_{2} \) by means of another linear-time, syntactic substitution procedure, such that:

\[
\varphi \equiv \bigvee_{M \subseteq \mu(\varphi)} \bigg( \varphi[M]_{1}^{\Sigma} \land \bigwedge_{\psi \in M} \text{GF}(\varphi[N]_{1}^{\Sigma}) \land \bigwedge_{\psi \in N} \text{FG}(\varphi[M]_{1}^{\Pi}) \bigg) \tag{5}
\]

Example 9. For \( \varphi = (F(a \land G(b \lor Fc)) \in \Sigma_{3} \), the still-to-be-defined normal form (4) will yield:

\[
\varphi \equiv^{S_{\varphi}} (GFa \land Gb) \lor (GFa \land GFc)
\]

Indeed, since \( \varphi \in \mu(\varphi) \), every stable word satisfying \( \varphi \) must satisfy it infinitely often, and so equivalence for stable words holds, although the formulas are not equivalent. For Equation (5) we will obtain:

\[
\varphi \equiv F(a \land ((b \lor Fc) U Gb)) \lor (Fa \land GFc)
\]

Observe that the right-hand-side belongs to \( \Delta_{2} \).

5 The Formulas \( \varphi[M]_{1}^{\Pi} \) and \( \varphi[N]_{1}^{\Sigma} \)

We recall the definitions of the formulas \( \varphi[M]_{1}^{\Pi} \) and \( \varphi[N]_{1}^{\Sigma} \), introduced in [5, 21] with a slightly different notation.

The formula \( \varphi[M]_{1}^{\Pi} \). Define \( \mathcal{P}_{M} := \bigcup_{N \subseteq \nu(\varphi)} \mathcal{P}_{M,N} \). Observe that \( \mathcal{P}_{M} \) is the language of the words \( w \) such that \( M = G\mathcal{F}_{w} \). The formula \( \varphi[M]_{1}^{\Pi} \) is defined with the goal of satisfying the following identity:

\[
\varphi \equiv^{S_{\varphi} \cap \mathcal{P}_{M}} \varphi[M]_{1}^{\Pi} \tag{6}
\]

Intuitively, the identity states that within the universe of the stable words of \( \mathcal{P}_{M} \), the formula \( \varphi \) can be replaced by the simpler formula \( \varphi[M]_{1}^{\Pi} \).

All insights required to define \( \varphi[M]_{1}^{\Pi} \) are illustrated by the following examples, where we assume that \( w \in S_{\varphi} \land \mathcal{P}_{M} \):

- \( \varphi = Fa \land Gb \) and \( M = \{Fa\} \). Since \( M = G\mathcal{F}_{w} \), we have \( Fa \in G\mathcal{F}_{w} \), which implies \( w \models GFa \). So \( w \models Fa \land Gb \iff w \models Gb \), and so we can set \( \varphi[M]_{1}^{\Pi} := tt \land Gb \), i.e., we can define \( \varphi[M]_{1}^{\Pi} \) as the result of substituting \( tt \) for \( Fa \) in \( \varphi \). The yet-to-be-defined substitution in fact replaces the abbreviation \( Fa = ttu_{a} \) by \( twa \equiv tt \).

- \( \varphi = Fa \land Gb \) and \( M = \emptyset \). Since \( M = \mathcal{F}_{w} \), we have \( Fa \notin \mathcal{F}_{w} \), and so \( w \not\models Fa \). In other words, \( w \models Fa \land Gb \iff w \models Gb \), and so we can set \( \varphi[M]_{1}^{\Pi} := F \land Gb \).

- \( \varphi = G(bUc) \) and \( M = \{bUc\} \). Since \( M = G\mathcal{F}_{w} \), we have \( bUc \in G\mathcal{F}_{w} \), and so \( w \models G(bUc) \). This does not imply \( w \models bUc \) for all suffixes of \( w \), but it implies that \( c \) will hold infinitely often in the future. So
\( w \models G(b \text{Uc}) \text{ iff } w \models G(b \text{Wc}), \) and so we can define \( \varphi[M]_1^{\Pi} \models G(b \text{Wc}). \)

**Definition 10 ([5, 21]).** Let \( M \subseteq \mu(\varphi) \) be a set of formulas. The formula \( \varphi[M]_1^{\Pi} \) is inductively defined as follows:

\[
(\varphi U \psi)[M]_1^{\Pi} :=
\begin{cases}
\varphi[M]_1^{\Pi} \, W \, \psi[M]_1^{\Pi} & \text{if } \varphi U \psi \models M \\
\varphi[M]_1^{\Pi} & \text{otherwise.}
\end{cases}
\]

\[
(\varphi M \psi)[M]_1^{\Pi} :=
\begin{cases}
\varphi[M]_1^{\Pi} \, R \, \psi[M]_1^{\Pi} & \text{if } \varphi M \psi \models M \\
\varphi[M]_1^{\Pi} & \text{otherwise.}
\end{cases}
\]

All other cases are defined homomorphically, e.g., \( \alpha[M]_1^{\Pi} := \alpha \) for every \( \alpha \in Ap, \) \( (X \varphi)[M]_1^{\Pi} := X(\varphi[M]_1^{\Pi}), \) and \( (W \varphi)[M]_1^{\Pi} := (\varphi[M]_1^{\Pi}) \, W \, (\varphi[M]_1^{\Pi}). \)

The following lemma, proved in [5, 21], shows that \( \varphi[M]_1^{\Pi} \) indeed satisfies Equation (6). Since the notation of \([5, 21]\) is slightly different, we include proofs with the new notation for the cited results in the appendix of the extended version of this paper [23] for convenience.

**Lemma 11 ([5, 21]).** Let \( w \) be a word, and let \( M \subseteq \mu(\varphi) \) be a set of formulas.

1. If \( \mathcal{F}_w^\varphi \subseteq M \) and \( w \models \varphi \), then \( w \models \varphi[M]_1^{\Pi}. \)
2. If \( M \subseteq \mathcal{G}_w^\varphi \) and \( w \models \varphi[M]_1^{\Pi}, \) then \( w \models \varphi. \)
3. \( \varphi \equiv_{S_0 \cap \mathcal{P}_N} \varphi[M]_1^{\Pi} \)

Observe that the first two statements do not assume that \( w \) is stable. This is an aspect we will later make use of for the definition of the normalisation procedure.

**The formula \( \varphi[N]_1^\Sigma.** Let \( \mathcal{P}_N := \bigcup_{M \subseteq \mu(\varphi)} \mathcal{P}_{M, N}. \) The formula \( \varphi[N]_1^\Sigma \) is designed to satisfy

\( \varphi \equiv_{S_0 \cap \mathcal{P}_N} \varphi[N]_1^\Sigma \) (7)

and its definition is completely dual to that of \( \varphi[M]_1^{\Pi}. \)

**Definition 12 ([5, 21]).** Let \( N \subseteq v(\varphi) \) be a set of formulas. The formula \( \varphi[N]_1^\Sigma \) is inductively defined as follows:

\[
(\varphi R \psi)[N]_1^\Sigma :=
\begin{cases}
\text{tt} & \text{if } \varphi R \psi \models N \\
\varphi[N]_1^\Sigma \, M \, \psi[N]_1^\Sigma & \text{otherwise.}
\end{cases}
\]

\[
(\varphi W \psi)[N]_1^\Sigma :=
\begin{cases}
\text{tt} & \text{if } \varphi W \psi \models N \\
\varphi[N]_1^\Sigma \, U \, \psi[N]_1^\Sigma & \text{otherwise.}
\end{cases}
\]

All other cases are defined homomorphically.

The dual of Lemma 11 also holds:

**Lemma 13 ([5, 21]).** Let \( w \) be a word, and let \( N \subseteq v(\varphi) \) be a set of formulas.

1. If \( \mathcal{G}_w^\varphi \subseteq N \) and \( w \models \varphi \), then \( w \models \varphi[N]_1^\Sigma. \)
2. If \( N \subseteq \mathcal{G}_w^\varphi \) and \( w \models \varphi[N]_1^\Sigma, \) then \( w \models \varphi. \)
3. \( \varphi \equiv_{S_0 \cap \mathcal{P}_N} \varphi[N]_1^\Sigma \)

**A normal form for stable words.** We use the following result from [5, 21] to characterise the stable words of a partition \( \mathcal{P}_{M, N} \) that satisfy \( \varphi: \)

**Lemma 14 ([5, 21]).** Let \( w \) be a word, and let \( M \subseteq \mu(\varphi) \) and \( N \subseteq v(\varphi). \) Then define:

\[
\Phi(M, N) := \bigwedge_{\varphi \in M} \text{GF}(\varphi[N]_1^\Sigma) \land \bigwedge_{\varphi \in N} \text{FG}(\varphi[M]_1^{\Pi})
\]

We have:

1. If \( M \subseteq \mathcal{G}_w^\varphi \) and \( N \subseteq \mathcal{F}_w^\varphi, \) then \( w \models \Phi(M, N). \)
2. If \( w \models \Phi(M, N), \) then \( M \subseteq \mathcal{G}_w^\varphi \) and \( N \subseteq \mathcal{F}_w^\varphi. \)

Equipped with this lemma, let us show that a stable word of \( \mathcal{P}_{M, N} \) satisfies \( \varphi \) iff it satisfies \( \varphi[M]_1^{\Pi} \land \Phi(M, N). \) Let \( w \) be a stable word of \( \mathcal{P}_{M, N}. \) If \( w \) satisfies \( \varphi, \) then it satisfies \( \varphi[M]_1^{\Pi} \) by Lemma 11.3 and \( \Phi(M, N) \) by Lemma 14.1 (recall that, since \( w \in \mathcal{P}_{M, N}, \) we have \( M \subseteq \mathcal{G}_w^\varphi \) and \( N \subseteq \mathcal{F}_w^\varphi \) by Equation (2)). For the other direction, assume that \( w \) satisfies \( \varphi[M]_1^{\Pi} \land \Phi(M, N). \) Then we have \( M \subseteq \mathcal{G}_w^\varphi \) by Lemma 14.2 and so \( w \) satisfies \( \varphi \) by Lemma 11.2. (This direction does not even require \( \varphi \) to be stable.)

Since every word belongs to some element of the partition, we obtain a normal form for stable words:

**Proposition 15.**

\[
\varphi \equiv_{S_0} \bigwedge_{M \subseteq \mu(\varphi)} \bigwedge_{N \subseteq v(\varphi)} \left( \varphi[M]_1^{\Pi} \land \bigwedge_{\varphi \in M} \text{GF}(\varphi[N]_1^\Sigma) \land \bigwedge_{\varphi \in N} \text{FG}(\varphi[M]_1^{\Pi}) \right)
\]

**Proof.** Define \( \Phi(M, N) \) as in Lemma 14 and let \( w \in S_0 \) be a stable word. We show that \( w \) satisfies \( \varphi \) iff it satisfies \( \varphi[M]_1^{\Pi} \) and \( \Phi(M, N) \) for some \( M \subseteq \mu(\varphi) \) and \( N \subseteq v(\varphi). \)

Assume \( w \models \varphi. \) Let \( M := \mathcal{G}_w^\varphi \) and \( N := \mathcal{F}_w^\varphi. \) By Lemma 14.1 \( w \models \Phi(M, N) \) holds. Since \( w \) is stable, we have \( \mathcal{F}_w = \mathcal{G}_w^\varphi = M \) (see Equation (3)). By Lemma 11.1 we have \( w \models \varphi[M]_1^{\Pi}, \) and we are done.

Assume \( w \models (\varphi[M]_1^{\Pi} \land \Phi(M, N)) \) for some \( M \subseteq \mu(\varphi) \) and \( N \subseteq v(\varphi). \) Using the second part of Lemma 14 we get \( M \subseteq \mathcal{G}_w^\varphi. \) Applying Lemma 11.2 we get \( w \models \varphi. \)

**Example 16.** Let \( \varphi = (\varphi \land G(b \lor Fc)). \) We have \( \mu(\varphi) = \{ \varphi, Fc \} \) and \( v(\varphi) = \{ G(b \lor Fc) \}. \) So there are four possible choices for \( M \) and two for \( N. \) It follows that the right-hand-side of Proposition 15 has eight disjuncts. However, all disjuncts with \( \varphi \notin M \) are equivalent to \( \text{ff} \) because then \( \varphi[M]_1^{\Pi} = \text{ff,} \) and the same holds for all disjuncts with \( \varphi \in M \) and \( N = \emptyset \) because \( \varphi[\emptyset]_1^{\Pi} = \text{ff}. \)

The two remaining disjuncts are \( M_1 = \{ \varphi \}, N_1 = \{ G(b \lor Fc) \}, \) and \( M_2 = \{ \varphi, Fc \}, N_2 = \{ G(b \lor Fc) \}. \) For both we have \( \varphi[M_1]_1^{\Pi} = \varphi[M_2]_1^{\Pi} = \text{tt}. \) Further, for the first disjunct we have

\[
\text{GF}(\varphi[N_1]_1^\Sigma) \land \text{FG}((G(b \lor Fc))[M_1]_1^{\Pi}) = \text{GF}a \land \text{FG}b
\]
and for the second we get
\[ GF(\varphi(N_2)_{1}) \land GF(Fc)(N_2)_{1}) \land FG((G(b \lor Fc))(M_2)_{1}) \]
\[ \equiv GFa \land GFc \land FG(Gtt) \equiv GFa \land GFc . \]

Together we obtain \( F(a \land G(b \lor Fc)) \equiv S_{G} \land GFa \land (FGb \lor GFc) . \)

6 A Normal Form for LTL

Proposition 15 has little interest in itself because of the restriction to stable words. However, it serves as the starting point for our search for an unrestricted normal form, valid for all words. Observe that Lemma 14 does not depend on \( w \) being stable. Contrary to Lemma 11.1 refers to \( FT_{w} \) and we crucially depend on stability to replace it by \( GF_{w} \). Consequently, we only need to find a replacement for the first conjunct and can leave the rest of the structure, i.e. the enumeration of all possible combinations of \( M \subseteq \mu(\varphi) \) and \( \Phi(M, N) \), unchanged. More precisely, we search for a mapping \( \varphi(\cdot) \) that assigns to every \( M \subseteq \mu(\varphi) \) a formula \( \varphi(M) \in \Sigma_{2} \) such that:

\[
\varphi \equiv \bigvee_{M \in \mu(\varphi)} \bigg( \bigg( \bigwedge_{\psi \in M} GF\big(\varphi[N]\big)_{1} \land \bigwedge_{\psi \in N} FG\big(\varphi[M]\big)_{1} \bigg) \bigg) \tag{8}
\]

The following lemma gives sufficient conditions for \( \varphi(M) \).

Lemma 17. For every \( M \subseteq \mu(\varphi) \), let \( \varphi(M) \) be a formula satisfying:

(a) For every \( M' \subseteq \mu(\varphi); M \subseteq M' \implies \varphi(M) \models \varphi(M') \)
(b) For every word \( w; w \models \varphi \implies w \models \varphi(FT_{w}) \)

Then Equation (8) holds.

Proof. Assume that \( (a, b) \) hold, and let \( w \) be a word. We show that \( \varphi \) iff it satisfies the right-hand-side of (8).

(\Rightarrow) Assume \( w \models \varphi \). By (b) we have \( w \models \varphi(FT_{w}) \). We claim that the disjunct of the right-hand-side of Equation (8) with \( M := FT_{w} \) and \( N := \Phi \) holds. Indeed, \( w \models \varphi(M) \) trivially holds, and the rest follows from Lemma 14.1.

(\Leftarrow) Assume \( \varphi \) satisfies the right-hand side of Equation (8). Then there exist \( M \subseteq \mu(\varphi) \) and \( N \subseteq \nu(\varphi) \) such that \( w \models \varphi(M) \) holds, \( w \models GF(\varphi[N])_{1} \) holds for every \( \psi \in M \), and \( w \models FG(\varphi[M])_{1} \) holds for every \( \psi \in N \). Lemma 14.2 yields \( M \subseteq FT_{w} \), and (a) yields \( \varphi(FT_{w}) \). Applying (b) we get \( w \models \varphi \).

Note that Lemma 17 can also be dualised and we could search for a mapping \( \varphi(\cdot) \) that assigns to every \( N \subseteq \nu(\varphi) \) a formula \( \varphi(N) \in \Pi_{1} \) such that Equation (8) holds.

Unfortunately we cannot simply take \( \varphi(M) := \varphi[M]_{1} \) or \( \varphi(N) := \varphi[N]_{1} \); both choices satisfy condition (a) of Lemma 17, as proven by Lemma 18, but fail to satisfy condition (b) as shown by Example 19.

\footnote{This lemma is needed again for the proof of Theorem 23.}

Lemma 18. \( \varphi[1]_{1} \) and \( \varphi[1]_{1} \) have the following properties:
For every \( M, M' \subseteq \mu(\varphi) \) and \( N, N' \subseteq \nu(\varphi) \):

\[ M \subseteq M' \implies \varphi[M]_{1} \models \varphi[M']_{1} \]
\[ N \subseteq N' \implies \varphi[N]_{1} \models \varphi[N']_{1} \]

Proof. (a) By induction on \( \varphi \). We show only two cases, since all other cases are either trivial or analogous.

Case \( \varphi = \psi_{1}U\psi_{2} \). Assume \( w \models \varphi[M]_{1} \) holds. Due to the definition of \( \varphi[M]_{1} \) we have \( \varphi \in M \) and thus also \( \varphi \in M' \). Thus we have \( w \models (\psi_{1}[M]_{1})W(\psi_{2}[M'_{1}] \) and applying the induction hypothesis we get \( w \models (\psi_{1}[M'_{1}]W(\psi_{2}[M'_{1}] \) for every \( (\psi_{1}[M'_{1}]W(\psi_{2}[M'_{1}] \). Hence \( w \models \varphi[M']_{1} \).

Case \( \varphi = \psi_{1}W_{1}\psi_{2} \). Assume \( w \models \varphi[N]_{1} \) holds. If \( \varphi \in N' \) then \( w \models \varphi[N']_{1} \) trivially holds. If \( \varphi \notin N' \) then also \( \varphi \notin N \), and we get \( w \models (\psi_{1}[N]_{1}W_{1}(\psi_{2}[N]_{1}) \). Using the induction hypothesis we get \( w \models (\psi_{1}[N']_{1}U_{1}(\psi_{2}[N']_{1}) \), and we are done.

\( \square \)

Example 19. Let us first exhibit a formula \( \varphi \) and a word \( w \) such that \( w \models \varphi \), but \( w \not\models \varphi[FT_{w}]_{1} \). For this take \( \varphi = Fa \) and \( w = \{a\} \{a\}^{\omega} \). Thus \( w \models \varphi \) and \( FT_{w} = \emptyset \). However, \( Fa[0]_{1} = \emptyset \) and hence \( w \not\models \varphi(FT_{w})_{1} \).

We now move to the second case. Let us exhibit \( \varphi \) and \( w \) such that \( w \not\models \varphi \) but \( w \models \varphi[FT_{w}]_{1} \). Dually, let \( \varphi = Ga \) and \( w = \{a\} \{a\}^{\omega} \). Then \( w \not\models \varphi \), but \( FT_{w} = \{Ga\} \). However, \( (Ga)[\{Ga\]_{1} = \text{ff} \) and hence \( w \models (Ga)[FT_{w}]_{1} \).

The key to finding a mapping \( \varphi(\cdot) \) satisfying both conditions of Lemma 17 is the technical result below, for which we offer the following intuition. The following equivalence is a valid law of LTL:

\[ GF \equiv \varphi \land U \varphi \tag{9} \]

In order to prove that a word \( w \) satisfies the right-hand-side we can take an arbitrary index \( i \geq 0 \), prove that \( w_{j} \models \varphi \) holds for every \( j < i \), and then prove that \( w_{i} \models GF \). Since we are free to choose \( i \), we can pick it such that \( w_{i} \) is a stable word, which allows us to apply the machinery of Section 5 and obtain:

Lemma 20. For every word \( w \):

\[ w \models GF \iff w \models \varphi \land U \varphi[FT_{w}]_{1} \]

Proof. We prove both directions separately.  

(\Rightarrow) Assume \( w \models GF \). Holds. Let \( w_{j} \) be a stable suffix of \( w \). By the definition of stability we have \( FT_{w} = FT_{w_{j}} = GF \Phi \) for every \( j \geq i \). By Lemma 11.1, we have

\[ w_{j} \models \varphi \implies w_{j} \models GF \Phi \Phi \] for every \( j \geq i \)

and so in particular \( w_{i} \models GF \Phi \). We proceed as follows:

\[ \begin{align*}
& w \models GF \\
& \implies w_{i} \models GF \Phi \Phi \\
& \implies w \models U \varphi \land V \Phi \Phi
\end{align*} \]

\[ w \models \varphi \land U \varphi[FT_{w}]_{1} \]
This is an immediate consequence of Lemma 11.2. □

With the help of the standard LTL-equivalences
\[ \varphi W \psi \equiv \varphi U (\varphi \lor G \varphi \psi) \]  
\[ \varphi R \psi \equiv (\varphi \lor G \varphi \psi) M \psi \]

Lemma 20 can be extended to a more powerful proposition.

**Proposition 21.** For all formulas \( \varphi, \psi \), and for every word \( w \):

\[ w \models \varphi W \psi \iff w \models \varphi U (\varphi \lor G(\varphi G T_{w}^{\psi})) \]
\[ w \models \varphi R \psi \iff w \models (\varphi \lor G(\varphi G T_{w}^{\psi})) M \psi \]

**Proof:** We only prove the first statement. The proof of the second is dual.

(⇒) Assume \( w \models \varphi W \psi \). We split this branch of the proof further, by a case distinction on whether \( w \models G \varphi \psi \) holds. If \( w \models G \varphi \psi \) holds, then by Lemma 20 we have \( w \models \varphi U G(\varphi G T_{w}^{\psi}) \), and so \( w \models \varphi U (\varphi \lor G(\varphi G T_{w}^{\psi})) \) holds. Assume now that \( w \not\models G \varphi \psi \). Then we simply derive:

\[ w \models \varphi U \psi \]

\[ \iff w \models G(\varphi G T_{w}^{\psi}) \]

(⇐) By Lemma 11.2 we have \( (w j \models (G \varphi) G T_{w}^{\psi} \implies (w j \models (G \varphi) G T_{w}^{\psi}) \)

for all \( j \geq 0 \). Thus \( w j \models (G \varphi) G T_{w}^{\psi} \implies w j \models G \varphi \) for all \( j \geq 0 \) and we can simply derive:

\[ w \models \varphi U \psi \]

(Proposition 21) □

Lemma 21 gives us all we need to define a formula \( \varphi[M]_{2}^{\psi} \) satisfying Equation (8).

**Definition 22.** Let \( \varphi \) be a formula and let \( M \subseteq \mu(\varphi) \). The formula \( \varphi[M]_{2}^{\psi} \) is inductively defined as follows for \( R \) and \( W \):

\[ (\varphi R \psi)[M]_{2}^{\psi} = (\varphi[M]_{2}^{\psi} \lor G(\varphi[M]_{1}^{\psi})) M \psi[M]_{2}^{\psi} \]
\[ (\varphi W \psi)[M]_{2}^{\psi} = \varphi[M]_{2}^{\psi} U (\varphi[M]_{1}^{\psi} \lor G(\varphi[M]_{1}^{\psi})) \]

and homomorphically for all other cases.

A straightforward induction on \( \varphi \) shows that \( \varphi[M]_{2}^{\psi} \in \Sigma_{2} \), justifying our notation. We prove that \( \varphi[M]_{2}^{\psi} \) satisfies (8) by checking that it satisfies the conditions of Lemma 17.

**Theorem 23.** Let \( \varphi \) be a formula. Then:

\[ \varphi \equiv \bigvee_{M \subseteq \mu(\varphi)} \left( \varphi[M]_{2}^{\psi} \land \bigwedge_{\psi \in M} G(\varphi[N]_{1}^{\psi}) \land \bigwedge_{\psi \in N} F(\varphi[M]_{1}^{\psi}) \right) \]

**Proof:** We show that conditions (a) and (b) of Lemma 17 hold.

(a) The proof is an easy induction on \( \varphi \), applying Lemma 18 where necessary.

(b) We prove that

\[ \forall w. w \models \varphi \iff w \models \varphi[G T_{w}^{\psi}]_{2}^{\psi} \]  

holds by structural induction on \( \varphi \). We make use of the identity

\[ \varphi[M]_{2}^{\psi} = \psi[M \cap \mu(\varphi)]_{2}^{\psi} \]

which follows immediately from the fact that formulas in \( M \setminus \mu(\varphi) \) are not subformulas of \( \psi \).

The base of the induction is \( \varphi \in \{ \texttt{t}, \texttt{f}, a, \ldots \} \). In all these cases we have \( \varphi = \varphi[G T_{w}^{\psi}]_{2}^{\psi} \) by definition, and so (12) holds vacuously. All other cases in which \( \varphi[M]_{2}^{\psi} \) is defined homomorphically are handled in the same way. We consider only one of them:

Case \( \varphi = \psi U \psi \). By assumption, the induction hypothesis (12) holds for \( \psi_{1} \) and \( \psi_{2} \), giving:

\[ \forall u. (u \models \psi_{1} \implies w \models \psi_{1}[G T_{w}^{\psi}]_{2}^{\psi}) \]  

\[ \forall v. (v \models \psi_{2} \implies w \models \psi_{2}[G T_{w}^{\psi}]_{2}^{\psi}) \]

In order to use these two equivalences for the induction step, we need to replace \( G T_{u}^{\psi_{1}} \) and \( G T_{v}^{\psi_{2}} \) by \( G T_{w}^{\psi_{1}} \) in the context of \( \Sigma_{2} \). For this we instantiate \( u \models w \) and \( v \models w \) for arbitrary \( i, j \geq 0 \) in (14) and (15). With this choice \( u \) and \( v \) are suffixes of \( w \), and so thus we get \( G T_{u}^{\psi_{1}} = G T_{w}^{\psi_{1}} = G T_{w}^{\psi_{1}} \). Notice further that, by intersection with \( \mu(\cdot) \), we have \( G T_{u}^{\psi_{1}} = G T_{w}^{\psi_{1}} \cap \mu(\psi_{1}) \) and \( G T_{u}^{\psi_{1}} = G T_{w}^{\psi_{1}} \cap \mu(\psi_{2}) \). From (13) we obtain:

\[ \forall i. (w i \models \psi_{1} \implies w i \models \psi_{1}[G T_{w}^{\psi}]_{2}^{\psi_{1}}) \]  

\[ \forall j. (w j \models \psi_{2} \implies w j \models \psi_{2}[G T_{w}^{\psi}]_{2}^{\psi_{2}}) \]

Applying (16) and (17) we get:

\[ w \models \psi_{1} U \psi_{2} \]

\[ \iff \exists k. w k \models \psi_{1} \land (\forall \ell < k. w \ell \models \psi_{1}) \]

\[ \exists k. w k \models \psi_{2} [G T_{w}^{\psi}]_{2}^{\psi_{2}} \land (\forall \ell < k. w \ell \models \psi_{1}[G T_{w}^{\psi}]_{2}^{\psi_{2}}) \]

\[ w \models (\psi_{1} U \psi_{2})[G T_{w}^{\psi}]_{2}^{\psi_{2}} \]

which concludes the proof.

The remaining cases are \( \varphi = \psi_{1} R \psi_{2} \) and \( \varphi = \psi_{1} W \psi_{2} \). Again, we only consider one of them, the other one being analogous.

Case \( \varphi = \psi_{1} W \psi_{2} \). The argumentation is only slightly more complicated than that of the \( \psi_{1} U \psi_{2} \) case. By induction hypothesis (16) and (17) hold. With the help of Lemma 20 we derive:

\[ w \models \psi_{1} W \psi_{2} \]

\[ \iff w \models \psi_{1} U (\psi_{2} \lor G(\psi_{2}[G T_{w}^{\psi}]_{2}^{\psi_{2}})) \]  

(Proposition 21)

\[ w \models \psi_{1} U (\psi_{2} \lor G(\psi_{2}[G T_{w}^{\psi}]_{2}^{\psi_{2}})) \]

\[ \iff w \models \psi_{1}[G T_{w}^{\psi}]_{2}^{\psi_{2}} U (\psi_{2}[G T_{w}^{\psi}]_{2}^{\psi_{2}} \lor G(\psi_{2}[G T_{w}^{\psi}]_{2}^{\psi_{2}})) \]

(16) and (17)

\[ w \models (\psi_{1} W \psi_{2})[G T_{w}^{\psi}]_{2}^{\psi_{2}} \]

□
Example 24. Let \( \phi = F(a \land G(b \lor Fc)) \). We have \( \mu(\phi) = \{ (a, Fc) \text{ and } (b, Fc) \} \), and so the right-hand-side of Theorem 23 has eight disjuncts. However, contrary to Example 16, we have \( \phi[M]_\psi^2 \neq \emptyset \) for every \( M \subseteq \{ (a, Fc) \} \). Let \( \Phi(M, N) \) be the disjunct for given sets \( M, N \). We consider two cases:

Case \( M \models \emptyset, N \models \emptyset \). In this case \( \Phi(\emptyset, \emptyset) = \phi[\emptyset]_\psi^2 \), because the conjunctions over \( M \) and \( N \) are vacuous. We have:

\[
\Phi(\emptyset, \emptyset) = \phi[\emptyset]_\psi^2
\]

\[
= F(a \land (G(b \lor Fc)[\emptyset]_\psi^2))
\]

\[
= F(a \land ((b \lor Fc) W F)[\emptyset]_\psi^2))
\]

\[
= F(a \land (b \lor Fc)[\emptyset]_\psi^2 U (F F \lor G((b \lor Fc)[\emptyset]_\psi^2))))
\]

\[
= F(a \land ((b \lor Fc) U Gb))
\]

Case \( M \models \{ (a, Fc) \}, N \models \{ (b \lor Fc) \} \). We get:

\[
\phi[M]_\psi^2 = F(a \land ((b \lor Fc)[M]_\psi^2 U (F F \lor G((b \lor Fc)[M]_\psi^2))))
\]

\[
= F(a \lor ((b \lor Fc) U F F \lor G((b \lor Fc)[M]_\psi^2))))
\]

Further, we have \( FG((b \lor Fc)[M]_\psi^2) = FG((b \lor Fc)[N]_\psi^2) = FG((b \lor Fc) = GF((b \lor Fc) = GF((b \lor Fc) \lor (F F \lor G((b \lor Fc)[M]_\psi^2))))
\]

Repeating this process for all possible sets \( M, N \) and bringing the resulting formula in disjunctive normal form we finally get

\[\phi \equiv F(a \lor ((b \lor Fc) U F F \lor G((b \lor Fc)[M]_\psi^2)))) \lor (Fa \land GFc)\]

6.1 Complexity of the Normalisation Procedure

We show that the normalisation procedure has at most single exponential blowup in the length of the formula, improving on the previously known non-elementary bound.

Proposition 25. Let \( \phi \) be a formula with length \( n \). Then there exists an equivalent formula \( \phi_{\Delta_2} \) in \( \Delta_2 \) of length \( 2^{n+O(1)} \).

Proof. Let \( \psi \) be an arbitrary formula. We let \( |\psi| \) denote the length of formula and start by giving bounds on \( \psi[M]_\psi^1 \), \( \psi[N]_\psi^1 \), and \( \psi[M]_\psi^2 \). For this let \( M \subseteq \mu(\psi) \) and \( N \subseteq \nu(\psi) \) be sets of formulas. We obtain by induction on the structure of \( \psi \) that \( |\psi[M]_\psi^1| \leq |\psi| \), \( |\psi[N]_\psi^1| \leq |\psi| \), and \( |\psi[M]_\psi^2| \leq 2^{|\psi|+1} \).

Consider now the right-hand side of Theorem 23 as the postulated \( \phi_{\Delta_2} \). Using these bounds we calculate the maximal size of a disjunct and obtain:

\[2n+1 + n(n + 3) + n(n + 3) + 1 = 2n+1 + 2n^2 + 6n + 1\]

For sufficiently large \( n \), i.e. \( n > 5 \), we can bound this by \( 2n^2+2\). There exist at most \( 2^n \) disjuncts and thus the formula is at most of size \( 2^{n+2} \) for \( n > 5 \).

6.2 A Dual Normal Form

We obtained Theorem 23 by relying on the LTL equivalence (10) and (11) for \( W \) and \( R \). Using dual LTL-equivalences for \( U \) and \( M \), \( \phi U \psi \equiv (\phi \land F \psi) W \psi \) and \( \phi M \psi \equiv \phi R(\psi \land F \phi) \), we can also obtain a dual normalisation procedure:

Definition 26. Let \( \phi \) be a formula and let \( N \subseteq \nu(\psi) \) be a set of formulas. The formula \( \phi[N]_\psi^1 \) is inductively defined as follows for \( U \) and \( M \):

\[
(\phi U \psi)[N]_\psi^2 = (\phi[N]_\psi^1 \land F(\psi[N]_\psi^1)) \ W \psi[N]_\psi^1
\]

\[
(\phi M \psi)[N]_\psi^2 = \phi[N]_\psi^1 R(\psi[N]_\psi^1 \land F(\psi[N]_\psi^1))
\]

and homomorphically for all other cases.

Theorem 27. Let \( \phi \) be a formula. Then:

\[
\phi \equiv \bigvee_{M \subseteq \mu(\phi)} \left( \phi[N]_\psi^1 \land \bigwedge_{\psi \in M} GF(\psi[N]_\psi^2) \land \bigwedge_{\psi \in N} GF(\psi[N]_\psi^1) \right)
\]

7 A Translation from LTL to Deterministic Rabin Automata (DRW)

We apply our \( \Delta_2 \)-normalisation procedure to derive a new translation from LTL to DRW via weak alternating automata (AWW). While the previously existing normalisation procedures could also be used to translate LTL into DRW, the resulting DRW could have non-elementary size in the length of the formula, making them impractical. We show that, thanks to the single exponential blow-up of the new procedure, the new translation has double exponential blow-up, which is asymptotically optimal.

It is well-known [14, 24] that an LTL formula \( \phi \) of length \( n \) can be translated into an AWW with \( O(n) \) states. We show that, if \( \phi \) is in normal form, i.e., a disjunction as in Theorem 23, then the AWW can be chosen so that every path through the automaton switches at most once between accepting and non-accepting states. We then prove that determinising AWWs satisfying this additional property is much simpler than the general case.

The section is structured as follows: Section 7.1 introduces basic definitions, Section 7.2 shows how to translate an \( \Delta_2 \)-formula into AWWs with at most one switch, and Section 7.3 presents the determinisation procedure for this subclass of AWWs.

7.1 Weak and Very Weak Alternating Automata

Let \( X \) be a finite set. The set of positive Boolean formulas over \( X \), denoted by \( B^+(X) \), is the closure of \( X \cup \{ \tt, \ff \} \) under disjunction and conjunction. A set \( S \subseteq X \) is a model of \( \theta \in B^+(X) \) if the truth assignment that assigns true to the elements of \( S \) and false to the elements of \( X \setminus S \) satisfies \( \theta \). Observe, that if \( S \) is a model of \( \theta \) and \( S \subseteq S' \) then \( S' \) is also a model. A model \( S \) is minimal if no proper subset of \( S \) is a model. The set of minimal models is denoted \( M_\theta \). Two formulas are equivalent, denoted \( \theta \equiv \theta' \), if their set of minimal models is equal, i.e., \( M_\theta = M_{\theta'} \).
Alternating automata. An alternating Büchi word automaton over an alphabet $\Sigma$ is a tuple $\mathcal{A} = (\Sigma, Q, \theta_0, \delta, \alpha)$, where $Q$ is a finite set of states, $\theta_0 \in B^+(Q)$ is an initial formula, $\delta : Q \times \Sigma \to B^+(Q)$ is the transition function, and $\alpha \subseteq Q$ is the acceptance condition. A run of $\mathcal{A}$ on the word $w$ is a directed acyclic graph $G = (V, E)$ satisfying the following properties:

- $V \subseteq Q \times \mathbb{N}_0$, and $E = \bigcup_{l \geq 0}((Q \times \{l\}) \times (Q \times \{l + 1\}))$.
- There exists a minimal model $S$ of $\theta_0$ such that $(q, 0) \in V$ iff $q \in S$.
- For every $(q, l) \in V$, either $\delta(q, w[l]) \equiv \text{ff}$ or the set $\{q' : ((q, l), (q', l + 1)) \in E\}$ is a minimal model of $\delta(q, w[l])$.
- For every $(q, l) \in V \setminus (Q \times \{0\})$ there exists $q' \in Q$ such that $((q', l - 1), (q, l)) \in E$.

Runs can be finite or infinite. A run $G$ is accepting if

(a) $\delta(q, w[l]) \not\equiv \text{ff}$ for every $(q, l) \in V$, and
(b) every infinite path of $G$ visits $a$-nodes (that is, nodes $(q, l)$ such that $q \in \alpha$) infinitely often.

In particular, every finite run satisfying (a) is accepting. $\mathcal{A}$ accepts a word $w$ iff it has an accepting run $G$ on $w$. The language $L(\mathcal{A})$ recognised by $\mathcal{A}$ is the set of words accepted by $\mathcal{A}$. Two automata are equivalent if they recognise the same language.

Alternating co-Büchi automata are defined analogously, changing condition (b) by the co-Büchi condition (every infinite path of $G$ visits $a$-nodes finitely often). Finally, in alternating Rabin automata $\alpha$ is a set of Rabin pairs $(F, I) \subseteq Q \times Q$, and (b) is replaced by the Rabin condition (there exists a Rabin pair $(F, I) \in \alpha$ such that every infinite path visits states of $F$ only finitely often and states of $I$ infinitely often).

An automaton is deterministic if for every state $q \in Q$ and every letter $a \in \Sigma$ there exists $q' \in Q$ such that $\delta(q, a) = q'$, and non-deterministic if for every $q \in Q$ and every $a \in \Sigma$ there exists $Q' \subseteq Q$ such that $\delta(q, a) = \bigvee q' \in Q'$. The following definitions are useful for reasoning about runs: A set $U \subseteq Q$ is called a level. If $U \subseteq \alpha$, then $U$ is an $\alpha$-level. A level $U'$ is a successor of $U$ w.r.t. $\alpha$, also called $\alpha$-successor, if for every $q \in U'$ there is a minimal model $S_q$ of $\delta(q, a)$ such that $U' = \bigcup_{q \in U'} S_q$. The $k$-th level of a run $G = (V, E)$ is the set $\{q : (q, k) \in V\}$. Observe that a level can be empty, and empty levels are $\alpha$-levels. Further, by definition a level has no successors w.r.t. $\alpha$ iff it contains a state $q$ such that $\delta(q, a) \equiv \text{ff}$. In particular, every level of an accepting run has at least one successor.

Weak and very weak automata. Let $\mathcal{A} = (\Sigma, Q, \theta_0, \delta, \alpha)$ be an alternating (co-)Büchi automaton. We write $q \longrightarrow q'$ if there is $a \in \Sigma$ such that $q'$ belongs to some minimal model of $\delta(q, a)$. $\mathcal{A}$ is weak if there is a partition $Q_1, \ldots, Q_m$ of $Q$ such that

- for every $q, q' \in Q$, if $q \longrightarrow q'$ then there are $i \leq j$ such that $q \in Q_i$ and $q' \in Q_j$, and
- for every $1 \leq i \leq m$: $Q_i \subseteq \alpha$ or $Q_i \cap \alpha = \emptyset$.

$\mathcal{A}$ is very weak or linear if it is weak and every class $Q_i$ of the partition is a singleton (|$Q_i|$ = 1). We let AWW and A1W denote the set of weak and very weak alternating automata, respectively. Observe that for every weak automaton with a co-Büchi acceptance condition we can define a Büchi acceptance condition on the same structure recognising the same language. Thus we will from now on assume that every weak automaton is equipped with a Büchi acceptance condition.

We define the height of a weak alternating automaton. The definition is very similar, but not identical, to the one of [6]. A weak automaton $\mathcal{A}$ has height $n$ if every path $q \rightarrow q' \rightarrow q'' \cdots$ of $\mathcal{A}$ alternates at most $n - 1$ times between $\alpha$ and $Q \setminus \alpha$. For example, the automaton in Figure 3 has height 3. We let AWW[$n$] (A1W[$n$]) denote the sets of all (very)-weak alternating automata with height at most $n$. Further, we let AWW[$n$, $\mathcal{A}$] (resp. AWW[$n$, $\mathcal{R}$]) denote the set of automata of AWW[$n$] whose initial formula satisfies $\theta_0 \in \mathcal{B}(\alpha)^+$ (resp. $\theta_0 \in \mathcal{B}(Q \setminus \alpha)^+$). For example the automaton of Figure 3 belongs to A1W[3, $\mathcal{R}$].

7.2 Translation of LTL to A1W[2]

In the standard translation [24] of LTL to A1W, the states of the A1W for a formula $\varphi$ are subformulas of $\varphi$, or negations thereof. We show that, at the price of a slightly more complicated translation, the resulting A1W for a $\Delta_1$-formula belongs to A1W[1]. Thus by using Theorem 23 every LTL formula can be translated to an A1W[2]. The idea of the construction is to use subformulas as states ensuring that

1. the transition relation can only lead from a formula to another formula at the same level or a lower level in the syntactic-future hierarchy (Figure 1b), and
2. accepting states are $\Pi_1$ subformulas.

This immediately leads to “at most one alternation”. However, there is a little technical problem: the level of a formula is not always well-defined, because some formulas do not belong to one single lowest level of the hierarchy. For
example, \( X_a \) belongs to both \( \Pi_1 \) and \( \Sigma_1 \). So we need a mechanism to disambiguate these states. Formally we proceed as follows:

A formula is proper if it is neither a Boolean constant (tt, ff) nor a conjunction or disjunction. A state in our modified translation is an expression of the form \( \langle \psi \rangle_{\Gamma} \), where \( \psi \) is a proper formula, and \( \Gamma \) is a smallest class of the syntactic-future hierarchy without the zeroth-level (Definition 5) that contains \( \psi \). Hence we start with the classes \( \Sigma_1 \) and \( \Pi_1 \) and \( \Gamma \) lies strictly above \( \Lambda_0 \). Observe that for some formulas there is more than one smallest class. For example, since \( X_a \in \Sigma_1 \cap \Pi_1 \), both \( \Sigma_1 \) and \( \Pi_1 \) are smallest classes containing \( X_a \), and so both \( \langle X_a \rangle_{\Sigma_1} \) and \( \langle X_a \rangle_{\Pi_1} \) are states. For other formulas the class is unique. For example, the only state for \( aWb \) is \( \langle aWb \rangle_{\Pi_1} \).

We assign to every formula \( \psi \) of LTL and every class \( \Gamma \) a Boolean combination of states, denoted \([\psi]_{\Gamma}\), as follows:

- \([tt]_{\Sigma} = tt\) and \([ff]_{\Sigma} = ff\).
- \([\psi_1 \lor \psi_2]_{\Gamma} = [\psi_1]_{\Gamma} \lor [\psi_2]_{\Gamma}\).
- \([\psi_1 \land \psi_2]_{\Gamma} = [\psi_1]_{\Gamma} \land [\psi_2]_{\Gamma}\).
- If \( \psi \) is a proper formula, then \([\psi]_{\Gamma} = \lor_{\Gamma} [\psi]_{\Gamma'}\) where \( \Gamma' \leq \Gamma \) means that \( \Gamma' = \Gamma \) or \( \Gamma' \) is below \( \Gamma \).

For example, we obtain \([X_a]_\Sigma \subseteq [\psi]_{\Sigma} \forall [X_a]_{\Pi_1} \) and \([X_a]_\Sigma \subseteq [\psi]_{\Pi_1} \). Moreover, \([Fa]_{\Pi_1} \) if \( \varphi \), since there is no \( \Gamma' \leq \Pi_1 \) such that \( Fa \in \Gamma' \).

Let \( \varphi \in \Lambda_i \) for some \( i \geq 0 \), and let \( sf(\varphi) \) be the set of proper subformulas of \( \varphi \). The automaton \( A_\varphi \) is defined as follows:

\[ Q = \{ \langle \psi \rangle_{\Gamma} : \psi \in sf(\varphi), \Gamma \leq \Lambda_i \} \]

\[ \emptyset = \{ \langle \psi \rangle_{\Pi_1} : \psi \in Q \} \]

\[ \alpha = \{ \langle \psi \rangle_{\Pi_1} : \varphi \in Q \} \]

\[ \delta = \text{the restriction to } Q \times \Sigma \text{ of the function } \delta : B^1(Q) \times \Sigma \rightarrow \Sigma \text{ } (notice that we overload } \delta) \text{ defined inductively as follows:} \]

\[ \delta((a)_{\Gamma}, \sigma) = \text{if } a \in \sigma \text{ then } tt \text{ else } ff \]

\[ \delta((-a)_{\Gamma}, \sigma) = \text{if } a \notin \sigma \text{ then } tt \text{ else } ff \]

\[ \delta((X\varphi)_{\Gamma}, \sigma) = [\varphi]_{\Gamma} \]

\[ \delta((\varphi U \psi)_{\Gamma}, \sigma) = \delta([\varphi \land (\varphi \land X(\varphi U \psi))]_{\Sigma}, \sigma) \]

\[ \delta((\varphi W \psi)_{\Gamma}, \sigma) = \delta([\varphi \land (\varphi \land X(\varphi W \psi))]_{\Sigma}, \sigma) \]

\[ \delta((\varphi R \psi)_{\Gamma}, \sigma) = \delta([\varphi \land (\varphi \land X(\varphi R \psi))]_{\Sigma}, \sigma) \]

\[ \delta((\varphi M \psi)_{\Gamma}, \sigma) = \delta([\varphi \land (\varphi \land X(\varphi M \psi))]_{\Sigma}, \sigma) \]

All other cases (tt, ff, \&, \land, and \lor) are defined homomorphically. Observe that the \( \Gamma' \)-bound for the U, W, R, and M cases suffice, since every \( \Gamma \) is closed under conjunction, disjunction and application of X

An example of this construction is displayed in Figure 3. The states are labelled \( q_0 = \langle \varphi \rangle_{\Sigma_2}, q_1 = \langle G(b \lor XFc) \rangle_{\Pi_1} \), and \( q_2 = \langle Fc \rangle_{\Sigma_1} \).

**Lemma 28.** Let \( \varphi \) be a formula of \( \Lambda_i \). The automaton \( A_\varphi \) belongs to A1W[1], has \( |sf(\varphi)| \) states, and recognises \( L(\varphi) \).

**Proof.** Let us first show that \( A_\varphi \) belongs to A1W[1]. It follows immediately from the definition of \( A_\varphi \) that for every two states \( \langle \psi_1 \rangle_{\Gamma} \) and \( \langle \psi_2 \rangle_{\Gamma} \) of \( A_\varphi \), if \( \langle \psi_1 \rangle_{\Gamma} \rightarrow \langle \psi_2 \rangle_{\Gamma} \) then \( \Gamma' \leq \Gamma \). So in every path there are at most \((i-1)\) alternations between \( \Sigma \) and \( \Pi \) classes. Since the states of \( \alpha \) are those annotated with \( \Pi \) classes, there are also at most \((i-1)\) alternations between \( \alpha \) and non-\( \alpha \) states in a path.

To show that \( A_\varphi \) has at most \( |sf(\varphi)| \) states, observe that for every formula \( \psi \) there are all at most two smallest classes of the syntactic-future hierarchy containing \( \psi \). So \( A_\varphi \) has at most two states for each formula of \( sf(\varphi) \).

To prove that \( A_\varphi \) recognises \( L(\varphi) \) one shows by induction on \( \psi \) that \( A_\varphi \) recognises \( L(\psi) \) from every Boolean combination of states \( [\psi]_{\Sigma} \) such that \( \psi \in \Gamma \). The proof is completely analogous to the one appearing in [24].

**7.3 Determinisation of AWW[2]**

We present a determinisation procedure for AWW[2] and AWW[2][A] inspired by the break-point construction from [13]. We only describe the construction for AWW[2][R], as the one for AWW[2][A] is dual. The following lemma states the key property of AWW[2][R]:

**Lemma 29.** Let \( \mathcal{A} \) be an AWW[2][R]. \( \mathcal{A} \) accepts a word \( w \) if and only if there exists a run \( G = (V, E) \) of \( \mathcal{A} \) on \( w \) such that

\[ \delta(q, w[l]) \neq ff \text{ for every } (q, l) \in V, \]

\[ \text{there is a threshold } k \geq 0 \text{ such that for every } l \geq k \text{ and for every node } (q, l) \in V \text{ the state } q \text{ is accepting.} \]

**Proof.** Assume that \( \mathcal{A} \) accepts \( w \). Let \( G = (V, E) \) be an accepting run of \( \mathcal{A} \) on \( w \). Since \( \mathcal{A} \) is an AWW[2][R], every path has by definition at most one alternation of accepting and rejecting states and all states occurring in the initial formula are marked as rejecting. Hence if a node \( (q, l) \in V \) is accepting, i.e. \( q = a \), then all its descendants are accepting. Let \( V_r \subseteq V \) be the set of rejecting nodes of \( V \), i.e., the nodes \( (q, l) \in V \) such that \( q \neq a \). Since the descendants of accepting nodes are accepting, the subgraph \( G_r = (V_r, E \cap (V_r \times V_r)) \) is acyclic and connected. If \( V_r \) is infinite, then by König's lemma \( G_r \) has an infinite path of non-accepting nodes, contradicting that \( G \) is an accepting run. So \( G_r \) is finite, and we can choose the threshold \( k \) as the largest level of a node of \( V_r \), plus one.

Assume such a run \( G = (V, E) \) exists. Condition (a) of an accepting run holds by hypothesis. For condition (b), just observe that, since the descendants of accepting nodes are accepting, and every infinite path of \( G \) contains a node of the form \( (q, k) \), where \( k \) is the threshold level, every infinite path visits accepting nodes infinitely often.

However, Lemma 29 does not hold for AWW[3][R]:

**Example 30.** Let \( \mathcal{A} \) be the automaton shown in Figure 3 and let \( w = \{a\} \{\{b\} \{c\}\}^{\omega} \). Observe that \( \mathcal{A} \) accepts \( w \). We prove by contradiction that no run of \( \mathcal{A} \) on \( w \) satisfies the properties described in Lemma 29. Assume such a run exists. By the
definition of $\delta$, the run must be infinite. Further, by assumption there exists a threshold $k$ such that all successor levels of the run are exactly $\{q_1\}$. But there exists $k' > k$ such that $w[k'] = \{c\}$. Since $\delta(q_1, \{c\}) = q_1 \wedge q_2$, the $(k' + 1)$-th level of the run contains $q_2$. Contradiction.

Given an automaton $A$ from AWW[2,R], we construct a deterministic co-Büchi automaton $D$ such that $L(A) = L(D)$. A state of the DCW $D$ is a pair $(\text{Levels}, \text{Promising})$, where $\text{Levels} \subseteq 2^Q$ and $\text{Promising} \subseteq 2^a \cap \text{Levels}$. It follows that $D$ has at most $3^{2^n}$ states. Intuitively, after reading a finite word $w_{0k} = a_0 \ldots a_k$ the automaton $D$ is in the state $(\text{Levels}_k, \text{Promising}_k)$, where $\text{Levels}_k$ contains the $k$-th levels of every run of $A$ on all words with $w_{0k}$ as prefix, and $\text{Promising}_k \subseteq \text{Levels}_k$ contains the $a$-levels of $\text{Levels}_k$ that can still "generate" an accepting run. For this, when $D$ reads $a_{k+1}$, it moves from $(\text{Levels}_k, \text{Promising}_k)$ to $(\text{Levels}_{k+1}, \text{Promising}_{k+1})$, where $\text{Levels}_{k+1}$ contains the successors w.r.t. $a_{k+1}$ of $\text{Levels}_k$, and $\text{Promising}_{k+1}$ is defined as follows:

- If $\text{Promising}_k \neq \emptyset$, then $\text{Promising}_{k+1}$ contains the successors w.r.t. $a_{k+1}$ of $\text{Promising}_k$,
- If $\text{Promising}_k = \emptyset$, then $\text{Promising}_{k+1}$ contains the $a$-levels of $\text{Levels}_{k+1}$.

Finally, the co-Büchi condition contains the states $(\text{Levels}, \text{Promising})$ such that $\text{Promising} = \emptyset$.

Intuitively, during its run on a word $w$, the automaton $D$ tracks the promising levels, removing those without successors, because they can no longer produce an accepting run. If the promising set becomes empty infinitely often, then every run of $A$ on $w$ contains a level without successors, and so $A$ does not accept $w$. If after some number of steps, say $k$, the promising set never becomes empty again, then $A$ has a run on $w$ such that every level is an $a$-level and has at least one successor, and so this run is accepting.

For the formal definition of $D$ it is convenient to identify subsets of $2^Q$ and $2^a$ with formulas of $B^+(Q), B^+(a)$ (i.e., we identify a formula and its set of models). Further, we lift $\delta: Q \times X \times B(Q)^+ \rightarrow B(a)^+$ to $\delta': B^+(Q) \times X \times B^+(a)$ in the canonical way. Finally, given $q \in B(Q)$ and $S \subseteq Q$, we let $q[ff/S]$ denote the result of substituting $ff$ for every state of $Q \setminus a$ in $\delta(q, a)$. With these notations, the deterministic Büchi automaton equivalent to $A$ can be described in four lines:

$$
\begin{align*}
D &= (\Sigma, Q', q'_0, \delta', a'),
Q' &= B^+(Q) \times B^+(a),
q'_0 &= (\theta_0, ff),
a' = \{((\delta', ff) : \delta \in B^+(a))\},
\delta'(q, p, a) &= \begin{cases} 
(\delta(q, a), \delta(p, a)) & \text{if } p \neq ff \\
(\delta(q, a), \delta(q, a)[ff/Q \setminus a]) & \text{otherwise.}
\end{cases}
\end{align*}
$$

**Lemma 31.** For every $A \in \text{AWW}[2, R]$ with $n$ states, the deterministic co-Büchi automaton $D$ defined above satisfies $L(A) = L(D)$, and has $3^{2^n}$ states. Dually, for every $A' \in \text{AWW}[2, A]$ with $n'$ states, there exists a deterministic Büchi automaton $D'$ that has $3^{2^{n'}}$ states and that satisfies $L(A') = L(D')$.

**Proof.** Assume $w$ is accepted by $A$. Let $G = (V, E)$ be an accepting run of $A$ on $w$. By Lemma 29 there exists an index $k$ such that all levels of $G$ after the $k$-th one are contained in $a$ and have at least one successor. Therefore, the run $(\text{Levels}_0, \text{Promising}_0), (\text{Levels}_1, \text{Promising}_1), \ldots$ of $D$ on $w$ satisfies $\text{Promising}_i \neq \emptyset$ for almost all $i$, and so $D$ accepts.

Assume $w$ is accepted by $D$. Let $(\text{Levels}_0, \text{Promising}_0), (\text{Levels}_1, \text{Promising}_1), \ldots$ be the run of $D$ on $w$. By definition, there is a $k \geq 0$ such that $\text{Promising}_i \neq \emptyset$ for every $i \geq k$. Choose levels $U_0, U_1, \ldots, U_k$ such that

- $U_k \in \text{Promising}_k$, and
- for every $1 \leq i \leq k$, choose $U_{i-1}$ as a predecessor of $U_i$ (this is always possible by the definition of $\delta'$).

Further, for every $i \geq k$ choose $U_{i+1}$ as a successor of $U_i$. Now, let $G = (V, E)$ be the graph given by

- for every $i \geq 0$, $(q, l) \in V$ iff $q \in U_i$; and
- $\langle(q, l), (q', l+1)\rangle \in E$ iff $q \in U_i$ and $q' \in S_q$, where $S_q$ is the minimal model of $\delta(q, w[l])$ used in the definition of successor level.

It follows immediately from the definitions that $G$ is an accepting run of $A$. The second part is proven by complementing $A'$, applying the just described construction, and replacing the co-Büchi condition by a Büchi condition. □

This result leads to a determinisation procedure for AWW[2].

**Lemma 32.** For every $A = (Q, \Sigma, \theta_0, \delta, \alpha) \in \text{AWW}[2]$ with $n = |Q|$ states and $m = |M_{\theta_0}|$ minimal models of $\theta_0$ there exists an equivalent deterministic Rabin automaton $D$ with $2^{2^{2n+6m+2}}$ states and with $m$ Rabin pairs.

**Proof.** Let $A = (Q, \Sigma, \theta_0, \delta, \alpha)$. Given $Q' \subseteq Q$, let $A_{Q'}$ be the AWW[2] obtained from $A$ by substituting $\bigwedge q \in Q' q$ for the initial formula $\theta_0$. We claim that for each minimal model $S \in M_{\theta_0}$ we can construct a deterministic Rabin automaton (DRW) $D_S$ with at most $2^{2^{2n+6}}$ states and a single Rabin pair, recognising the same language as $A_S$. Let us first see how to construct $D_S$, assuming the claim holds. By the claim we have $\mathcal{L}(A) = \bigcup_{S \in M_{\theta_0}} \mathcal{L}(A_S)$. We can define $D$ as the union of all the automata $D_S$. Recall that given two DRWs with $n_1$,$n_2$ states and $p_1,p_2$ Rabin pairs we can construct a DRW for the union of their languages with $n_1 \times n_2$ states and $n_1 + n_2$ pairs. Since $\theta_0$ has $m$ models, $D$ has at most $m$ Rabin pairs and $2^{2^{2n+6m+2}}$ states.

It remains to prove the claim. Partition $S$ into $S \cap a$ and $S' \alpha$. We have $A_{S'\alpha} \in \text{AWW}[2, A]$ and $A_{S\alpha'} \in \text{AWW}[2, R]$. By Lemma 31 there exists a deterministic Büchi automaton $D_{S'\alpha}$ and a deterministic co-Büchi automaton $D_{S\alpha'}$ equivalent to $A_{S'\alpha}$ and $A_{S\alpha'}$, respectively, both with at most $3^{2^n}$ states. Intersecting these two automata yields a deterministic Rabin automaton with at most $3^{2^{2n+6}}$ states and a single Rabin pair, and we are done. □
7.4 Translation of LTL to DRW

We combine the normalisation procedure and the translation of LTL to A1W of the previous section to obtain for every formula of LTL an equivalent DRW of double exponential size. Given a formula \( \varphi \) we have:

\[
\varphi_{M,N} = \left( \varphi[M]_{2}^{T} \land \bigwedge_{\psi \in M} \left( GF(\psi[N]_{2}^{T}) \land \bigwedge_{\psi \in N} FG(\psi[M]_{2}^{T}) \right) \right)
\]

Using the results of Section 7.2, we translate each formula \( \varphi_{M,N} \) to an A1W[2], and then, applying the normalisation algorithm of Section 7.3, to a DRW. Finally, using the well-known union operation for DRWs, we obtain a DRW for \( \varphi \).

In order to bound the number of states of the final DRW, we first need to determine the number of states of the A1W for each \( \varphi_{M,N} \).

Lemma 33. Let \( \varphi \) be a formula. For every \( M \subseteq \mu(\varphi) \) and \( N \subseteq \nu(\varphi) \), there exists an A1W[2] with \( O(|sf(\varphi)|) \) states that recognises \( \varphi_{M,N} \).

Proof. By Lemma 28, some A1W[2] with \( O(|sf(\varphi_{M,N})|) \) states recognises \( \varphi_{M,N} \). So it suffices to show that \( |sf(\varphi_{M,N})| \in O(|sf(\varphi)|) \), which follows from these claims, proved in the appendix of the extended version of this paper [23]:

1. \( |\cup \{ sf(\psi[N]_{2}^{T}) : \psi \in \varphi \} | \leq |sf(\varphi)| \);
2. \( |\cup \{ sf(\psi[M]_{2}^{T}) : \psi \in \varphi \} | \leq |sf(\varphi)| \);
3. \( |sf(\varphi[M]_{2}^{T})| \leq 3|sf(\varphi)| \).

Proposition 34. Let \( \varphi \) be a formula with \( n \) proper subformulas. There exists a deterministic Rabin automaton recognising \( \mathcal{L}(\varphi) \) with \( 2^{2^{O(n)}} \) states and \( 2^{n} \) Rabin pairs.

Proof. By Lemma 33 the set \( sf(\varphi_{M,N}) \) has at most \( O(n) \) elements for every \( M,N \). Further, due to Lemma 28 the automaton \( \mathcal{A}_{\varphi_{M,N}} \) belongs to A1W[2] and has at most \( O(n) \) states. Applying the construction of Lemma 32 we obtain a DRW with \( 2^{2^{O(n)}} \) states and a single Rabin pair. Using the union operation for DRWs we obtain a DRW for \( \varphi \) with \( 2^{2^{O(n)}} 2^{2^{O(n)}} = 2^{2^{2^{O(n)}}} \) states.

Remark 35. The construction of Lemma 31 is close to Miyano and Hayashi’s translation of alternating automata to non-deterministic automata [13], and to Schneider’s translation of \( \Sigma_{2} \) formulas to deterministic co-Büchi automata [18, p.219], all based on the break-point idea.

7.5 Determinisation of Lower Classes

We now determinise AWW[1]. A deterministic automaton is terminal-accepting if all states are rejecting except a single accepting sink with a self-loop, and terminal-rejecting if all states are accepting except a single rejecting sink with a self-loop. It is easy to see that terminal-accepting and terminal-rejecting deterministic automata are closed under union and intersection. When applied to AWW[1, A], the construction of Lemma 31, yields automata whose states have a trivial Promising set (either the empty set or the complete level). Further, the successor of an \( \alpha \)-level is also an \( \alpha \)-level. From these observations we easily get:

Corollary 36. Let \( \mathcal{A} \) be an automaton with \( n \) states.

- If \( \mathcal{A} \in \text{AWW}[1, \mathbb{R}] \) (resp. \( \mathcal{A} \in \text{AWW}[1, \mathbb{A}] \)), then there exists a deterministic terminal-accepting (resp. terminal-rejecting) automaton recognising \( \mathcal{L}(\mathcal{A}) \) with \( 2^{2^{n}} \) states.
- If \( \mathcal{A} \in \text{AWW}[1] \), then there exists deterministic weak automaton recognising \( \mathcal{L}(\mathcal{A}) \) with \( 2^{2^{n+\log_{2}|M|}+1} \) states.

7.6 Preliminary Experimental Evaluation

We expect the LTL-to-DRW translation of this paper to produce automata similar in size (number of states, Rabin pairs) to the translations presented in [5, 21], which have been implemented using Owl [7] and have been extensively tested. Indeed, the “Master Theorem” of [5, 21] characterises the words satisfying a formula \( \varphi \) as those for which there exist sets \( M, N \) of subformulas satisfying three conditions, and so it has the same rough structure as our normal form. Further, for each disjoint of our normal form the automata constructions used in [5, 21] and the ones used in this paper are similar. Finally, in preliminary experiments we have compared the LTL-to-DRW translations from [21] and a prototype implementation, without optimisations, of the normalisation procedure of this paper. As benchmark sets we used the “Dwyer”-patterns [4], pre-processed as described in [21, Ch. 8], and the “Parametrised” formula set from [21, Ch. 8]. We observed that on the first set for 60% of the formulas the number of states of the resulting DRWs was equal, for 17% the number of states obtained using the construction of this paper was smaller, and for 23% the number of states was larger. On the second set the ratios were: 76% equal, 21% smaller, and 3% larger. For both sets combined we observed that in 85% of all 164 cases the difference in number of states was less than or equal to three.

We concluded that the main advantage of our translation is not its performance, but its modularity (it splits the procedure into a normalisation and a simplified translation phase) and its suitability for symbolic automata constructions. We leave a detailed experimental comparison and possible integration in Owl [7] (which in particular requires to examine different options for formula and automata simplification, as well as an extensive comparison to existing translations) for future work.

8 A Hierarchy of Alternating Weak and Very Weak Automata

The expressive power of weak and very weak alternating automata has been studied by Gurumurthy et al. in [6] and
by Pelánek and Strejček in [16], respectively. Both papers identify the number of alternations between accepting and non-accepting states as an important parameter, and define a hierarchy of automata classes based on it. Let \( \text{AWW}_G[k] \) denote the class of AWW with at most \( k-1 \) alternations defined in [6]. Similarly, let \( \text{A1W}_P[k, A] \) and \( \text{A1W}_P[k, R] \) denote the classes of A1W with at most \( k-1 \) alternations and accepting or non-accepting initial state, respectively, defined in [16]. Finally, define \( \text{A1W}_P[k] = \text{A1W}_P[k, A] \cup \text{A1W}_P[k, R] \). Figure 4 shows the results of [6] and [16]. We abuse language, and, for example, write \( \Pi_2 = \text{A1W}_P[2, A] \) to denote that the class of languages satisfying formulas in \( \Pi_2 \) and the class of languages recognised by automata in \( \text{A1W}_P[2, A] \) coincide.

Unfortunately, the results of [6] and [16] do not “match”. Due to slight differences in the definitions of height, e.g. the treatment of \( \delta(\cdot) = \emptyset \) and \( \delta(\cdot) = \text{tt} \), the restriction to very weak automata of \( \text{AWW}_G[k] \) does not match any class \( \text{A1W}_P[k'] \) (that is, \( \text{AWW}_G[k] \cap \text{A1W} \neq \text{A1W}_P[k'] \)) and, vice versa, extending \( \text{A1W}_P[k] \) does not yield any \( \text{AWW}_G[k'] \). We show that our new definition of height unifies the two hierarchies, yielding the pleasant result shown in Figure 5. The result follows from Lemmas 28, 31 and 32, Corollary 36, and from constructions appearing in [6, 9, 16].

**Proposition 37.** \( \text{AWW}[2] = \omega\text{-regular} \), \( \text{AWW}[2, A] = \text{DBW} \), \( \text{AWW}[2, R] = \text{DCW} \), \( \text{AWW}[1] = \text{DWW} \), \( \text{AWW}[1, A] = \text{SAFETY} \), \( \text{AWW}[1, R] = \text{CO-SAFETY} \). \( \text{A1W}[1, R] = \Sigma_1 \), \( \text{A1W}[1, A] = \Pi_1 \), \( \text{A1W}[1] = \Delta_1 \), \( \text{A1W}[2, R] = \Sigma_2 \), \( \text{A1W}[2, A] = \Pi_2 \), \( \text{A1W}[2] = \Delta_2 \).

Moreover, our single exponential normalisation procedure for LTL transfers to a single exponential normalisation procedure for A1W:

**Lemma 38.** Let \( \mathcal{A} \) be an A1W with \( n \) states over an alphabet with \( m \) letters. There exists \( \mathcal{A}' \in \text{A1W}[2] \) with \( 2^{O(nm)} \) states such that \( L(\mathcal{A}) = L(\mathcal{A}') \).

**Proof.** The translation from A1W to LTL used in Proposition 37 (an adaption of [9]) yields a formula \( \chi_{\mathcal{A}} \) with at most \( O(mn) \) proper subformulas. Applying our normalisation procedure to \( \chi_{\mathcal{A}} \) yields an equivalent formula in \( \Delta_2 \) with at most \( 2^{O(mn)} \) proper subformulas (Lemma 33). Applying Lemma 28 we obtain the postulated automaton \( \mathcal{A}' \). \( \Box \)

## 9 Conclusion

We have presented a purely syntactic normalisation procedure for LTL that transforms a given formula into an equivalent formula in \( \Delta_2 \), i.e., a formula with at most one alternation between least- and greatest-fixpoint operators. The procedure has single exponential blow-up, improving on the prohibitive non-elementary cost of previous constructions. The much better complexity of the new procedure (recall that normalisation procedures for CNF and DNF are also exponential) makes it attractive for its implementation and use in tools. We have presented a first promising application, namely a novel translation from LTL to DRW with double exponential blow-up. Finally, we have shown that the normalisation procedure for LTL can be transferred to a normalisation procedure for very weak alternating automata.

Currently we do not know if our normalisation procedure is asymptotically optimal. We conjecture that this is the case. For the translation of A1W to AWW[2] we also have no further insight, besides the straightforward double exponential upper bound.

## Acknowledgments

The authors want to thank Orna Kupferman for the suggestion to examine the expressive power of weak alternating automata and the anonymous reviewers for their helpful comments and remarks.

This work is partly funded by the German Research Foundation (DFG) project “Verified Model Checkers” (SI7422601) and partly funded by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme under grant agreement PaVeS (No 787-367).
