LTL Semantic Tableaux and Alternating ω-automata via Linear Factors

Martin Sulzmann[1] and Peter Thiemann[2]

[1] Karlsruhe University of Applied Sciences
martin.sulzmann@hs-karlsruhe.de
[2] Faculty of Engineering, University of Freiburg
thiemann@acm.org

Abstract. Linear Temporal Logic (LTL) is a widely used specification framework for linear time properties of systems. The standard approach for verifying such properties is by transforming LTL formulae to suitable ω-automata and then applying model checking. We revisit Vardi's transformation of an LTL formula to an alternating ω-automaton and Wolper's LTL tableau method for satisfiability checking. We observe that both constructions effectively rely on a decomposition of formulae into *linear factors*. Linear factors have been introduced previously by Antimirov in the context of regular expressions. We establish the notion of linear factors for LTL and verify essential properties such as expansion and finiteness. Our results shed new insights on the connection between the construction of alternating ω-automata and semantic tableaux.

1 Introduction

Linear Temporal Logic (LTL) is a widely used specification framework for linear time properties of systems. An LTL formula describes a property of an infinite trace of a system. Besides the usual logical operators, LTL supports the temporal operators $\bigcirc \varphi$ (φ holds in the next step of the trace) and $\varphi \mathbf{U} \psi$ (φ holds for all steps in the trace until ψ becomes true). LTL can describe many relevant safety and liveness properties.

The standard approach to verify a system against an LTL formula is model checking. To this end, the verifier translates a formula into a suitable ω-automaton, for example, a Büchi automaton or an alternating automaton, and applies the model checking algorithm to the system and the automaton. This kind of translation is widely studied because it is the enabling technology for model checking [22,19,18]. Significant effort is spent on developing translations that generate (mostly) deterministic automata or that minimize the number of states in the generated automata [6,2]. Any improvement in these dimensions is valuable as it speeds up the model checking algorithm.

Our paper presents a new approach to understanding and proving the correctness of Vardi's construction of alternating automata (AA) from LTL formulae [17]. Our approach is based on a novel adaptation to LTL of linear factors, a concept from Antimirov's construction of partial derivatives of regular expressions [1]. Interestingly, a similar construction yields a new explanation for Wolper's construction of semantic tableaux [21] for checking satisfiability of LTL formulae. Thus, we can establish that both constructions are effectively isomorphic to each other.

The paper contains the following contributions.

– Definition of linear factors and partial derivatives for LTL formulae (Section 3). We establish their properties and prove correctness.
– Transformation from LTL to AA based on linear factors. The resulting transformation is essentially the standard LTL to AA transformation [17]; it is correct by construction of the linear factors (Section 4).
– Construction of semantic tableaux to determine satisfiability of LTL formulae using linear factors (Section 5). Our method corresponds closely to Wolper's construction and comes with a free correctness proof.

The online version of this paper contains an appendix with all proofs. [3]

1.1 Preliminaries

We write $\omega = \{0, 1, 2, \dots\}$ for the set of natural numbers and Σ^ω for the set of infinite words over alphabet Σ with symbols ranged over by $x, y \in \Sigma$. We regard a word $\sigma \in \Sigma^\omega$ as a map and write σ_i ($i \in \omega$) for the i-th symbol. For $n \in \omega$, we write $\sigma[n \dots]$ for the suffix of σ starting at n, that is, the function $i \mapsto \sigma_{n+i}$, for $i \in \omega$. We write $x\sigma$ for prepending symbol x to σ, that is, $(x\sigma)_0 = x$ and $(x\sigma)_{i+1} = \sigma_i$, for all $i \in \omega$. The notation $\mathbb{P}(X)$ denotes the power set of X.

2 Linear Temporal Logic

Linear temporal logic (LTL) [12] enhances propositional logic with the temporal operators $\bigcirc \varphi$ (φ will be true in the next step) and $\varphi \mathbf{U} \psi$ (φ holds until ψ becomes true). LTL formulae φ, ψ are defined accordingly where we draw atomic propositions p, q from a finite set $\mathbf{AP}$.

Definition 1 (Syntax of LTL).

$$\varphi, \psi ::= p \mid \mathbf{tt} \mid \neg\varphi \mid \varphi \wedge \psi \mid \bigcirc \varphi \mid \varphi \mathbf{U} \psi$$

We apply standard precedence rules to parse a formula ($\neg$, $\bigcirc \varphi$, and other prefix operators bind strongest; then $\varphi \mathbf{U} \psi$ and the upcoming $\varphi \mathbf{R} \psi$ operator; then conjunction and finally disjunction with the weakest binding strength; as the latter are associative, we do not group their operands explicitly). We use parentheses to deviate from precedence or to emphasize grouping of subformulae.

A model of an LTL formula is an infinite word $\sigma \in \Sigma^\omega$ where Σ is a finite set of symbols and there is an interpretation $\mathcal{I} : \Sigma \to \mathbb{P}(\mathbf{AP})$ that maps a symbol $x \in \Sigma$ to the finite set of atomic predicates that are true for x.

[3] `https://arxiv.org/abs/1710.06678`

Definition 2 (Semantics of LTL). *The formula φ holds on word $\sigma \in \Sigma^\omega$ as defined by the judgment $\sigma \models \varphi$.*

$$\begin{aligned}
\sigma \models p &\Leftrightarrow p \in \mathcal{I}(\sigma_0) \\
\sigma \models \mathbf{tt}& \\
\sigma \models \neg\varphi &\Leftrightarrow \sigma \not\models \varphi \\
\sigma \models \varphi \wedge \psi &\Leftrightarrow \sigma \models \varphi \text{ and } \sigma \models \psi \\
\sigma \models \bigcirc \varphi &\Leftrightarrow \sigma[1 \ldots] \models \varphi \\
\sigma \models \varphi \,\mathbf{U}\, \psi &\Leftrightarrow \exists n \in \omega, (\forall j \in \omega, j < n \Rightarrow \sigma[j \ldots] \models \varphi) \text{ and } \sigma[n \ldots] \models \psi
\end{aligned}$$

We say φ is satisfiable *if there exists $\sigma \in \Sigma^\omega$ such that $\sigma \models \varphi$.*

Definition 3 (StandardDerived LTL Operators).

$$\begin{aligned}
\mathbf{ff} &= \neg\mathbf{tt} & \\
\varphi \vee \psi &= \neg(\neg\varphi \wedge \neg\psi) & \textit{disjunction} \\
\varphi \,\mathbf{R}\, \psi &= \neg(\neg\varphi \,\mathbf{U}\, \neg\psi) & \textit{release} \\
\Diamond \psi &= \mathbf{tt} \,\mathbf{U}\, \psi & \textit{eventually/finally} \\
\Box \psi &= \mathbf{ff} \,\mathbf{R}\, \psi & \textit{always/globally}
\end{aligned}$$

For many purposes, it is advantageous to restrict LTL formulae to positive normal form (PNF). In PNF negation only occurs adjacent to atomic propositions. Using the derived operators, all negations can be pushed inside by using the de Morgan laws. Thanks to the release operator, this transformation runs in linear time and space. The resulting grammar of formulae in PNF is as follows.

Definition 4 (Positive Normal Form).

$$\varphi, \psi ::= p \mid \neg p \mid \mathbf{tt} \mid \mathbf{ff} \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \bigcirc \varphi \mid \varphi \,\mathbf{U}\, \psi \mid \varphi \,\mathbf{R}\, \psi$$

From now on, we assume that all LTL formulae are in PNF.

We make use of several standard equivalences in LTL.

Theorem 1 (Standard results about LTL).

1. $\bigcirc (\varphi \wedge \psi) \Leftrightarrow (\bigcirc \varphi) \wedge (\bigcirc \psi)$
2. $\bigcirc (\varphi \vee \psi) \Leftrightarrow (\bigcirc \varphi) \vee (\bigcirc \psi)$
3. $\varphi \,\mathbf{U}\, \psi \Leftrightarrow \psi \vee (\varphi \wedge \bigcirc (\varphi \,\mathbf{U}\, \psi))$
4. $\varphi \,\mathbf{R}\, \psi \Leftrightarrow \psi \wedge (\varphi \vee \bigcirc (\varphi \,\mathbf{R}\, \psi))$

We also make use of the direct definition of a model for the release operation.

Lemma 1. *$\sigma \models \varphi \,\mathbf{R}\, \psi$ is equivalent to one of the following:*

$\forall n \in \omega, (\sigma[n \ldots] \models \psi \text{ or } \exists j \in \omega, ((j < n) \wedge \sigma[j \ldots] \models \varphi))$
$\forall n \in \omega, \sigma[n \ldots] \models \psi \text{ or } \exists j \in \omega, \sigma[j \ldots] \models \varphi \text{ and } \forall i \in \omega, i \leq j \Rightarrow \sigma[i \ldots] \models \psi.$

3 Linear Factors and Partial Derivatives

Antimirov [1] defines a linear factor of a regular expression as a pair of an input symbol and a next formula (regular expression). The analogue for LTL is a pair $\langle \mu, \varphi \rangle$ where μ is a propositional formula in monomial form (no modalities, see Definition 5) that models the set of first symbols whereas φ is a formal conjunction of temporal LTL formulae for the rest of the input. Informally, $\langle \mu, \varphi \rangle$ corresponds to $\mu \land \bigcirc \varphi$. A formula always gives rise to a set of linear factors, which is interpreted as their disjunction.

Definition 5 (Temporal Formulae, Literals and Monomials). *A* temporal formula *does not start with a conjunction or a disjunction.*

A literal ℓ *of* $\mathbf{AP}$ *is an element of* $\mathbf{AP} \cup \neg\mathbf{AP}$. *Negation of negative literals is defined by* $\neg(\neg p) = p$.

A monomial μ, ν *is either* $\mathbf{ff}$ *or a set of literals of* $\mathbf{AP}$ *such that* $\ell \in \mu$ *implies* $\neg \ell \notin \mu$. *The formula associated with a monomial* μ *is given by*

$$\Theta(\mu) = \begin{cases} \mathbf{ff} & \mu = \mathbf{ff} \\ \bigwedge \mu & \mu \textit{ is a set of literals.} \end{cases}$$

In particular, if $\mu = \emptyset$, then $\Theta(\mu) = \mathbf{tt}$. Hence, we may write $\mathbf{tt}$ for the empty-set monomial. As a monomial is represented either by $\mathbf{ff}$ or by a set of non-contradictory literals, its representation is unique.

We define a smart conjunction operator on monomials that retains the monomial structure.

Definition 6. *Smart conjunction on monomials is defined as their union unless their conjunction* $\Theta(\mu) \land \Theta(\nu)$ *is equivalent to* $\mathbf{ff}$*.*

$$\mu \odot \nu = \begin{cases} \mathbf{ff} & \mu = \mathbf{ff} \lor \nu = \mathbf{ff} \\ \mathbf{ff} & \exists \ell \in \mu \cup \nu.\ \neg\ell \in \mu \cup \nu \\ \mu \cup \nu & \textit{otherwise.} \end{cases}$$

Smart conjunction of monomials is correct in the sense that it produces results equivalent to the conjunction of the associated formulae.

Lemma 2. $\Theta(\mu) \land \Theta(\nu) \Leftrightarrow \Theta(\mu \odot \nu)$.

We define an operator $\mathcal{T}$ that transforms propositional formulae consisting of literals and temporal subformulae into sets of conjunctions. We assume that conjunction $\land$ simplifies formulae to normal form using associativity, commutativity, and idempotence. The normal form relies on a total ordering of formulae derived from an (arbitrary, fixed) total ordering on atomic propositions.

Definition 7 (Set-Based Conjunctive Normal Form).

$$\begin{aligned} \mathcal{T}(\varphi \land \psi) &= \{\varphi' \land \psi' \mid \varphi' \in \mathcal{T}(\varphi), \psi' \in \mathcal{T}(\psi)\} \\ \mathcal{T}(\varphi \lor \psi) &= \mathcal{T}(\varphi) \cup \mathcal{T}(\psi) \\ \mathcal{T}(\varphi) &= \{\varphi\} && \textit{if } \varphi \textit{ is a temporal formula} \end{aligned}$$

Lemma 3. $\bigvee(\mathcal{T}\varphi) \Leftrightarrow \varphi$.

Definition 8 (Linear Factors). *The set of* linear factors $\textsc{lf}(\varphi)$ of an LTL formula *in PNF is defined as a set of pairs of a monomial and a PNF formula in conjunctive normal form.*

$$\begin{array}{ll}
\textsc{lf}(\ell) & = \{\langle\{\ell\}, \mathbf{tt}\rangle\} \\
\textsc{lf}(\mathbf{tt}) & = \{\langle\mathbf{tt}, \mathbf{tt}\rangle\} \\
\textsc{lf}(\mathbf{ff}) & = \{\} \\
\textsc{lf}(\varphi \vee \psi) & = \textsc{lf}(\varphi) \cup \textsc{lf}(\psi) \\
\textsc{lf}(\varphi \wedge \psi) & = \{\langle\mu', \varphi' \wedge \psi'\rangle \mid \langle\mu, \varphi'\rangle \in \textsc{lf}(\varphi), \langle\nu, \psi'\rangle \in \textsc{lf}(\psi), \mu' = \mu \odot \nu \neq \mathbf{ff}\} \\
\textsc{lf}(\bigcirc \varphi) & = \{\langle\mathbf{tt}, \varphi'\rangle \mid \varphi' \in \mathcal{T}(\varphi)\} \\
\textsc{lf}(\varphi \,\mathbf{U}\, \psi) & = \textsc{lf}(\psi) \cup \{\langle\mu, \varphi' \wedge \varphi \,\mathbf{U}\, \psi\rangle \mid \langle\mu, \varphi'\rangle \in \textsc{lf}(\varphi)\} \\
\textsc{lf}(\varphi \,\mathbf{R}\, \psi) & = \{\langle\mu', \varphi' \wedge \psi'\rangle \mid \langle\mu, \varphi'\rangle \in \textsc{lf}(\varphi), \langle\nu, \psi'\rangle \in \textsc{lf}(\psi), \mu' = \mu \odot \nu \neq \mathbf{ff}\} \\
& \quad \cup \{\langle\nu, \psi' \wedge \varphi \,\mathbf{R}\, \psi\rangle \mid \langle\nu, \psi'\rangle \in \textsc{lf}(\psi)\}
\end{array}$$

By construction, the first component of a linear factor is never $\mathbf{ff}$. Such pairs are eliminated from the beginning by the tests for $\mu \odot \nu \neq \mathbf{ff}$.

We can obtain shortcuts for the derived operators "always" and "eventually".

Lemma 4.
$$\begin{aligned}
\textsc{lf}(\Diamond \psi) &= \textsc{lf}(\psi) \cup \{\langle\mathbf{tt}, \Diamond \psi\rangle\} \\
\textsc{lf}(\Box \psi) &= \{\langle\nu, \psi' \wedge \Box \psi\rangle \mid \langle\nu, \psi'\rangle \in \textsc{lf}(\psi)\}
\end{aligned}$$

Example 1. Consider the formula $\Box \Diamond p$.

$$\begin{aligned}
\textsc{lf}(\Diamond p) &= \textsc{lf}(p) \cup \{\langle\mathbf{tt}, \Diamond p\rangle\} \\
&= \{\langle p, \mathbf{tt}\rangle, \langle\mathbf{tt}, \Diamond p\rangle\} \\
\textsc{lf}(\Box \Diamond p) &= \{\langle\mu, \varphi' \wedge \Box \Diamond p\rangle \mid \langle\mu, \varphi'\rangle \in \textsc{lf}(\Diamond p)\} \\
&= \{\langle\mu, \varphi' \wedge \Box \Diamond p\rangle \mid \langle\mu, \varphi'\rangle \in \{\langle p, \mathbf{tt}\rangle, \langle\mathbf{tt}, \Diamond p\rangle\}\} \\
&= \{\langle p, \Box \Diamond p\rangle, \langle\mathbf{tt}, \Diamond p \wedge \Box \Diamond p\rangle\}
\end{aligned}$$

Definition 9 (Linear Forms). *A formula* $\varphi = \bigvee_{i\in I} b_i \wedge \bigcirc \varphi_i$ *is in* linear form *if each* b_i *is a conjunction of literals and each* φ_i *is a temporal formula.*

The formula associated to a set of linear factors is in linear form as given by the following mapping.

$$\Theta(\{\langle\mu_i, \varphi_i\rangle \mid i \in I\}) = \bigvee_{i\in I} (\Theta(\mu_i) \wedge \bigcirc \varphi_i)$$

Each formula can be represented in linear form by applying the transformation to linear factors. The expansion theorem states the correctness of this transformation.

Theorem 2 (Expansion). *For all* φ, $\Theta(\textsc{lf}(\varphi)) \Leftrightarrow \varphi$.

The partial derivative of a formula φ with respect to a symbol $x \in \Sigma$ is a set of formulae Ψ such that $x\sigma \models \varphi$ if and only if $\sigma \models \bigvee \Psi$. Partial derivatives only need to be defined for formal conjunctions of temporal formulae as we can apply the $\mathcal{T}$ operator first.

Definition 10 (Partial Derivatives). *The partial derivative of a formal conjunction of temporal formulae with respect to a symbol $x \in \Sigma$ is defined by*

$$\begin{aligned}
\partial_x(\varphi) &= \{\varphi' \mid \langle \mu, \varphi' \rangle \in \mathrm{LF}(\varphi), x \models \mu\} && \textit{if } \varphi \textit{ is a temporal formula} \\
\partial_x(\mathbf{tt}) &= \{\mathbf{tt}\} \\
\partial_x(\varphi \wedge \psi) &= \{\varphi' \wedge \psi' \mid \varphi' \in \partial_x(\varphi), \psi' \in \partial_x(\psi)\}.
\end{aligned}$$

Example 2. Continuing the example of $\Box \Diamond p$, we find for $x \in \Sigma$:

$$\begin{aligned}
\partial_x(\Box \Diamond p) &= \{\varphi' \mid \langle \mu, \varphi' \rangle \in \mathrm{LF}(\Box \Diamond p), x \models \mu\} \\
&= \{\varphi' \mid \langle \mu, \varphi' \rangle \in \{\langle p, \Box \Diamond p \rangle, \langle \mathbf{tt}, \Diamond p \wedge \Box \Diamond p \rangle\}, x \models \mu\} \\
&= \begin{cases} \{\Box \Diamond p, \Diamond p \wedge \Box \Diamond p\} & p \in \mathcal{I}(x) \\ \{\Diamond p \wedge \Box \Diamond p\} & p \notin \mathcal{I}(x) \end{cases}
\end{aligned}$$

As it is sufficient to define the derivative for temporal formulae, it only remains to explore the definition of $\partial_x(\Diamond p)$.

$$\begin{aligned}
\partial_x(\Diamond p) &= \{\varphi' \mid \langle \mu, \varphi' \rangle \in \mathrm{LF}(\Diamond p), x \models \mu\} \\
&= \{\varphi' \mid \langle \mu, \varphi' \rangle \in \{\langle p, \mathbf{tt} \rangle, \langle \mathbf{tt}, \Diamond p \rangle\}, x \models \mu\} \\
&= \begin{cases} \{\mathbf{tt}, \Diamond p\} & p \in \mathcal{I}(x) \\ \{\Diamond p\} & p \notin \mathcal{I}(x) \end{cases}
\end{aligned}$$

3.1 Properties of Partial Derivatives

A descendant of a formula is either the formula itself or an element of the partial derivative of a descendant by some symbol. As in the regular expression case, the set of descendants of a fixed LTL formula is finite. Our finiteness proof follows the method suggested by Broda and coworkers [3]. We look at the set of iterated partial derivatives of a formula φ, which turns out to be just the set of temporal subformulae of φ. This set is finite and closed under the partial derivative operation. Thus, finiteness follows.

Definition 11 (Iterated Partial Derivatives).

$$\begin{aligned}
\partial^+(\ell) &= \{\ell\} \\
\partial^+(\mathbf{tt}) &= \{\mathbf{tt}\} \\
\partial^+(\mathbf{ff}) &= \{\mathbf{ff}\} \\
\partial^+(\varphi \vee \psi) &= \partial^+(\varphi) \cup \partial^+(\psi) \\
\partial^+(\varphi \wedge \psi) &= \partial^+(\varphi) \cup \partial^+(\psi) \\
\partial^+(\bigcirc \varphi) &= \{\bigcirc \varphi\} \cup \partial^+(\varphi) \\
\partial^+(\Diamond \varphi) &= \{\Diamond \varphi\} \cup \partial^+(\varphi) \\
\partial^+(\Box \varphi) &= \{\Box \varphi\} \cup \partial^+(\varphi) \\
\partial^+(\varphi \mathbf{U} \psi) &= \{\varphi \mathbf{U} \psi\} \cup \partial^+(\psi) \cup \partial^+(\varphi) \\
\partial^+(\varphi \mathbf{R} \psi) &= \{\varphi \mathbf{R} \psi\} \cup \partial^+(\psi) \cup \partial^+(\varphi)
\end{aligned}$$

It is trivial to see that the set $\partial^+(\varphi)$ is finite because it is a subset of the set of subformulae of φ.

Lemma 5 (Finiteness). *For all φ, $\partial^+(\varphi)$ is finite.*

The iterated partial derivative only consider subformulae whereas the partial derivative elides disjunctions but returns a set of formal conjunctions. To connect both the following definition is required.

Definition 12 (Subsets of Formal Conjunctions). *For an ordered set $X = \{x_1, x_2, \dots\}$, we define the set of all formal conjunctions of X as follows.*

$$\mathcal{S}(X) = \{\mathbf{tt}\} \cup \{x_{i_1} \wedge \ldots \wedge x_{i_n} \mid n \geq 1, i_1 < i_2 < \cdots < i_n\}$$

We regard a subset of $\mathcal{S}(X)$ as a positive Boolean formula over X in conjunctive normal form.

Clearly, if a set of formulae Φ is finite, then so is $\mathcal{S}(\Phi)$, where we assume an arbitrary, but fixed total ordering on formulae.

The set of temporal subformulae of a given formula φ is also a formal conjunction of subformulae.

Lemma 6. *For all φ, $\mathcal{T}(\varphi) \subseteq \mathcal{S}(\partial^+(\varphi))$.*

Lemma 7 (Closedness under derivation).

1. *For all $x \in \Sigma$, $\partial_x(\varphi) \subseteq \mathcal{S}(\partial^+(\varphi))\}$.*
2. *For all $\varphi' \in \partial^+(\varphi)$ and $x \in \Sigma$, $\partial_x(\varphi') \subseteq \mathcal{S}(\partial^+(\varphi))$.*

From Lemmas 6 and 7 it follows that the set of descendants of a fixed LTL formula φ is finite. In fact, we can show that the cardinality of this set is exponential in the size of φ. We will state this result for a more "direct" definition of partial derivatives which does not require having to compute linear factors first.

Definition 13 (Direct Partial Derivatives). *Let $x \in \Sigma$. Then, $pd_x(\cdot)$ maps LTL formulae to sets of LTL formulae and is defined as follows.*

$$\begin{array}{ll}
pd_x(\mathbf{tt}) & = \{\mathbf{tt}\} \\
pd_x(\mathbf{ff}) & = \{\} \\
pd_x(\ell) & = \begin{cases} \{\mathbf{tt}\} & x \models \ell \\ \{\} & \mathit{otherwise} \end{cases} \\
pd_x(\varphi \vee \psi) & = pd_x(\varphi) \cup pd_x(\psi) \\
pd_x(\varphi \wedge \psi) & = \{\varphi' \wedge \psi' \mid \varphi' \in pd_x(\varphi), \psi' \in pd_x(\psi)\} \\
pd_x(\bigcirc \varphi) & = \mathcal{T}(\varphi) \\
pd_x(\varphi \,\mathbf{U}\, \psi) & = pd_x(\psi) \cup \{\varphi' \wedge \varphi \,\mathbf{U}\, \psi \mid \varphi' \in pd_x(\varphi)\} \\
pd_x(\varphi \,\mathbf{R}\, \psi) & = \{\varphi' \wedge \psi' \mid \varphi' \in pd_x(\varphi), \psi' \in pd_x(\psi)\} \cup \{\psi' \wedge \varphi \,\mathbf{R}\, \psi \mid \psi' \in pd_x(\psi)\} \\
pd_x(\Diamond \varphi) & = pd_x(\varphi) \cup \{\Diamond \varphi\} \\
pd_x(\Box \varphi) & = \{\varphi' \wedge \Box \varphi \mid \varphi' \in pd_x(\varphi)\}
\end{array}$$

where conjunctions of temporal formulae are normalized as usual.

For $w \in \Sigma^*$, we define $pd_\varepsilon(\varphi) = \{\varphi\}$ and $pd_{xw}(\varphi) = \bigcup_{\varphi' \in pd_x(\varphi)} pd_w(\varphi')$. For $L \subseteq \Sigma*$, we define $pd_L(\varphi) = \bigcup_{w \in L} pd_w(\varphi)$. We refer to the special case $pd_{\Sigma^*}(\varphi)$ as the set of partial derivative descendants *of φ.*

Example 3. Consider the formula $\Box \Diamond p$. We calculate

$$\begin{array}{ll}
pd_p(\Diamond p) & = \{\mathbf{tt}, \Diamond p\} \\
pd_p(\Box \Diamond p) & = \{\mathbf{tt} \wedge \Box \Diamond p, \Diamond p \wedge \Box \Diamond p\} \\
& \quad \text{(normalize)} \\
& = \{\Box \Diamond p, \Diamond p \wedge \Box \Diamond p\} \\
pd_p(\Diamond p \wedge \Box \Diamond p) & = \{\mathbf{tt} \wedge \mathbf{tt} \wedge \Box \Diamond p, \Diamond p \wedge \Box \Diamond p, \mathbf{tt} \wedge \Diamond p \wedge \Box \Diamond p, \Diamond p \wedge \Diamond p \wedge \Box \Diamond p\} \\
& \quad \text{(normalize)} \\
& = \{\Box \Diamond p, \Diamond p \wedge \Box \Diamond p\}
\end{array}$$

Lemma 8. *For all φ and $x \in \Sigma$, $\partial_x(\varphi) = pd_x(\varphi)$.*

The next result follows from Theorem 2 and Lemma 8.

Lemma 9. *For all φ, $\varphi \Leftrightarrow \bigvee_{x \in \Sigma, \varphi' \in pd_x(\varphi)} x \wedge \bigcirc \varphi'$.*

Definition 14. *The* size *of a temporal formula φ is the sum of the number of literals, temporal and Boolean operators in φ.*

If φ has size n, the number of subformulae in φ is bounded by $O(n)$.

Lemma 10. *For all φ, the cardinality of $pd_{\Sigma^*}(\varphi)$ is bounded by $O(2^n)$ where n is the size of φ.*

4 Alternating ω-Automata

We revisit the standard construction from LTL formula to alternating ω-automata as reported by Vardi [16]. We observe that the definition of the transition function for formulae in PNF corresponds to partial derivatives.

The transition function of an alternating automaton yields a set of sets of states, which we understand as a disjunction of conjunctions of states. The disjunction models the nondeterministic alternatives that the automaton can take in a step, whereas the conjunction models states that need to succeed together. Many presentations use positive Boolean formulae at this point, our presentation uses the set of minimal models of such formulae.

Definition 15. *A tuple $\mathcal{A} = (Q, \Sigma, \delta, \alpha_0, F)$ is an* alternating ω-automaton *(AA) [9] if Q is a finite set of states, Σ an alphabet, $\alpha_0 \subseteq \mathbb{P}(Q)$ a set of sets of states, $\delta : Q \times \Sigma \to \mathbb{P}(Q)$ a transition function, and $F \subseteq Q$ a set of accepting states.*

A run of $\mathcal{A}$ on a word σ *is a digraph $G = (V, E)$ with nodes $V \subseteq Q \times \omega$ and edges $E \subseteq \bigcup_{i \in \omega} V_i \times V_{i+1}$ where $V_i = Q \times \{i\}$, for all i.*

- $\{q \in Q \mid (q, 0) \in V\} \in \alpha_0$.
- *For all $i \in \omega$:*
  - *If $(q', i+1) \in V_{i+1}$, then $((q, i), (q', i+1)) \in E$, for some $q \in Q$.*
  - *If $(q, i) \in V_i$, then $\{q' \in Q \mid ((q, i), (q', i+1)) \in E\} \in \delta(q, \sigma_i)$.*

A run G on σ is accepting *if every infinite path in G visits a state in F infinitely often (Büchi acceptance). Define the language of $\mathcal{A}$ as*

$$\mathcal{L}(\mathcal{A}) = \{\sigma \mid \text{ there exists an accepting run of } \mathcal{A} \text{ on } \sigma\}.$$

Definition 16 ([16,10]). *The alternating ω-automaton $\mathcal{A}(\varphi) = (Q, \Sigma, \delta, \alpha_0, F)$ resulting from φ is defined as follows. The set of states is $Q = \partial^+(\varphi)$, the set of initial states $\alpha_0 = \mathcal{T}(\varphi)$, the set of accepting states $F = \{\mathbf{tt}\} \cup \{\varphi \mathbf{R} \psi \mid \varphi \mathbf{R} \psi \in Q\}$, and the transition function δ is defined by induction on the formula argument:*

- $\delta(\mathbf{tt}, x) = \{\mathbf{tt}\}$
- $\delta(\mathbf{ff}, x) = \{\}$
- $\delta(\ell, x) = \{\mathbf{tt}\}$, *if* $x \models \ell$
- $\delta(\ell, x) = \{\}$, *otherwise*
- $\delta(\varphi \vee \psi, x) = \delta(\varphi, x) \cup \delta(\psi, x)\}$
- $\delta(\varphi \wedge \psi, x) = \{q_1 \wedge q_2 \mid q_1 \in \delta(\varphi, x), q_2 \in \delta(\psi, x)\}$
- $\delta(\bigcirc \varphi, x) = \mathcal{T}(\varphi)$
- $\delta(\varphi \mathbf{U} \psi, x) = \delta(\psi, x) \cup \{q \wedge \varphi \mathbf{U} \psi \mid q \in \delta(\varphi, x)\}$
- $\delta(\varphi \mathbf{R} \psi, x) = \{q_1 \wedge q_2 \mid q_1 \in \delta(\varphi, x), q_2 \in \delta(\psi, x)\} \cup \{q \wedge \varphi \mathbf{R} \psi \mid q \in \delta(\psi, x)\}$

We deviate slightly from Vardi's original definition by representing disjunction as a set of states. For example, in his definition $\delta(\mathbf{ff}, x) = \mathbf{ff}$, which is equivalent to the empty disjunction. Another difference is that we only consider formulae in PNF whereas Vardi covers LTL in general. Hence, Vardi's formulation treats negation by extending the set of states with negated subformulae. For example, we find $\delta(\neg\varphi, x) = \overline{\delta(\varphi, x)}$ where $\overline{\Phi}$ calculates the dual of a set Φ of formulae obtained by application of the de Morgan laws. The case for negation can be dropped because we assume that formulae are in PNF. In exchange, we need to state the cases for $\varphi \vee \psi$ and for $\varphi \mathbf{R} \psi$ which can be derived easily from Vardi's formulation by exploiting standard LTL equivalences.

The accepting states in Vardi's construction are all subformulae of the form $\neg(\varphi \mathbf{U} \psi)$, but $\neg(\varphi \mathbf{U} \psi) = (\neg\varphi) \mathbf{R} (\neg\psi)$, which matches our definition and others in the literature [5].

Furthermore, our construction adds $\mathbf{tt}$ to the set of accepting states, which is not present in Vardi's paper. It turns out that $\mathbf{tt}$ can be eliminated from the accepting states if we set $\delta(\mathbf{tt}, x) = \{\}$. This change transforms an infinite path with infinitely many $\mathbf{tt}$ states into a finite path that terminates when truth is established. Thus, it does not affect acceptance of the AA.

The same definition is given by Pelánek and Strejček [11] who note that the resulting automaton is restricted to be a 1-weak alternating automaton. For this class of automata there is a translation back to LTL.

We observe that the definition of the transition function in Definition 16 corresponds to the direct definition of partial derivatives in Definition 13.

Lemma 11. *Let $\mathcal{A}(\varphi)$ be the alternating ω-automaton for a formula φ according to Definition 16. For each $\psi \in Q$ and $x \in \Sigma$, we have that $\delta(\psi, x) = pd_x(\psi)$.*

Finally we provide an independent correctness result for the translation from LTL to AA that relies on the correctness of our construction of linear factors.

Theorem 3. *Let φ be an LTL formula. Consider the alternating automaton $\mathcal{A}(\varphi)$ given by*

- $Q = \partial^+(\varphi)$,
- $\delta(\psi, x) = \partial_x(\psi)$, *for all* $\psi \in Q$ *and* $x \in \Sigma$,
- $\alpha_0 = \mathcal{T}(\varphi)$,
- $F = \{\mathbf{tt}\} \cup \{\varphi \mathbf{R} \psi \mid \varphi \mathbf{R} \psi \in Q\}$.

Then, $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{A}(\varphi))$ *using the Büchi acceptance condition.*

5 Semantic Tableau

We revisit Wolper's [21] semantic tableau method for LTL to check satisfiability of a formula φ. A tableau is represented as a directed graph built from nodes where nodes denote sets of formulae. A tableau starts with the initial node $\{\varphi\}$ and new nodes are generated by decomposition of existing nodes, i.e. formulae. Wolper's method requires a post-processing phase where unsatisfiable nodes are eliminated. The formula φ is satisfiable if there is a satisfiable path in the tableau. We observe that decomposition can be explained in terms of linear factors and some of the elimination (post-processing) steps can be obtained for free.

We largely follow Wolper's notation but start from formulae in PNF. In the construction of a tableau, a formula φ may be *marked*, written as $\varphi*$. A formula is *elementary* if it is a literal or its outermost connective is $\bigcirc$. We write S to denote a set of formulae. Hence, each node is represented by some S. A node is called a *state* if the node consists solely of elementary or marked formulae. A node is called a *pre-state* if it is the initial node or the immediate child of a state.

Definition 17 (Wolper's Tableau Decision Method [21]). *Tableau construction for φ starts with node $S = \{\varphi\}$. New nodes are created as follows.*

- *Decomposition rules: For each non-elementary unmarked $\varphi \in S$ with decomposition rule $\varphi \to \{S_1, \ldots, S_k\}$ as defined below, create k child nodes where the ith child is of the form $(S - \{\varphi\}) \cup S_i \cup \{\varphi*\}$.*

$$\begin{array}{ll}
\text{(D1)} & \varphi \vee \psi \to \{\{\varphi\}, \{\psi\}\} \\
\text{(D2)} & \varphi \wedge \psi \to \{\{\varphi, \psi\}\} \\
\text{(D3)} & \Diamond \varphi \to \{\{\varphi\}, \{\bigcirc \Diamond \varphi\}\} \\
\text{(D4)} & \Box \varphi \to \{\{\varphi, \bigcirc \Box \varphi\}\} \\
\text{(D5)} & \varphi \mathbf{U} \psi \to \{\{\psi\}, \{\varphi, \bigcirc (\varphi \mathbf{U} \psi)\}\} \\
\text{(D6)} & \varphi \mathbf{R} \psi \to \{\{\psi, \varphi \vee \bigcirc (\varphi \mathbf{R} \psi)\}\}
\end{array}$$

- *Step rule: For each node S consisting of only elementary or marked formulae, create a child node $\{\varphi \mid \bigcirc \varphi \in S\}$. Just create an edge if the node already exists.*

$$\begin{tikzcd}
 & S_0 = \{\Box\, p \wedge \Diamond\, \neg p\} \arrow[d, "2"] \\
 & S_1 = \{\Box\, p \wedge \Diamond\, \neg p*, \Box\, p, \Diamond\, \neg p\} \arrow[d, "4"] \\
 & S_2 = \{\Box\, p \wedge \Diamond\, \neg p*, \Box\, p*, \Diamond\, \neg p, p, \bigcirc\, \Box\, p\} \arrow[dl, "3"'] \arrow[d, "3"] \\
S_3 = \{\Box\, p \wedge \Diamond\, \neg p*, \Box\, p*, \Diamond\, \neg p*, p, \bigcirc\, \Box\, p, \neg p\} & S_4 = \{\Box\, p \wedge \Diamond\, \neg p*, \Box\, p*, \Diamond\, \neg p*, p, \bigcirc\, \Box\, p, \bigcirc\, \Diamond\, \neg p\} \arrow[d] \\
 & S_5 = \{\Box\, p, \Diamond\, \neg p\} \arrow[d, "4"] \\
 & S_6 = \{\Box\, p*, \Diamond\, \neg p, p, \bigcirc\, \Box\, p\} \arrow[dl, "3"'] \arrow[d, "3"] \\
S_7 = \{\Box\, p*, \Diamond\, \neg p*, p, \bigcirc\, \Box\, p, \neg p\} & S_8 = \{\Box\, p*, \Diamond\, \neg p*, p, \bigcirc\, \Box\, p, \bigcirc\, \Diamond\, \neg p\} \arrow[uu, bend right=60]
\end{tikzcd}$$

Fig. 1. Tableau before elimination: $\Box\, p \wedge \Diamond\, \neg p$

Elimination of (unsatisfiable) nodes proceeds as follows. A node is eliminated if one of the conditions E1-3 applies.

- *E1: The node contains p and its negation.*
- *E2: All successors have been eliminated.*
- *E3: The node is a pre-state and contains a formula of the form $\Diamond\, \psi$ or $\varphi\, \mathbf{U}\, \psi$ that is not satisfiable.*

A formula $\Diamond\, \psi$, $\varphi\, \mathbf{U}\, \psi$ is satisfiable in a pre-state, if there is a path in the tableau leading from that pre-state to a node containing the formula ψ.

Theorem 4 (Wolper [21]). *An LTL formula φ is satisfiable iff the initial node generated by the tableau decision procedure is not eliminated.*

Example 4. Consider $\Box\, p \wedge \Diamond\, \neg p$. Figure 1 shows the tableau generated before elimination. In case of decomposition, edges are annotated with the number of the respective decomposition rule. For example, from the initial node S_0 we reach node S_1 by decomposition via (D2). Node S_4 consists of only elementary and marked nodes and therefore we apply the step rule to reach node S_5. The same applies to node S_3. For brevity, we ignore its child node because this node is obviously unsatisfiable (E1). The same applies to node S_7.

We consider elimination of nodes. Nodes S_3, S_4, S_7 and S_8 are states. Therefore, S_0 and S_5 are pre-states. Nodes S_3 and S_7 can be immediately eliminated due to E1. Node S_5 contains $\Diamond \neg p$. This formula is not satisfiable because there is not path from S_5 along which we reach a node which contains $\neg p$. Hence, we eliminate S_5 due to E3. All other nodes are eliminated due to E3. Hence, we conclude that the formula $\Box p \wedge \Diamond \neg p$ is unsatisfiable.

We argue that marked formulae and intermediate nodes are not essential in Wolper's tableau construction. Marked formulae can simply be dropped and intermediate nodes can be removed by exhaustive application of decomposition. This optimization reduces the size of the tableau and allows us to establish a direct connection between states/pre-states and linear factors/partial derivatives.

Definition 18 (Decomposition and Elimination via Rewriting). *We define a rewrite relation among sets of sets of nodes ranged over by N.*

$$\frac{\text{“}\varphi \to \{S_1, \ldots, S_n\}\text{”} \in \{D1, \ldots, D6\}}{\{S \cup \{\varphi\}\} \cup N \rightarrowtail \{S \cup S_1\} \cup \cdots \cup \{S \cup S_n\} \cup N}$$

$$\frac{N' = \{S \mid S \in N \wedge (\forall \ell \in S)\ \neg\ell \notin S\}}{N \rightarrowtail N'}$$

where the premise of the first rule corresponds to one of the decomposition rules D1-D6. The second rule corresponds to the elimination rule E1 applied globally. We write $N_1 \rightarrowtail^* N_k$ for $N_1 \rightarrowtail \cdots \rightarrowtail N_k$ where no further rewritings are possible on N_k. We write $\varphi \rightarrowtail^* N$ as a shorthand for $\{\{\varphi\}\} \rightarrowtail^* N$.

As the construction does not mark formulae, we call S a state node if S only consists of elementary formulae. By construction, for any set of formulae S we find that $\{S\} \rightarrowtail^* N$ for some N which only consists of state nodes. In our optimized Wolper-style tableau construction, each $S' \in N$ is a ‘direct’ child of S where intermediate nodes are skipped. We also integrate the elimination condition E1 into the construction of new nodes.

The step rule is pretty much the same as in Wolper's formulation. The (mostly notational) difference is that we represent a pre-state node with a single formula. That is, from state node S we generate the child pre-state node $\{\bigwedge_{\bigcirc \psi \in S} \psi\}$ whereas Wolper generates $\{\psi \mid \bigcirc \psi \in S\}$.

Definition 19 (Optimized Tableau Construction Method). *We consider tableau construction for φ. We assume that Q denotes the set of pre-state formulae generated so far and Q_j denotes the set of active pre-state nodes in the j-th construction step. We start with $Q = Q_0 = \{\varphi\}$. We consider the j-th construction step.*

Decomposition: *For each node $\psi \in Q_j$ we build $\{\{\psi\}\} \rightarrowtail^* \{S_1, \ldots, S_n\}$ where each state node S_i is a child of pre-state node $\{\psi\}$.*

Step: *For each state node S_i, we build $\varphi_i = \bigwedge_{\bigcirc \varphi \in S_i} \psi$ where pre-state node $\{\varphi_i\}$ is a child of S_i. We set $Q_{j+1} = \{\varphi_1, \ldots, \varphi_n\} - Q$ and then update the set of pre-state formulae generated so far by setting $Q = Q \cup \{\varphi_1, \ldots, \varphi_n\}$.*

Construction continues until no new children are created.

Theorem 5 (Correctness Optimized Tableau). *For all φ, φ is satisfiable iff the initial node generated by the optimized Wolper-style tableau decision procedure is not eliminated by conditions E2 and E3.*

Example 5. Consider $\Box\, p \wedge \Diamond\, \neg p$. Our variant of Wolper's tableau construction method yields the following.

$$\begin{tikzcd}
S_0 = \{\Box\, p \wedge \Diamond\, \neg p\} \arrow[d, "decomp"] \\
S'_4 = \{p, \bigcirc \Box\, p, \bigcirc \Diamond\, \neg p\} \arrow[u, bend right=60, swap, "step"]
\end{tikzcd}$$

Node S'_4 corresponds to node S_4 in Figure 1. Nodes S_1, S_2, and S_3 from the original construction do not arise in our variant because we skip intermediate nodes and eliminate aggressively during construction whereas Wolper's construction method gives rise S_5. We avoid such intermediate nodes and immediately link S'_4 to the initial node S_0.

Next, we show that states in the optimized representation of Wolper's tableau method correspond to linear factors and pre-states correspond to partial derivatives.

Let $S = \{\ell_1, \ldots, \ell_n, \bigcirc \varphi_1, \ldots, \bigcirc \varphi_m\}$ be a (state) node. We define $[\![S]\!] = \langle \ell_1 \odot \ldots \odot \ell_n, \varphi_1 \wedge \ldots \wedge \varphi_m\rangle$ where for cases $n = 0$ and $m = 0$ we assume $\mathbf{tt}$. Let $N = \{S_1, \ldots, S_n\}$ where each S_i is a state. We define $[\![N]\!] = \{[\![S_1]\!], \ldots, [\![S_n]\!]\}$.

Lemma 12. *Let $\varphi \neq \mathbf{ff}$, N such that $\varphi \rightarrowtail^* N$. Then, we find that* $\textsc{lf}(\varphi) = [\![N]\!]$.

Case $\mathbf{ff}$ is excluded due to $LF(\mathbf{ff}) = \{\}$.

So, any state node generated during the optimized Wolper tableau construction corresponds to an element of a linear factor. An immediate consequence is that each pre-state corresponds to a partial derivative. Hence, we can reformulate the optimized Wolper tableau construction as follows.

Theorem 6 (Tableau Construction via Linear Factors). *The optimized variant of Wolper's tableau construction for φ can be obtained as follows.*

1. *Each formula $\psi \neq \mathbf{tt}$ in the set of all partial derivative descendants $pd_{\Sigma^*}(\varphi)$ corresponds to a pre-state.*
2. *For each $\psi \in pd_{\Sigma^*}(\varphi)$ where $\psi \neq \mathbf{tt}$, each $\langle \nu, \psi' \rangle \in \textsc{lf}(\psi)$ is state where $\langle \nu, \psi' \rangle$ is a child of ψ, and if $\psi' \neq \mathbf{tt}$, ψ' is a child of $\langle \nu, \psi' \rangle$.*

We exclude $\mathbf{tt}$ because Wolper's tableau construction stops once we reach $\mathbf{tt}$.

Example 6. Consider $\neg p \wedge \bigcirc \neg p \wedge q \,\mathbf{U}\, p$ where

$$\begin{aligned}
\textsc{lf}(\neg p) &= \{\langle \neg p, \mathbf{tt}\rangle\} \\
\textsc{lf}(\mathbf{tt}) &= \{\langle \mathbf{tt}, \mathbf{tt}\rangle\} \\
\textsc{lf}(\bigcirc \neg p) &= \{\langle \mathbf{tt}, \neg p\rangle\} \\
\textsc{lf}(q \,\mathbf{U}\, p) &= \{\langle p, \mathbf{tt}\rangle, \langle q, q \,\mathbf{U}\, p\rangle\} \\
\textsc{lf}(\neg p \wedge q \,\mathbf{U}\, p) &= \{\langle \neg p \wedge q, q \,\mathbf{U}\, p\rangle\} \\
\textsc{lf}(\neg p \wedge \bigcirc \neg p \wedge q \,\mathbf{U}\, p) &= \{\langle \neg p \wedge q, \neg p \wedge q \,\mathbf{U}\, p\rangle\}
\end{aligned}$$

We carry out the tableau construction using linear factors notation where we use LF to label pre-state (derivatives) to state (linear factor) relations and PD to label state to pre-state relations.

$$\begin{tikzcd}
& \neg p \land \bigcirc \neg p \land q \,\mathbf{U}\, p \arrow[d, "LF"] \\
& \langle \neg p \land q, \neg p \land q \,\mathbf{U}\, p \rangle \arrow[d, "PD"] \\
& \neg p \land q \,\mathbf{U}\, p \arrow[d, "LF"] \\
& \langle \neg p \land q, q \,\mathbf{U}\, p \rangle \arrow[d, "PD"] \\
& q \,\mathbf{U}\, p \arrow[dl, "LF"'] \arrow[d, "LF"] \\
\langle p, \mathbf{tt} \rangle & \langle q, q \,\mathbf{U}\, p \rangle
\end{tikzcd}$$

The reformulation of Wolper's tableau construction in terms of linear factors and partial derivatives allows us to establish a close connection to the construction of Vardi's alternating ω-automaton. Each path in the tableau labeled by LF and PD corresponds to a transition step in the automaton. The same applies to transitions with one exception. In Wolper's tableau, the state $\langle \ell, \mathbf{tt} \rangle$ is considered final whereas in Vardi's automaton we find transitions $\delta(\ell, \mathbf{tt}) = \{\mathbf{tt}\}$. So, from Theorem 3 and Theorem 6 we can derive the following result.

Corollary 1. *Vardi's alternating ω-automaton derived from an LTL formula is isomorphic to Wolper's optimized LTL tableau construction assuming we ignore transitions $\delta(\ell, \mathbf{tt}) = \{\mathbf{tt}\}$.*

6 Related Work and Conclusion

There have been numerous works that study the translation of LTL to ω-automata [10,17,16,8,4] and semantic tableaux [21,14,13]. The fact that there is a deep connection between both constructions appears to be folklore knowledge. For example, see [8]: "The central part of the automaton construction algorithm is a tableau-like procedure." Couvreur [4] mentions explicitly that "[his automaton construction] is also based on tableau procedures [20,21]." To the best of our knowledge, we are the first to establish a concise connection between the two constructions by means of linear factors and partial derivatives, as shown in Corollary 1. Both concepts have been studied previously in the standard regular expression setting [1] and also in the context of ω-regular languages [15]. We show that both concepts are applicable in the LTL setting and establish their essential properties.

Like some earlier works [8,4], our algorithm can operate on the fly and thus avoid the construction of the full automaton if the algorithm can provide an answer earlier. Further efficiency gains (in checking satisfiability) can be achieved by using Tarjan's algorithm [7] and this improvement is also compatible with our algorithm.

References

1. Valentin M. Antimirov. Partial derivatives of regular expressions and finite automaton constructions. *Theoretical Computer Science*, 155(2):291–319, 1996.
2. Tomás Babiak, Mojmír Kretínský, Vojtech Rehák, and Jan Strejček. LTL to büchi automata translation: Fast and more deterministic. In *TACAS*, volume 7214 of *LNCS*, pages 95–109. Springer, 2012.
3. Sabine Broda, António Machiavelo, Nelma Moreira, and Rogério Reis. Partial derivative automaton for regular expressions with shuffle. In *Proc. of DCFS*, volume 9118 of *LNCS*, pages 21–32. Springer, 2015.
4. Jean-Michel Couvreur. On-the-fly verification of linear temporal logic. In *World Congress on Formal Methods*, volume 1708 of *LNCS*, pages 253–271. Springer, 1999.
5. Bernd Finkbeiner and Henny Sipma. Checking finite traces using alternating automata. *Formal Methods in System Design*, 24(2):101–127, 2004.
6. Paul Gastin and Denis Oddoux. Fast LTL to büchi automata translation. In *CAV*, volume 2102 of *LNCS*, pages 53–65. Springer, 2001.
7. Jaco Geldenhuys and Antti Valmari. More efficient on-the-fly LTL verification with Tarjan's algorithm. *Theor. Comput. Sci.*, 345(1):60–82, 2005.
8. Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proceedings of the Fifteenth IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification XV*, pages 3–18, London, UK, UK, 1996. Chapman & Hall, Ltd.
9. Christof Löding and Wolfgang Thomas. Alternating automata and logics over infinite words. In *IFIP TCS*, volume 1872 of *Lecture Notes in Computer Science*, pages 521–535. Springer, 2000.
10. David E. Muller, Ahmed Saoudi, and Paul E. Schupp. Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In *LICS*, pages 422–427. IEEE Computer Society, 1988.
11. Radek Pelánek and Jan Strejček. Deeper connections between LTL and alternating automata. In *CIAA*, volume 3845 of *Lecture Notes in Computer Science*, pages 238–249. Springer, 2005.
12. Amir Pnueli. The temporal logic of programs. In *Proc. of SFCS '77*, pages 46–57. IEEE Computer Society, 1977.
13. Mark Reynolds. A new rule for LTL tableaux. In *Proc. of GandALF'16*, volume 226 of *EPTCS*, pages 287–301, 2016.
14. Stefan Schwendimann. A new one-pass tableau calculus for PLTL. In *Proc. of TABLEAUX '98*, volume 1397 of *LNCS*, pages 277–292. Springer, 1998.

15. Peter Thiemann and Martin Sulzmann. From ω -regular expressions to büchi automata via partial derivatives. In *Proc. of LATA'15*, volume 8977 of *LNCS*, pages 287–298. Springer, 2015.
16. Moshe Y. Vardi. Nontraditional applications of automata theory. In *Proceedings of the International Conference on Theoretical Aspects of Computer Software*, TACS '94, pages 575–597, London, UK, UK, 1994. Springer-Verlag.
17. Moshe Y. Vardi. Alternating automata: Unifying truth and validity checking for temporal logics. In *Proc. of CADE-14*, pages 191–206. Springer, 1997.
18. Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *LICS*, pages 332–344. IEEE Computer Society, 1986.
19. Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Inf. Comput.*, 115(1):1–37, 1994.
20. Pierre Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1/2):72–99, 1983.
21. Pierre Wolper. The tableau method for temporal logic: an overview. *Logique et Analyse*, 28(110-111), 1985.
22. Pierre Wolper, Moshe Y. Vardi, and A. Prasad Sistla. Reasoning about infinite computation paths (extended abstract). In *FOCS*, pages 185–194. IEEE Computer Society, 1983.