RWTH Aachen

Department of Computer Science
Aachener Informatik Berichte

Deciding LTL over Mazurkiewicz Traces

Benedikt Bollig and Martin Leucker

ISSN 0935–3232 · Technical Report Series · AIB-2001-02

RWTH Aachen · Department of Computer Science · February 2001

The publications of the Department of Computer Science of RWTH Aachen (*Aachen University of Technology*) are in general accessible through the World Wide Web and anonymous FTP:

ftp://ftp.informatik.rwth-aachen.de/pub/reports/index.html

Deciding LTL over Mazurkiewicz Traces

Benedikt Bollig and Martin Leucker*

Lehrstuhl für Informatik II
RWTH Aachen, Germany
Email: {bollig, leucker}@informatik.rwth-aachen.de

Abstract. Linear time temporal logic (LTL) has become a well established tool for specifying the dynamic behaviour of reactive systems with an interleaving semantics, and the automata-theoretic approach has proven to be a very useful mechanism for performing automatic verification in this setting. Especially alternating automata turned out to be a powerful tool in constructing efficient yet simple to understand decision procedures and directly yield further *on–the–fly* model checking procedures. In this paper we exhibit a decision procedure for LTL over Mazurkiewicz traces which generalises the classical automata-theoretic approach to a linear time temporal logic interpreted no longer over sequences but certain partial orders. Specifically, we construct a (linear) alternating Büchi automaton accepting the set of linearisations of traces satisfying the formula at hand. The salient point of our technique is to apply a notion of independence-rewriting to formulas of the logic. Furthermore, we show that the class of *linear* and *trace-consistent* alternating Büchi automata corresponds exactly to LTL formulas over Mazurkiewicz traces, lifting a similar result from Löding and Thomas formulated in the framework of LTL over words.

1 Introduction

Linear time Temporal Logic (LTL) as proposed by Pnueli [Pnu77] has become a well established tool for specifying the dynamic behaviour of distributed systems. The traditional approach towards automatic program verification is model checking specifications in LTL. A basic feature of LTL has been that its formulas are interpreted over sequences. Typically, such a sequence will model a computation of a system; a sequence of states visited by the system or a sequence of actions executed by the system during the course of the computation.

The automata-theoretic approach of Vardi and Wolper [VW86] for satisfiability checking has proven to be very useful and efficient for performing the automatic program verification. In its purest form, this amounts to the construction of a Büchi automaton accepting precisely the set of sequences which satisfies the specification expressed as an assertion of LTL. In the last years, a shift towards the employment of alternating automata for defining decision procedures took place. Alternating automata provide simple, efficient, and easy to understand decision procedures. They have proven to be useful for defining satisfiability algorithms for LTL over words [Var96], branching time logics [BVW94,KVW00] over

* Part of this work was done during the second author's stay at BRICS. He is grateful for the hospitality and the overall support.

finite transition systems, and the μ-calculus over (infinite) prefix-recognisable graphs [KV00]. The idea is that the states of the automaton are constructed essentially from the subformula closure of the specification formula, and the automaton operates in a tableau-like fashion. The satisfiability problem is then solved by checking whether the constructed automaton accepts any strings.

This approach forms the conceptual basis of many verification algorithms. Several tools (e.g. SPIN [GHP97]) being employed in industry are built upon this translation from formulas to automata. To improve performance, however, a number of substantial optimisations must be incorporated. One observation is that the state space of the product automaton needs seldomly to be fully constructed. Often the answer to the verification problem can be established by investigating only a subset of states, and this subset might be considerably smaller than the entire state space. This is the main idea underlying the so-called *on-the-fly* verification techniques. To support on-the-fly checking, an automaton corresponding to a formula should be defined in a *top-down* manner. This means that, given a formula φ and one of its subformulas ψ, a part of the automaton $\mathcal{A}_\varphi$ should be constructable without constructing $\mathcal{A}_\psi$, where $\mathcal{A}_\eta$ denotes the automaton accepting the models of η. In this way, the automaton for a given formula and an underlying transition system may only be constructed partly viz if the model checking or satisfiability question can already be answered by considering this part.

In many applications the computations of a distributed system will constitute interleavings of the occurrences of causally independent actions. Consequently, the computations can be naturally grouped together into equivalence classes where two computations are equated in case they are two different interleavings of the same partially ordered stretch of behaviour. It turns out that many of the properties expressed as LTL-formulas happen to have the so called "all-or-none" property. Either all members of an equivalence class of computations will have the desired property or none will do ("leads to deadlock" is one such property). For verifying such properties one has to check them for just one member of each equivalence class. This is the insight underlying many of the partial-order based verification methods (e.g. [Pel98,Val91]). As may be guessed, the importance of these methods lies in the fact that via these methods the computational resources required for the verification task can often be dramatically reduced.

It is often the case that the equivalence classes of computations generated by a distributed system constitute objects called Mazurkiewicz traces [DR95,Maz77]. They can be canonically represented as restricted labelled partial orders. This opens up an alternative way of exploiting the non-sequential nature of the computations of a distributed system and the attendant partial-order based methods. It consists of developing linear time temporal logics that can be interpreted directly over Mazurkiewicz traces. In these logics, every specification is guaranteed to have the "all-or-none" property and hence can be subjected to the partial-order based reduction methods during the verification process.

A number of linear time temporal logics to be interpreted directly over Mazurkiewicz traces (e.g. [APP95,Thi94,TW97]) has been proposed starting with TrPTL [Thi94]. There are several possible routes towards extending linear time temporal logics to traces. TrPTL is based on *locations*, where one reasons explicitly about a distribution of computing agents cooperating through some communication structure given as an alphabet distribution. Another option [APP95] is to view *events* as the partial order computation points in time, and base the specifications upon the relationship between individual events. Together these paradigms constitute the *local* trace logics. In contrast, in the *global* view of computations *configurations* are seen as instantaneous snapshots of the system at hand. In this sense, a configuration is a global view capturing a collection of simultaneous local views.

The "right" temporal logic for traces should be equal in expressive power to first-order logic for traces (FO). It follows from [EM96] that such a logic would capture exactly those properties of LTL which have the "all-or-none" property and hence are amenable to partial-order verification. However, none of the local logics are known to be expressively equivalent to FO. This lead Thiagarajan and Walukiewicz to define the configuration based LTrL [TW97], which they indeed prove equivalent to FO. LTrL was later refined [DG00] to a straightforward formulation of LTL for traces essentially extending Kamp's Theorem [Kam68] to the setting of traces.

While both the event based and location based logics all have elegant (exponential-time) decision procedures smoothly extending the classical automata-theoretic approach to the setting of traces, no such smooth extension exists for global logics such as LTL. The essence of this anomaly is the complications which arise as a consequence of the fact that the satisfiability problem for LTL has a nonelementary lower bound [Wal98]. However, experience [HJJK95] has shown that decision procedures can still be useful in practice despite discouraging lower bounds.

Gastin, Meyer, and Petit [GMP98a,GMP98b] do give a direct decision procedure for LTL based on automata. However, the construction of the automaton corresponding to a given LTL-specification φ proceeds by induction on φ, thus in a *bottom-up* manner. Hence it is not an extension of the classical automata-theoretic approach, and more important, it requires the construction of the full automaton, so optimisations such as on-the-fly checking cannot be applied. A further drawback is its high complexity. While an exponential blow-up is unavoidable for nested *until-formulas*, the procedure has also an exponential blow-up for every negation. Since nested *until*-formulas are rare in specifications but negations are typical for specifying unwanted behaviour, this limits the practical applicability of this procedure.

In this paper we propose a decision procedure for LTL for traces directly extending the classical approach [Var96]. Our procedure is based upon an extended subformula closure and independence rewriting of formulas of LTL. We employ this to construct a tableau-style alternating Büchi automaton accepting the set of linearisations of traces satisfying the specification at hand. In this sense, our procedure fills the missing gap for global trace logics by extending the classical approach to this last remaining case. Our procedure corresponds exactly to the version given in [Var96] when restricted to an empty independence relation. Furthermore, our automata can be constructed on-the-fly, which is crucial. Last but not least, for the fragment of LTL without *until*-formulas, our procedure is exponential.

In [LT00], it was shown that word languages definable by LTL-formulas over words correspond to the languages of *linear* alternating Büchi automata. We prove that our construction yields a linear Büchi automaton as well. Furthermore, we show that our linear Büchi automata accept *trace-consistent* languages. Conversely, we show that the class of trace-consistent languages definable by linear alternating Büchi automata coincides with the class of languages which are definable by LTL-formulas over Mazurkiewicz traces for a given dependency relation. This shows that linear alternating automata are a natural tool for linear time temporal logic, over words as well as over Mazurkiewicz traces.

In the next section we recall Mazurkiewicz traces and some related notions, which will play a crucial role for our present purposes. In Section 3 we introduce the basic object of our study, LTL, and interpret it directly over the domain of Mazurkiewicz traces. Following this, we give in Section 4 a brief account of alternating Büchi automata underlying our decision procedure to be presented in Section 5. There we supply a proof of correctness of our construction before giving a few concluding remarks in Section 6.

Acknowledgement: We would like to thank Jesper G. Henriksen for fruitful discussions and valuable comments on this work.

2 Mazurkiewicz Traces

A *(Mazurkiewicz) trace alphabet* is a pair (Σ, I), where Σ, the alphabet, is a finite set and $I \subseteq \Sigma \times \Sigma$ is an irreflexive and symmetric *independence relation*. Usually, Σ consists of the *actions* performed by a distributed system while I captures a static notion of causal independence between actions. We define $D = (\Sigma \times \Sigma) - I$ to be the *dependency relation* which is then reflexive and symmetric.

For the rest of the section we fix a trace alphabet (Σ, I). We will use aIb to denote that the actions a and b are independent, i.e., that $(a, b) \in I$, and use similar notation for $(a, b) \in D$. We extend the notion to sets of actions $X, Y \subseteq \Sigma$, and let XIY denote the fact that each pair of actions $a \in X$ and $b \in Y$ is pairwise independent. Moreover, XDY will denote that X is dependent upon Y, i.e., that there exists a pair of actions $a \in X$ and $b \in Y$ with a and b dependent. For convenience, we will write $\{a\}IY$ as aIY etc.

For the purpose of interpreting LTL over traces we will adopt the viewpoint that traces are restricted labelled partial orders of events and hence have an explicit representation of causality and concurrency.

Let $T = (E, \leq, \lambda)$ be a Σ-labelled poset. In other words, $(E, \leq)$ is a poset and $\lambda : E \to \Sigma$ is a labelling function. λ can be extended to subsets of E in the straightforward manner. For $e \in E$ we define ${\downarrow}e = \{x \in E \mid x \leq e\}$ and ${\uparrow}e = \{x \in E \mid e \leq x\}$. We let $\lessdot$ be the *covering relation* given by $x \lessdot y$ iff $x < y$ and for all $z \in E$, $x \leq z \leq y$ implies $x = z$ or $z = y$.

A *(Mazurkiewicz) trace* (over (Σ, I)) is a Σ-labelled poset $T = (E, \leq, \lambda)$ satisfying:

- ${\downarrow}e$ is a finite set for each $e \in E$.
- For every $e, e' \in E$, $e \lessdot e'$ implies $\lambda(e)\ D\ \lambda(e')$.
- For every $e, e' \in E$, $\lambda(e)\ D\ \lambda(e')$ implies $e \leq e'$ or $e' \leq e$.

We shall let $\mathbb{TR}(\Sigma, I)$ denote the class of traces over (Σ, I). As usual, a trace language L is a subset of traces, i.e., $L \subseteq \mathbb{TR}(\Sigma, I)$. Throughout the paper we will not distinguish between isomorphic elements in $\mathbb{TR}(\Sigma, I)$. We will refer to members of E as *events*.

Let $T = (E, \leq, \lambda)$ be a trace over (Σ, I). The finite prefixes of T, to be called configurations, will play a crucial role in what follows. A *configuration* of T is a finite subset of events $c \subseteq E$ with ${\downarrow}c = c$ where ${\downarrow}c = \bigcup_{e \in c} {\downarrow}e$. The set of configurations of T will be denoted $\mathcal{C}_T$. Trivially, $\emptyset \in \mathcal{C}_T$ is the case for any trace T. $\mathcal{C}_T$ can be equipped with a natural transition relation $\longrightarrow_T \subseteq \mathcal{C}_T \times \Sigma \times \mathcal{C}_T$ given by: $c \stackrel{a}{\longrightarrow}_T c'$ iff there exists an $e \in E$ such that $\lambda(e) = a$, $e \notin c$ and $c' = c \cup \{e\}$. Configurations of $\mathcal{C}_T$ are the trace-theoretic analogues of finite prefixes of strings. As will become apparent in Section 3, the formulas of LTL are to be interpreted at configurations of traces.

In its original formulation [Maz77], Mazurkiewicz introduced traces as certain equivalence classes of strings, and this correspondence turns out to be essential to our developments here. To bring this out, let Σ^* be the set of finite strings over Σ and Σ^ω be the set of (countably) infinite strings generated by Σ with $\omega = \{0, 1, 2, \dots\}$. We set $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$ and denote the empty word by ε. We let w, w' range over Σ^ω and u, v with or without primes range over Σ^*. Finally, we take $\mathrm{prf}(w)$ to be the set of finite prefixes of w and let $alph(w)$ denote the set of actions occurring in w.

Next, let $T = (E, \leq, \lambda) \in \mathbb{TR}(\Sigma, I)$. Then $w \in \Sigma^\infty$ is a *linearisation* of T iff there exists a map $\rho : \mathrm{prf}(w) \to \mathcal{C}_T$, such that the following conditions are met:

- $\rho(\varepsilon) = \emptyset$.
- $\rho(v) \stackrel{a}{\longrightarrow}_T \rho(va)$ for each $va \in \mathrm{prf}(w)$.
- For every $e \in E$, there exists some $u \in \mathrm{prf}(w)$ such that $e \in \rho(u)$.

The function ρ will be called a *run map* of the linearisation w. Note that the run map of a linearisation is unique. In what follows, we shall take $\mathrm{lin}(T)$ to be the *set of linearisations* of the trace T.

A set $p \subseteq \Sigma$ is called a *D-clique* iff $p \times p \subseteq D$. The equivalence relation $\approx\ \subseteq \Sigma^\infty \times \Sigma^\infty$ induced by I is given by: $w \approx w'$ iff $w{\restriction}p = w'{\restriction}p$ for every D-clique p. Here and elsewhere, if $X \subseteq \Sigma$, $w{\restriction}X$ is the sequence obtained by erasing from w all occurrences of letters in $\Sigma - X$. We take $[w]_\approx$ to denote the $\approx$-equivalence class of $w \in \Sigma^\infty$.

It is not hard to show that elements of $\mathbb{TR}(\Sigma, I)$ and $\approx$-equivalence classes are two representations of the same class of objects: A labelled partial-order $T \in \mathbb{TR}(\Sigma, I)$ is represented by $\mathrm{lin}(T)$ and vice versa (see [DR95] for a proof of this fact and a more thorough account of traces). We exploit this duality of representation and let T_w denote the trace corresponding to $[w]_\approx$. Moreover, for each $v \in \mathrm{prf}(w)$ we will use c_v to denote the configuration of $\mathcal{C}_{T_w}$ given by $\rho(v)$.

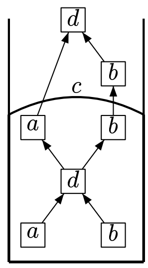

Fig. 1. A trace over (Σ, I).

To illustrate these concepts, consider the trace alphabet (Σ, I) with $\Sigma = \{a, b, d\}$ and $I = \{(a, b), (b, a)\}$. An example trace T over (Σ, I) is depicted in Figure 1 with smaller elements (with respect to $\leq$) appearing below larger elements. Furthermore, it can easily be verified that $abdbabd \in \mathrm{lin}(T)$ so $T = T_{abdbabd}$, but $adabbbd \notin \mathrm{lin}(T)$. The configuration $c \in \mathcal{C}_T$ consists of the first two a's, first d, first two b's and is also denoted by c_{abdab} which is identical to c_{badab} as $abdab \approx badab$.

We transfer considering traces as equivalence classes to the level of languages and call a word language $L \subseteq \Sigma^\omega$ *trace-consistent* if for all words $w, w' \in \Sigma^\omega$ with $w \approx w'$, it holds $w \in L$ iff $w' \in L$.

3 LTL for Mazurkiewicz Traces

In this section we bring out the syntax and semantics of the linear time temporal logic LTL, which will be our basic object of study. It was originally introduced for strings by Pnueli [Pnu77]. It was later equipped with a trace semantics [TW97] and proved expressively equivalent to first-order logic for traces by Diekert and Gastin [DG00], and this is the version we will consider here.

The formulas of LTL are parameterised by a trace alphabet (Σ, I) and are defined inductively as follows:

$$\mathrm{LTL}(\Sigma, I) ::= \mathrm{tt} \mid \neg\varphi \mid \varphi \vee \psi \mid \langle a \rangle \varphi \mid \varphi \mathcal{U} \psi \ , \quad a \in \Sigma.$$

Formulas of $\mathrm{LTL}(\Sigma, I)$ are interpreted over configurations of traces over (Σ, I). More precisely, given a trace $T \in \mathbb{TR}(\Sigma, I)$, a configuration $c \in \mathcal{C}_T$, and a formula $\varphi \in \mathrm{LTL}(\Sigma, I)$, the notion of $T, c \models \varphi$ is defined inductively via:

- $T, c \models \mathrm{tt}$.
- $T, c \models \neg\varphi$ iff $T, c \not\models \varphi$.
- $T, c \models \varphi \vee \psi$ iff $T, c \models \varphi$ or $T, c \models \psi$.
- $T, c \models \langle a \rangle \varphi$ iff there exists a $c' \in \mathcal{C}_T$ such that $c \xrightarrow{a}_T c'$ and $T, c' \models \varphi$.
- $T, c \models \varphi \mathcal{U} \psi$ iff there exists a $c' \in \mathcal{C}_T$ with $c \subseteq c'$ such that $T, c' \models \psi$ and all $c'' \in \mathcal{C}_T$ with $c \subseteq c'' \subset c'$ satisfy φ.

We will freely use the standard abbreviations such as e.g. $\mathrm{ff} = \neg\mathrm{tt}$, $\varphi \wedge \psi = \neg(\neg\varphi \vee \neg\psi)$. Furthermore, we sometimes abbreviate $T, \emptyset \models \varphi$ by $T \models \varphi$.

A simple example of a formula of LTL is $\varphi = \langle a \rangle \langle b \rangle \psi$. Note that for the trace of Figure 1 it holds that $T \models \varphi$ if and only if $T, c_{ab} \models \psi$. Moreover, φ is equivalent to $\varphi' = \langle b \rangle \langle a \rangle \psi$ over this particular trace alphabet because aIb, i.e., the models of φ and φ' and coincide. Such considerations will play a prominent role when we define the decision procedure in Section 5.

For bringing out the decision procedure itself, it will be convenient to assume that the syntax of LTL is augmented with an indexed until operator $\Phi \mathcal{U}^Z \psi$ where $\Phi = \{\varphi_1^{Y_1}, \ldots, \varphi_n^{Y_n}\}$ is a finite set of indexed formulas. Formally, it will have the following semantics:

- $T, c \models \Phi \mathcal{U}^Z \psi$ iff there exists a $c' \in \mathcal{C}_T$ with $c \subseteq c'$ such that $T, c' \models \psi$ and $\lambda(c' - c)IZ$, and, for each $1 \leq i \leq n$ and every c'' with $c \subseteq c'' \subset c'$ and $\lambda(c'' - c)IY_i$, it holds $c'' \models \varphi_i$.

Hence, a trace satisfies the formula $\Phi \mathcal{U}^Z \psi$ in the configuration c iff there is a future configuration c' satisfying ψ and all the actions from c to c' are independent from the actions in Z. Furthermore, the configurations between c and c' which can be reached from c by performing actions independent of Y_i all satisfy φ_i.

Note that $\varphi \mathcal{U} \psi$ can be identified with $\{\varphi^{\emptyset}\} \mathcal{U}^{\emptyset} \psi$ and we will not always make this distinction explicit. It's not hard to see that $\Phi \mathcal{U}^Z \psi$ is expressible within FO, so this indexed modality is derivable within LTL itself.

We remark that, in case of the empty independence relation, $\mathrm{LTL}(\Sigma, I)$ and LTL interpreted over words (denoted by $\mathrm{LTL}(\Sigma)$) coincide in the expected manner. Thus, we identify LTL over words with $\mathrm{LTL}(\Sigma)$, especially in the proof of Theorem 3, and retain us from introducing LTL over words formally.

4 Alternating Büchi Automata

Alternating automata extend nondeterministic automata by universal choices. The transition function denotes no longer a set of possible next states but a (positive) Boolean combination. In this section we recall the notion of alternating automata along the lines of [Var96] where alternating Büchi automata are used for model checking LTL over strings. However, we modified the definition of a run to reflect the ideas presented in [LT00].

For a finite set X of variables let $\mathcal{B}^+(X)$ be the set of *positive Boolean formulas* over X, i.e., the smallest set such that

- $X \subseteq \mathcal{B}^+(X)$
- $\mathrm{tt}, \mathrm{ff} \in \mathcal{B}^+(X)$
- $\varphi, \psi \in \mathcal{B}^+(X) \Rightarrow \varphi \wedge \psi \in \mathcal{B}^+(X), \varphi \vee \psi \in \mathcal{B}^+(X)$

In the following, we assume for every positive Boolean formula that it is in disjunctive normal form and that it is reduced with respect to idempotence and commutation. Hence, for a set X with $|X|$ elements the size of $\mathcal{B}^+(X)$ is bounded by $2^{2^{|X|}}$. This can easily be seen by considering the formulas as sets of sets.

We say that a set $Y \subseteq X$ *satisfies* (or is a *model* of) a formula $\varphi \in \mathcal{B}^+(X)$ iff φ evaluates to tt when the variables in Y are assigned to tt and the members of $X \backslash Y$ are assigned to ff. A model is called *minimal* if none of its proper subsets is a model. For example, $\{q_1, q_3\}$ as well as $\{q_2, q_3\}$ are minimal models of the formula $(q_1 \vee q_2) \wedge q_3$.

Later in our construction, logical formulas will take over the role of states. Therefore, we should formally distinguish between disjunction of formulas and disjunctions of states. However, to simplify our presentation, we identify these disjunctions when the context makes clear which one is meant. In particular, given a formula φ in disjunctive normal form, $\varphi = \bigvee \bigwedge \varphi_{ij}$ where every φ_{ij} is no (top level) disjunction or conjunction, we identify φ with the positive Boolean combination of states φ_{ij}. To avoid confusion, we sometimes write $\mathrm{st}(\varphi)$ to denote $\{\varphi_{ij} \mid \bigvee \bigwedge \varphi_{ij}\}$.

An *Alternating Büchi Automaton* (ABA) over an alphabet Σ is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, \mathcal{F})$ such that Q is a finite nonempty set of *states*, $q_0 \in Q$ is an *initial state*, $\mathcal{F} \subseteq Q$ is a set of accepting states and $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$ is a *transition function*.

Because of universal quantification, a *run* over an infinite string is no longer a sequence but a labelled directed acyclic graph. A node's label reflects one of the current states of the automaton, and the edges reflect transitions of the automaton with respect to the input string. Hence, this graph should have a unique "root" labelled with q_0. Furthermore, it has to be dividable into "levels" $i \in \mathbb{N}$ corresponding to the ith input letter. Every node except the root must have a "predecessor". For a node v, the labels of nodes of level $i+1$ connected with v should further be a model for the transition in state $l(v)$ reading the i's letter. More precisely:

A *run* over an infinite string $w = a_0 a_1 \ldots \in \Sigma^\omega$ is a Q–labelled directed acyclic graph (V, E) such that there exist labellings $l : V \to Q$ and $h : V \to \mathbb{N}$ which satisfy the following properties.

- $h^{-1}(0) = \{v\}$ with $l(v) = q_0$.
- $E \subseteq \bigcup_{i \in \mathbb{N}} (h^{-1}(i) \times h^{-1}(i+1))$.

- For every $v' \in V$ with $h(v') \geq 1$, $\{v \in V \mid (v, v') \in E\} \neq \emptyset$.
- For every $v, v' \in V$, $v \neq v'$, $l(v) = l(v')$ implies $h(v) \neq h(v')$.
- For every $v \in V$, $\{l(v') \mid (v, v') \in E\}$ is a minimal model of $\delta(l(v), a_{h(v)})$.

A run (V, E) is *accepting* if every maximal finite path ends in a node $v \in V$ with $\delta(l(v), a_{h(v)}) = \mathrm{tt}$ and every maximal infinite path, wrt. the labelling l, visits at least one final state infinitely often. The language $\mathcal{L}(\mathcal{A})$ of an automaton $\mathcal{A}$ is determined by all strings for which an accepting run of $\mathcal{A}$ exists.

Let us define a subclass of alternating automata corresponding to LTL–formulas over words, as shown in [LT00]. We call an alternating Büchi automaton $\mathcal{A}$ *linear* if for every of its runs (V, E), it holds: For all $n \geq 2$ and pairwise disjoint $v_1, \ldots, v_n \in V$ with $v_1 E v_2 \ldots E v_n$, $l(v_1) = l(v_n)$ implies $l(v_1) = l(v_2) = \ldots = l(v_n)$.

Finally, we call an alternating Büchi automaton $\mathcal{A}$ *trace-consistent* if its accepted language $\mathcal{L}(\mathcal{A})$ is trace-consistent. We will show that linear trace-consistent automata correspond to LTL–formulas over Mazurkiewicz traces.

Obviously, every Büchi automaton can be turned into an equivalent (wrt. the accepted language) alternating Büchi automaton. Vice versa, for every alternating Büchi automaton an equivalent Büchi automaton can be constructed with an exponential blow–up. The construction is described for example in [Var96]. Hence, it is easy to see that the emptiness problem for ABAs is exponential in its number of states.

5 A Decision Procedure for LTL

We have now set the scene to bring out our decision procedure for LTL. Our procedure generalises the classical approach by constructing an alternating Büchi automaton $\mathcal{A}_\varphi$ accepting the set of linearisations of traces satisfying a given formula φ. The states of this automaton are derived from an extended subformula closure, which we first define. Following this, we define a notion of independence-rewriting of such formulas, and this will eventually become the transition relation of $\mathcal{A}_\varphi$. We pin down the details of our construction and give a proof of correctness. Then we consider the complexity of our decision procedure and point out the correspondence between linear trace-consistent alternating automata and LTL over Mazurkiewicz traces.

5.1 The construction

In essence, we will construct an automaton $\mathcal{A}_\varphi$ that accepts a string $w \in \Sigma^\omega$ whenever the corresponding trace T_w satisfies φ. To appreciate the developments to come, we commence with a small example. Consider the example formula $\varphi = \langle a \rangle \langle b \rangle \psi$ of Section 3. Suppose that w is of the form abv for some $v \in \Sigma^\omega$. It is then not hard to see that $T_w, \emptyset \models \varphi$ if and only if $T_w, c_a \models \langle b \rangle \psi$.

Consider now instead $w' = bav$. Since the underlying domain is traces, $T_{w'}$ might still satisfy φ even though the first action is a b and not an a, because

aIb. In fact, $T_{w'}, \emptyset \models \varphi$ exactly when $T_{w'}, c_b \models \langle a \rangle \psi$. In this sense, the proof obligation at the empty configuration, "$\langle a \rangle \langle b \rangle \psi$", has been transformed by b to the proof obligation "$\langle a \rangle \psi$" at c_b; the a-action still has to be witnessed, but the present b has been matched. (Note by the way that either both or none of w and w' should be accepted, because $w \approx w'$ and hence $T_w = T_{w'}$.)

In effect, our automaton proceeds in this way by "independence-rewriting" the proof obligations by the actions read. The state space thus consists of all subformulas together with formulas obtained by transformations as described above. We will call this set the *extended closure* of φ.

Definition 1. *Let η be a formula of* LTL. *We take $ecl(\eta)$ to be the least set that satisfies the following:*

- *η itself is contained in its closure.*
- *For $\varphi \vee \psi \in ecl(\eta)$ it also contains the closure of φ and of ψ.*
- *For $\langle a \rangle \varphi \in ecl(\eta)$ it also contains the extended closure of φ as well as $\langle a \rangle \varphi'$ for every $\varphi' \in ecl(\varphi)$.*
- *For $\varphi \in ecl(\eta)$ it also contains $\neg \varphi \in ecl(\eta)$. We identify $\neg\neg\varphi$ with φ. Hence, $ecl(\eta)$ is closed under negation.*
- *For $\varphi \mathcal{U} \psi \in ecl(\eta)$ the closure contains $ecl(\varphi)$ as well as $ecl(\psi)$. Furthermore, for all $Z \subseteq \Sigma$, all $\psi' \in ecl(\psi)$, and all $\Phi \subseteq \{\varphi'^{Y} \mid \varphi' \in ecl(\varphi), Y \subseteq \Sigma\}$ the closure contains $\Phi U^Z \psi$.*
- *The closure is closed under positive Boolean combinations, i.e., $\mathcal{B}^+(ecl(\eta)) \subseteq ecl(\eta)$.*

Intuitively, the extended closure of a formula φ contains all formulas which may be obtained by substituting a subformula ψ of φ by ψ' where ψ' is a positive Boolean combination of formulas derived from ψ by applying this rule. This is because our automaton will be defined in the way that before it is considering a formula φ it may consider a subformula ψ of φ, transforming this into a positive Boolean combination of new formulas ψ'. This result is processed in the way that ψ is substituted by ψ' within φ.

We assume that all positive Boolean formulas are in disjunctive normal form and moreover that they are reduced wrt. idempotence and commutation. With these assumptions we can prove the following crucial result.

Proposition 1. *$ecl(\eta)$ is a finite set for each formula η of* LTL.

Proof. The proof proceeds by a standard induction. The claim is obvious for atomic formulas. For $\eta = \langle a \rangle \varphi$, the extended closure of η contains the extended closure of φ and for every element φ' of $ecl(\varphi)$ also $\langle a \rangle \varphi'$. Thus we get $|ecl(\varphi)| \cdot 2$ elements. Since the extended closure contains for every element also a negated one, we get another factor 2. Now, positive Boolean combination yields a double exponential blow–up. Altogether, we have $|ecl(\eta)| \leq 2^{2^{|ecl(\varphi)| \cdot 2 \cdot 2}}$. For $\eta = \{\varphi_1^{Y_1}, \ldots, \varphi_n^{Y_n}\} \mathcal{U}^Z \psi$ it can be verified that $|ecl(\eta)|$ is bounded by

$$2^{2^{\left(2^{2^{\sum_{i=1}^{n}(|ecl(\varphi_i)| \cdot 2^{|\Sigma|})}} \cdot 2^{|\Sigma|} \cdot 2^{2^{|ecl(\psi)|}}\right) \cdot 2}}$$

The three factors are upper bounds for the derivatives of $\{\dots\}$, $\mathcal{U}^Z$, and ψ, resp., and the powers bound their positive Boolean combination. $\square$

We will refer to formulas of this set as *extended formulas*. Furthermore, we will say that a formula is a *diamond-formula* in case it is of the form $\langle a\rangle\varphi$ for some extended formula φ and some $a \in \Sigma$. In a similar vein, we let the *until-formulas* consist of those of the form $\Phi\mathcal{U}^Z\psi$ with Φ being a finite set of extended formulas, ψ a single extended formula and $Z \subseteq \Sigma$.

For extended formulas we will make use of the important notion of its *dual*, which is obtained as usual by applying de Morgan's laws to push negations inwards as far as possible.

Definition 2. *The* dual *of an (extended) formula is given inductively as follows:*

- $\overline{\mathrm{tt}} = \mathrm{ff}$, $\overline{\mathrm{ff}} = \mathrm{tt}$.
- $\overline{\neg\varphi} = \varphi$.
- $\overline{\varphi \vee \psi} = \overline{\varphi} \wedge \overline{\psi}$, $\overline{\varphi \wedge \psi} = \overline{\varphi} \vee \overline{\psi}$.
- $\overline{\langle a\rangle\varphi} = \neg\langle a\rangle\varphi$.
- $\overline{\varphi\mathcal{U}\psi} = \neg(\varphi\mathcal{U}\psi)$.
- $\overline{\Phi\mathcal{U}^Z\psi} = \neg(\Phi\mathcal{U}^Z\psi)$.

We are now set to introduce the operator $||\_||_\_$ which will constitute the transition relation of the alternating automaton. Essentially, $||\varphi||_a$ is to be thought of as the independence-rewriting of φ by the action a.

It follows from the intuition conveyed earlier that it should be the case that $||\langle a\rangle\varphi||_a$ is φ. Then, for the case where aIb, $||\langle b\rangle\varphi||_a = \langle b\rangle\varphi'$ where $\varphi' = ||\varphi||_a$. Of course, whenever aDb and the actions are not identical, then $||\langle b\rangle\varphi||_a$ must be ff because b cannot be a next action. Definition 3 formally captures this intuition.

Definition 3. *For each extended formula φ and each action a, the operator $||\varphi||_a$ yields a formula of $\mathcal{B}^+(ecl(\varphi))$ and is defined inductively via:*

$$\begin{array}{ll}
||\mathrm{tt}||_a & = \mathrm{tt}. \\
||\varphi \vee \psi||_a & = ||\varphi||_a \vee ||\psi||_a. \\
||\neg\varphi||_a & = \overline{||\varphi||_a}. \\
||\langle b\rangle\varphi||_a & = \begin{cases} \varphi & \text{if } a = b \\ \langle b\rangle||\varphi||_a & \text{if } aIb \\ \mathrm{ff} & \text{if } aDb, a \neq b \end{cases} .
\end{array}$$

Note that since $ecl(\eta)$ is closed under positive Boolean combination, we have $ecl(\eta) = \mathcal{B}^+(ecl(\eta))$.

We now only need to specify the case of $||\Phi\mathcal{U}^Z\psi||_a$. This turns out to be inherently more complex, and before providing the precise definition, we carefully analyse the semantics of the indexed until modality in Figure 2. For this purpose, consider some trace T be given and suppose $c, c' \in \mathcal{C}_T$ such that $c \subseteq c'$. Furthermore, let c'' be a configuration between c and c' (Figure 2 (i)).

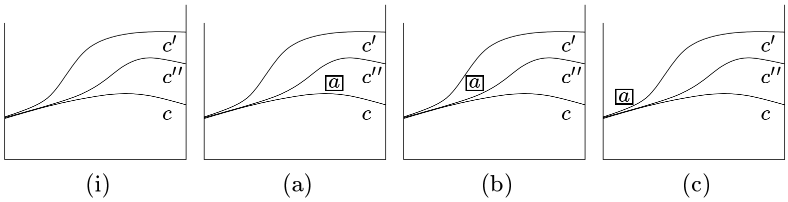

Fig. 2. Configuration and actions.

Suppose, we can augment c by an a-labelled event e to obtain a successor configuration c''' of c, i.e., $c \xrightarrow{a}_T c'''$. Then $c''' \subseteq c'' \subseteq c'$ or $c''' \not\subseteq c''$ but $c''' \subseteq c'$ or $c''' \not\subseteq c'$, as shown in Figure 2, resp. (a) – (c). In Case (b), it is obvious that $\lambda(c'' - c)Ia$ and for Case (c), we have $\lambda(c' - c)Ia$ (as well as $\lambda(c'' - c)Ia$).

The situation shown in Figure 2 can, in other words, be described in the following manner: The action a is

(a) neither in the future of c'' nor of c' (Case (a)),
(b) in the future of c'' (Case (b)), or
(c) in the future of c'' as well as of c' (Case (c)).

Consider a formula $\varphi\mathcal{U}\psi$ which is to be checked in the configuration c. In Case (c), we have to employ a for verifying ψ as well as φ. Note that for (c) we get two subcases depending upon whether $c' = c$ or $c' \supset c$. While ψ is not relevant to the first case, φ is required to hold in the configurations between c and c'. Note that these configurations are reached by actions independent of a.

For Case (a), we have to employ a for verifying φ in configuration c but not for c''. In Case (b), we have to prove φ considering a in the configuration c'' which might be equal to c as well as different from c. Note that in the latter case, every event of $c'' - c$ is independent of a.

Consequently, we define the rewriting operator for a formula $\Phi\mathcal{U}^Z\psi$ as follows.

Definition 4. *(extends Definition 3)*
Let

$$\Psi_1 = ||\psi||_a \quad \Psi_2 = \{||\varphi||_a^{Y\cup\{a\}} \mid \varphi^Y \in \Phi\}\mathcal{U}^{Z\cup\{a\}}||\psi||_a.$$

Moreover, we set $\Psi' = \Psi_1 \vee \Psi_2$. *Let*

$$\begin{aligned} ||\Phi||_a = \{\ & ||\varphi||_a^{Y\cup\{a\}} \mid \varphi^Y \in \Phi\} \\ \cup \{\ & \varphi^Y \qquad\qquad \mid \varphi^Y \in \Phi, aIY\} \end{aligned}$$

and

$$\Phi_1 = \bigwedge_{\varphi^Y \in \Phi} ||\varphi||_a \quad \Phi_2 = ||\Phi||_a\mathcal{U}^Z\psi$$

and $\Phi' = \Phi_1 \wedge \Phi_2$.

Then we define

$$||\Phi\mathcal{U}^Z\psi||_a = \begin{cases} \Psi' & \textit{if } aDZ \\ \Psi' \vee \Phi' & \textit{else } aIZ \end{cases}.$$

Note that Ψ' captures Case (c) in which an action a is employed for verifying ψ under the assumption that $c' = c$ (Ψ_1) or not (Ψ_2). Φ' covers the idea that a is not in the future of c' but is employed for verifying the obligations in Φ.

It's not hard to verify that $||\_||\_$ is well-defined. However, to show the correctness of our construction, the following proposition is essential. Suppose that we have given one linearisation of a trace and one formula and we want to check the formula wrt. the trace obtained from the linearisation. According to the following proposition it is possible to consider the word action by action and modifying the formula according to the rewriting operator.

Proposition 2. *Let η be any formula of* $\mathrm{LTL}(\Sigma, I)$*. Then for all $w \in \Sigma^\omega$ and for all $v \in \Sigma^*, a \in \Sigma, w' \in \Sigma^\omega$ with $vaw' \approx w$*

$$T_w, c_v \models \eta \textit{ if and only if } T_w, c_{va} \models ||\eta||_a$$

Proof. The proof proceeds by induction on the formula η. We only show the most important cases as the other cases follow in a similar manner. The cases where $\eta = \mathrm{tt}$ or $\eta = \mathrm{ff}$ are trivial.

Suppose $\eta = \varphi \vee \psi$. Then $T_w, c_v \models \varphi \vee \psi$ means by definition that $T_w, c_v \models \varphi$ or $T_w, c_v \models \psi$. By induction, this is equivalent to $T_w, c_{va} \models ||\varphi||_a$ or $T_w, c_{va} \models ||\psi||_a$ which is equivalent to $T_w, c_{va} \models ||\varphi \vee \psi||_a$ by definition of the rewrite operator.

Suppose $\eta = \neg\varphi$. By definition, $T_w, c_v \models \neg\varphi$ iff not $T_w, c_v \models \varphi$. Induction yields $T_w, c_{va} \not\models ||\varphi||_a$ which means $T_w, c_{va} \models \neg||\varphi||_a$. The dual of a formula is obviously logically equivalent to the negation of the formula so that the previous statement is equivalent to $T_w, c_{va} \models \overline{||\varphi||_a}$.

Suppose $\eta = \langle b \rangle\varphi$. By definition, $T_w, c_v \models \langle b \rangle\varphi$ if and only if there is a configuration c' such that $c_v \xrightarrow{b}_T c' = c_{vb}$ and $T_w, c' \models \varphi$. We consider three different cases:

- $b = a$: Then $c' = c_{va}$. Hence $T_w, c_{va} \models \varphi$
- $b \neq a, bDa$: Then $c_v \xrightarrow{b}_T c'$ and $c_v \xrightarrow{a}_T c_{va}$. However, then w is not a linearisation of the trace which is a contradiction.
- bIa: $T_w, c_{vb} \models \varphi$ is by induction equivalent to $T_w, c_{vba} \models ||\varphi||_a$. Since aIb, this means $T_w, c_{vab} \models ||\varphi||_a$ which is equivalent to $T_w, c_{va} \models \langle b \rangle||\varphi||_a$.

Putting together all the cases we get that $T_w, c_v \models \langle b \rangle\varphi$ if and only if $T_w, c_{va} \models ||\langle b \rangle\varphi||_a$.

The most involved case is $\eta = \Phi\mathcal{U}^Z\psi$. Let $\Phi = \{\varphi_1^{Y_1}, \dots, \varphi_N^{Y_N}\}$. Recall that $T_w, c_v \models \{\varphi_1^{Y_1}, \dots, \varphi_N^{Y_N}\}\mathcal{U}^Z\psi$ if and only if

$$\exists x \in \Sigma^*, y \in \Sigma^\omega, aw' \approx xy, xIZ, \text{ such that } T_w, c_{vx} \models \psi, \text{ and}$$
$$\forall i \in \{1, \dots, N\}, \forall x_1, x_2 \in \Sigma^* \text{ satisfying } x_1x_2 \approx x, x_1IY_i, x_2 \neq \varepsilon, \text{ it holds}$$
$$T_w, c_{vx_1} \models \varphi_i.$$

We consider here only the case where $aIY_i \cup Z$. The other cases follow similarly. Let us first discuss the "*if*"–part: We consider the following cases for x:

- $x = \varepsilon$: Then $T_w, c_{vx} \models \psi$ means $T_w, c_v \models \psi$ which implies by induction $T_w, c_{va} \models ||\psi||_a$. This shows (Ψ_1).
- $x \neq \varepsilon, a \notin alph(x)$:
  We consider the cases for ψ and φ_i simultaneously.

$\Rightarrow aIx$	$\Rightarrow aIx_1$
$T_w, c_{vx} \models \psi$	$T_w, c_{vx_1} \models \varphi_i$
$\overset{I.H.}{\Rightarrow} T_w, c_{vxa} \models \lVert\psi\rVert_a$	$\overset{I.H.}{\Rightarrow} T_w, c_{vx_1a} \models \lVert\varphi_i\rVert_a$
$\Rightarrow T_w, c_{vax} \models \lVert\psi\rVert_a$	$\Rightarrow T_w, c_{vax_1} \models \lVert\varphi_i\rVert_a$
$\Rightarrow \exists x \in \Sigma^*, xI(Z \cup \{a\}),$	$\Rightarrow \forall x_1, x_2 \in \Sigma^*, x_1x_2 \approx x, x_1I(Y_i \cup \{a\}),$
$T_w, c_{vax} \models \lVert\psi\rVert_a.$	$x_2 \neq \varepsilon, T_w, c_{vax_1} \models \lVert\varphi_i\rVert_a.$

  Hence, $T_w, c_{va} \models \{||\varphi||_a^{Y\cup\{a\}} \mid \varphi^Y \in \Phi\}\mathcal{U}^{Z\cup\{a\}}||\psi||_a$ which shows (Ψ_2).
- $x \neq \varepsilon, a \in alph(x)$:
  We easily see that $x \approx ax'$ and $a, x'IZ$ and $T_w, c_{vax'} \models \psi$. We will show (Φ_1) and (Φ_2). Let us consider x_1. If $x_1 = \varepsilon$ then $T_w, c_v \models \varphi_i$ implies by induction $T_w, c_{va} \models ||\varphi_i||_a$. If $x_1 \neq \varepsilon$ and $a \notin alph(x_1)$, we see that aIx_1 and x_1IY_i. Hence, $x_1I(Y_i \cup \{a\})$. Now, $T_w, c_{vx_1} \models \varphi_i$ yields by induction $T_w, c_{vx_1a} \models ||\varphi_i||_a$ proving $T_w, c_{vax_1} \models ||\varphi_i||_a$ since aIx_1. For the case $x_1 \neq \varepsilon$ but $a \in alph(x_1)$ we see that $x_1 \approx ax_1'$ and $T_w, c_{vax_1'} \models \varphi_i$. Summing up the cases for x_1 we get $T_w, c_{va} \models ||\Phi||_a\mathcal{U}^Z\psi$, which shows (Φ_2), and $T_w, c_{va} \models ||\varphi||_a$ for all $\varphi^Y \in \Phi$ which shows (Φ_1).

All together, we showed that Ψ' or Φ' hold in the until case proving the "*if*"–part. Now, let us consider the "*only–if*" part: Suppose $T_w, c_{va} \models ||\Phi\mathcal{U}^Z\psi||_a$, i.e.

$$T_w, c_{va} \models \Psi_1 \vee \Psi_2 \vee \Phi'.$$

We discuss the disjunction by drawing the conclusions of each formula

- $T_w, c_{va} \models ||\psi||_a$:
  This implies by induction that $T_w, c_v \models \psi$. Hence, $T_w, c_v \models \Phi\mathcal{U}^Z\psi$.
- $T_w, c_{va} \models \{||\varphi||_a^{Y\cup\{a\}} \mid \varphi^Y \in \Phi\}\mathcal{U}^{Z\cup\{a\}}||\psi||_a$:
  Then there exist x, y, $xI(Z\cup\{a\})$, $w \approx vaxy$ such that $T_w, c_{vax} \models ||\psi||_a$. Since xIa also $T_w, c_{vxa} \models ||\psi||_a$ which yields by induction $T_w, c_{vx} \models \psi$. We further know that for every proper prefix (modulo $\approx$) x_1 of x with $x_1I(Y \cup \{a\})$ we have $T_w, c_{vax_1} \models ||\varphi||_a$. Then $T_w, c_{vx_1a} \models ||\varphi||_a$ and, by induction, $T_w, c_{vx_1} \models \varphi$. Hence, $T_w, c_v \models \Phi\mathcal{U}^Z\psi$.
- $T_w, c_{va} \models \bigwedge_{\varphi^Y\in\Phi} ||\varphi||_a \wedge ||\Phi||_a\mathcal{U}^Z\psi$:
  We first obtain by induction that $T_w, c_v \models \varphi$ for every $\varphi^Y \in \Phi$. Let us consider

$$T_w, c_{va} \models (\{||\varphi||_a^{Y\cup\{a\}} \mid \varphi^Y \in \Phi\} \cup \{\varphi^Y \mid \varphi^Y \in \Phi, aIY\})\mathcal{U}^Z\psi$$

It implies that there is an x', independent on Z, such that $T_w, c_{vax'} \models \psi$. Since we are in the case of aIZ, we conclude that there is an x, xIZ ($x \approx ax'$) such that $T_w, c_{vx} \models \psi$. Now, consider $x_1x_2 \approx x$, $x_2 \neq \varepsilon$. For every $x_1I(Y \cup \{a\})$, x_1 a prefix of x', we know $T_w, c_{vax_1} \models ||\varphi||_a$ and, by induction, $T_w, c_{vx_1} \models \varphi$. For $x' = \varepsilon$ we already know $T_w, c_v \models \varphi$. For x_1IY and x_1Da we obtain $x_1 \approx ax'_1$, x'_1IY, since c_{va} is a valid configuration. By $T_w, c_{vax'_1} \models \varphi$ we deduce $T_w, c_{vx_1} \models \varphi$. Altogether, this shows $T_w, c_v \models \Phi\mathcal{U}^Z\psi$.

This concludes the proof. □

We can now finally bring the definition of the alternating Büchi automaton $\mathcal{A}_\varphi$ corresponding to a formula $\varphi \in \mathrm{LTL}(\Sigma, I)$ as follows.

Definition 5. *Given a formula $\varphi \in \mathrm{LTL}(\Sigma, I)$, the alternating Büchi automaton $\mathcal{A}_\varphi$ is the tuple $(Q, \Sigma, \delta, q_0, F)$ where*

- *$Q = ecl(\varphi)$ is the set of states.*
- *$\delta(q, a) = ||q||_a$ is the transition function.*
- *$q_0 = \varphi$ is the initial state.*
- *$F = \{\neg\psi \mid \neg\psi \in ecl(\varphi)\}$ is the set of accepting states.*

Note that we defined the set of final states to be all negated formulas. The intuitive idea is that failing to prove a proposition infinitely often suffices to assume that its negation is true. In the work of Vardi [Var96], one could likewise take all negated formulas as final states. Since in the case of LTL over words, only *until*-formulas may occur infinitely often, the set of final states is there restricted to negated until-formulas.

The correctness of the construction is summarised in the following theorem, which is the main contribution of the paper.

Theorem 1. *Let φ be a formula of $\mathrm{LTL}(\Sigma, I)$ and let its alternating Büchi automaton be given as $\mathcal{A}_\varphi = (Q, \Sigma, \delta, q_0, F)$. Then*

$$w \in \mathcal{L}(\mathcal{A}_\varphi) \text{ if and only if } T_w, \emptyset \models \varphi$$

for every $w \in \Sigma^\omega$.

Proof. For $w \in \Sigma^\omega$ we have to show that $\mathcal{A}_\varphi$ has an accepting run on w iff $T_w \models \varphi$. Note that every run has (at most) three types of paths

- finite paths ending in tt
- infinite paths on which from some point on every node is labelled by an *until- or diamond*-formula, or
- infinite paths on which from some point on every node is labelled by a negated *until- or diamond*-formula.

Let us give a sketch of the proof. For $\psi \in ecl(\varphi)$ and $w = aw' \in \Sigma^\omega$, let $\bar{\delta}(\psi, w)$ be the extension of δ defined by $\bar{\delta}(\psi, aw') = \bar{\delta}(\delta(\psi, a), w')$. By Proposition 2, $\bar{\delta}(\varphi, w) = \bar{\delta}(\delta(\varphi, a), w') = \bar{\delta}(||\varphi||_a, w')$ and $T_w, c_\epsilon \models \varphi$ iff $T_w, c_a \models ||\varphi||_a$. Now, consider an accepting run of $\mathcal{A}_\varphi$. Its finite paths end in tt, thus all proof obligations are proved. Conversely, a run should be accepted only if the finite paths end in tt, i.e., that all proof obligations are proved. Now, let us consider the infinite paths of a run. These can only occur by unwinding a (negated) until–formula infinitely often or by reading actions independent of the one given within a diamond–formula. This can be accepted iff the underlying until–formula or diamond–formula is preceded by a negation. This is captured by the acceptance condition for infinite paths given by the final states of the automaton. $\square$

5.2 State space

To simplify the presentation, we have defined the state space of our automaton in a straightforward manner and presented a simple argument to show that it is finite. Thus, we indeed obtained a decision procedure. Now, let us take a closer look to the states which are really needed in our construction. In other words, let us consider the states which are reachable from the initial state.

Given a formula $\varphi \in \mathrm{LTL}(\Sigma, I)$, a state ψ of $\mathcal{A}_\varphi$, and a set $Y \subseteq \Sigma$, let $reach_Y(\psi)$ denote the set of states reachable from ψ in $\mathcal{A}_\varphi$ via words whose actions are independent of Y. More precisely,

$$reach_Y(\psi) = \{\psi' \mid \exists w \in \Sigma^*,\ wIY :\ \psi' \in st(\bar{\delta}(\psi, w))\},$$

where $\bar{\delta}$ is the extension of δ defined in the obvious manner.

Proposition 3. *Given $\varphi \in \mathrm{LTL}(\Sigma, I)$, we get upper bounds for the number of states reachable from a state of $\mathcal{A}_\varphi$ wrt. Y inductively as follows:*

- $|reach_Y(\mathrm{tt})| = 1$
- $|reach_Y(\mathrm{ff})| = 1$
- $|reach_Y(\neg\psi)| = |reach_Y(\psi)|$
- $|reach_Y(\psi_1 \vee \psi_2)| \leq |reach_Y(\psi_1)| + |reach_Y(\psi_2)|$
- $|reach_Y(\psi_1 \wedge \psi_2)| \leq |reach_Y(\psi_1)| + |reach_Y(\psi_2)|$
- $|reach_Y(\langle a\rangle\psi)| \leq \begin{cases} |reach_Y(\psi)| + |reach_{Y\cup\{a\}}(\psi)| + 1 & \text{if } aIY \\ |reach_{Y\cup\{a\}}(\psi)| + 1 & \text{if } aDY \end{cases}$
- $|reach_Y(\{\varphi_1^{Y_1}, \dots, \varphi_n^{Y_n}\}\mathcal{U}^Z\psi)| \leq 2^{2^{\sum_{i=1}^n(|reach_Y(\varphi_i)|\cdot 2^{|\Sigma|})}} \cdot 2^{|\Sigma|} \cdot 2^{2^{|reach_Y(\psi)|}}$

Proof. The obvious cases are if the state formula is tt or ff.

Since negation is shifted inwards by the dual operator $^-$, the states reachable from $\neg\psi$ are the same states as reachable from ψ, except that every state is preceded by $\neg$. Thus, the cardinality is the same.

Given $\langle a\rangle\psi$, assume a to be independent of Y. Reading an action dependent on but different from a (and independent of Y) yields the state ff and in our formula the 1. Reading a yields the state ψ, thus, the states reachable from ψ are obviously reachable from $\langle a\rangle\psi$ ($|reach_Y(\psi)|$). The last possibility is reading an action b independent of a and Y. This yields formulas of the form $\langle a\rangle\psi'$ where ψ' is obtained by rewriting ψ by actions independent of Y and a. Since $\langle a\rangle\psi'$ distributes over disjunctions and conjunctions, we get the same number of states as obtained by considering the states reachable from ψ by words independent of $Y \cup \{a\}$ ($|reach_{Y\cup\{a\}}(\psi)|$).

The bound for until-formulas follows by a simple combinatorial argument. Before and after the $\mathcal{U}$ within an until-formula, only positive Boolean combinations of derivations of respectively Φ and ψ may occur, and the $\mathcal{U}$ is indexed with subsets of Σ. □

Let us call the fragment of LTL defined without until-formulas *Hennessy-Milner fragment.*

Proposition 4. *For a state ψ from the Hennessy-Milner fragment of* $\mathrm{LTL}(\Sigma, I)$*, we obtain $reach_Y(\psi) \leq |\psi|^{|\Sigma - Y|}$.*

Proof. The proof follows a simple induction of which we pick out two cases:

- Applying Proposition 3, the induction hypothesis, and the binomial formula[1],
$$\begin{aligned} |reach_Y(\psi_1 \vee \psi_2)| &\leq |reach_Y(\psi_1)| + |reach_Y(\psi_2)| \\ &\leq |\psi_1|^{|\Sigma-Y|} + |\psi_2|^{|\Sigma-Y|} \\ &\leq (|\psi_1| + |\psi_2| + 1)^{|\Sigma-Y|} \end{aligned}$$
- Assuming aIY,
$$\begin{aligned} |reach_Y(\langle a\rangle\psi)| &\leq |reach_Y(\psi)| + |reach_{Y\cup\{a\}}(\psi)| + 1 \\ &\leq |\psi|^{|\Sigma-Y|} + |\psi|^{|\Sigma-Y|-1} + 1 \\ &\leq (|\psi| + 1)^{|\Sigma-Y|} \end{aligned}$$
□

Due to the exponential blow–up the construction of an equivalent Büchi automaton for $\mathcal{A}_\varphi$ causes, we conclude

Theorem 2. *Checking satisfiability of a formula from the Hennessy-Milner fragment of* $\mathrm{LTL}(\Sigma, I)$ *can be done in exponential time.*

5.3 LTL and linear automata

Now we characterise $\mathrm{LTL}(\Sigma, I)$ as equivalent to that subclass of alternating Büchi automata which we called trace-consistent linear alternating Büchi automata, and we start observing the linearity of the above construction.

Proposition 5. *Given $\varphi \in \mathrm{LTL}(\Sigma, I)$, $\mathcal{A}_\varphi$ is linear.*

[1] $(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^{n-i} b^i$

Proof. We have to show that every path of every run only contains trivial cycles. Therefore we define a well-founded strict ordering relation[2] $\prec$ on the states of our automaton and show that $||\psi||_a$ yields a Boolean combination of strictly smaller states or ψ.

For a formula $\eta \in \mathrm{LTL}(\Sigma, I)$, $\prec \subseteq ecl(\eta) \times ecl(\eta)$ is inductively defined by

- $\varphi \prec \langle a \rangle \varphi$,
- $\langle a \rangle \varphi \prec \langle a \rangle \psi$ if $\varphi \prec \psi$,
- $\overline{\varphi} \prec \neg \psi$ if $\varphi \prec \psi$,
- $\psi_1^{Y_1} \prec \psi_2^{Y_2}$ if $\psi_1 \preceq \psi_2$ and $Y_1 \supseteq Y_2$ and one of the orderings is strict, i.e., $\psi_1 \prec \psi_2$ or $Y_1 \supsetneq Y_2$,
- $\bigvee \bigwedge \varphi_{ij} \prec \bigvee \bigwedge \psi_{ij}$ if $\{\varphi_{ij}\} \ll \{\psi_{ij}\}$ where $\ll$ is the (strict) (multi-)set ordering induced by $\prec$, i.e., $M_1 \ll M_2$ iff there exist a set X and an element $m \in M_2$ with $m' \prec m$ for all $m' \in X$ such that $M_1 = (M_2 - \{m\}) \cup X$. In other words, a set M_1 is smaller than M_2 if an element of M_2 is replaced by a set of smaller elements resulting in M_1.
- $\psi' \prec \Phi \mathcal{U}^Z \psi$ if $\psi' \prec \psi$,
- $\bigvee \bigwedge \varphi_{ij} \prec \Phi \mathcal{U}^Z \psi$ if $\{\varphi_{ij}^{\Sigma}\} \ll \Phi$,
- $\Phi_1 \mathcal{U}^{Z_1} \psi_1 \prec \Phi_2 \mathcal{U}^{Z_2} \psi_2$ if $\Phi_1 \underline{\ll} \Phi_2$ and $Z_1 \supseteq Z_2$ and $\psi_1 \preceq \psi_2$ and one of the orderings is strict, i.e., $\Phi_1 \ll \Phi_2$ or $Z_1 \supsetneq Z_2$ or $\psi_1 \prec \psi_2$, where $\underline{\ll}$ is the reflexive closure of $\ll$,

and contains its transitive closure. Here, $\preceq$ is the reflexive closure of $\prec$.

We easily verify that, given formulas $\varphi, \psi \in ecl(\eta)$, an action $a \in \Sigma$, and a minimal model Ψ of $||\psi||_a$ with $\varphi \in \Psi$, it holds $\varphi \preceq \psi$ and furthermore that for arbitrary $\varphi, \psi \in ecl(\eta)$, $\varphi \prec^+ \psi$ implies $\varphi \neq \psi$. We conclude the linearity of our construction. □

Theorem 3. *Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a trace-consistent linear alternating Büchi automaton. There is a formula $\varphi \in \mathrm{LTL}(\Sigma, I)$ such that*

$$T_w, \emptyset \models \varphi \text{ if and only if } w \in \mathcal{L}(\mathcal{A}_\varphi)$$

for every $w \in \Sigma^\omega$.

Before we are going to prove the previous theorem, let us mention two facts:

Proposition 6. *Let $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ be a linear alternating Büchi automaton. There is a formula $\varphi \in \mathrm{LTL}(\Sigma)$ such that $\mathcal{L}(\varphi) = \mathcal{L}(\mathcal{A})$.*

Proposition 7. *Let $L \subseteq \Sigma^\omega$ and $I \subseteq \Sigma \times \Sigma$ be an independence relation. Then the following statements are equivalent.*

1) L is trace-consistent wrt. I and LTL(Σ)*-definable.*
2) $\{T_w \mid w \in L\}$ is FO(Σ, I)-definable.

[2] i.e. a transitive and acyclic relation

3) $\{T_w \mid w \in L\}$ *is* LTL(Σ, I)*-definable.*

Proposition 6 was independently shown by [Roh97] and [LT00]. As aforementioned in the introduction, the equivalence of 1) and 2) traces back to [EM96], that one between 2) and 3) back to [DG00].

Now, we are ready to prove Theorem 3.

Proof. In accordance with proposition 6, given a trace-consistent linear alternating Büchi automaton $\mathcal{A}$, there is a formula $\psi \in \mathrm{LTL}(\Sigma)$ satisfying $\mathcal{L}(\psi) = \mathcal{L}(\mathcal{A})$ where $\mathcal{L}(\psi)$ is likewise trace-consistent. Employing the equivalences from proposition 7, it immediately follows the existence of a formula $\varphi \in \mathrm{LTL}(\Sigma, I)$ with $T_w, \emptyset \models \varphi$ if and only if $w \in \mathcal{L}(\mathcal{A})$ for every $w \in \Sigma^\omega$. □

Let us bring out two important consequences of the last theorem:

1. Given an LTL–formula φ over Mazurkiewicz traces, it is simple to construct a trace-consistent LTL–formula ψ over words defining the same set of ω–words. Just construct $\mathcal{A}_\varphi$, and for $\mathcal{A}_\varphi$ a corresponding formula ψ according to the proof given in [LT00].
2. Partial order reduction techniques work for LTL over Mazurkiewicz traces. Given an LTL–formula φ over Mazurkiewicz traces, consider its automaton $\mathcal{A}_\varphi$. It is a linear trace-consistent automaton over words. For this kind automata, several powerful partial–order reduction techniques have been developed, which will have the same success here [Pel98]. Hence, specifying with LTL over Mazurkiewicz traces promises—despite the bad worst–case runtime of its decision procedure—efficient verification tasks in practice.

Note that the first item even implies that the languages definable by LTL-formulas over Mazurkiewicz traces are FO-definable over Mazurkiewicz traces. Thus, we obtained one direction of the expressive completeness proof of [DG00].

6 Conclusion

We have exhibited a decision procedure demonstrating that the classical automata-theoretic approach can be generalised to configuration based temporal logics for traces such as LTL. In particular, Theorem 1 asserts that it is possible to directly construct an alternating Büchi automaton accepting the set of linearisations of traces satisfying the formula at hand.

The main idea underlying this construction is to use a notion of independence-rewriting to an extended subformula closure. It easily follows from [Wal98] that this closure must be of nonelementary size and, moreover, that this is unavoidable for any decision procedure directly generalising the classical automata-theoretic approach.

Our approach clearly yields an optimal (non-elementary) decision procedure and shares this similarity with [GMP98b]. We are sure that an actual implementation of our approach would compare favourably due to the fact that the automata need not necessarily be constructed in full and especially because it avoids an exponential blow-up for negation.

We showed that trace-consistent linear automata correspond to LTL over Mazurkiewicz traces wrt. language definability, transferring a similar result shown in [LT00] from the setting of words to the setting of traces. As a consequence, it is quite natural to construct trace-consistent linear automata as a tool to answer the satisfiability problem, and our approach exactly follows this idea.

It is easy to adapt our decision procedure for the unary fragment of LTL, where the until-modality is replaced by an eventually-modality with the obvious semantics. Unfortunately, it remains an open question whether the decision procedure obtained in this way is optimal.

References

[APP95] Rajeev Alur, Doron Peled, and Wojciech Penczek. Model-checking of causality properties. In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science*, pages 90–100, San Diego, California, 26–29 June 1995. IEEE Computer Society Press.

[BVW94] O. Bernholtz, M. Y. Vardi, and P. Wolper. An automata–theoretic approach to branching–time model checking. In D.L. Dill, editor, *Proceedings of the 6th International Conference on Computer–Aided Verification (CAV'94)*, volume 818 of *Lecture Notes in Computer Science*, pages 142–155. Springer–Verlag, 1994.

[DG00] Volker Diekert and Paul Gastin. LTL is expressively complete for mazurkiewicz traces. In *Proceedings of 27th International Colloquium on Automata, Languages and Programming (ICALP'2000)*, volume 1853 of *LNCS*, Springer-Verlag, 2000.

[DR95] Volker Diekert and Grzegorz Rozenberg, editors. *The Book of Traces*. World Scientific, Singapore, 1995.

[EM96] Werner Ebinger and Anca Muscholl. Logical definability on infinite traces. *Theoretical Computer Science*, 154(1):67–84, January 1996.

[GHP97] Jean-Charles Grégoire, Gerard J. Holzmann, and Doron A. Peled, editors. *The Spin Verification System*, volume 32 of *DIMACS series*. American Mathematical Society, 1997. ISBN 0-8218-0680-7, 203p.

[GMP98a] P. Gastin, R. Meyer, and A. Petit. A (non-elementary) modular decision procedure for LTrL. In *MFCS: Symposium on Mathematical Foundations of Computer Science*, volume 1450 of *Lecture Notes in Computer Science*, 1998.

[GMP98b] P. Gastin, R. Meyer, and A. Petit. A (non-elementary) modular decision procedure for LTrL. Technical report, LSV, ENS de Cachan, 1998. extended version of MFCS'98.

[HJJK95] J. G. Henriksen, J. Jensen, M. Joergensen, and N. Klarlund. MONA: Monadic second-order logic in practice. *Lecture Notes in Computer Science*, 1019:89–??, 1995.

[Kam68] H. W. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, Los Angeles, 1968.

[KV00] Kupferman and Vardi. An automata-theoretic approach to reasoning about infinite-state systems. In E. A. Emerson and A. P. Sistla, editors, *Proceedings of the 12th International Conference on Computer Aided Verification (CAV'00)*, volume 1855 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.

[KVW00] Orna Kupferman, Moshe Y. Vardi, and Pierre Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, March 2000.

[LT00] C. Löding and W. Thomas. Alternating automata and logics over infinite words. In *Proceedings of the IFIP International Conference on Theoretical Computer Science,* *IFIP TCS2000*, volume 1872 of *Lecture Notes in Computer Science*, pages 521–535. Springer, August 2000.

[Maz77] Antoni Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977.

[Pel98] Doron Peled. Ten years of partial order reduction. In *CAV, Computer Aided Verification*, number 1427 in LNCS, pages 17–28, Vancouver, BC, Canada, 1998. Springer.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on the Foundations of Computer Science (FOCS-77)*, pages 46–57, Providence, Rhode Island, October 31–November 2 1977. IEEE, IEEE Computer Society Press.

[Roh97] S. Rohde. *Alternating automata and the temporal logic of ordinals.* PhD thesis, University of Illinois at Urbana-Champaign, 1997.

[Thi94] P. S. Thiagarajan. A trace based extension of linear time temporal logic. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 438–447, Paris, France, 4–7 July 1994. IEEE Computer Society Press.

[TW97] P. S. Thiagarajan and I. Walukiewicz. An expressively complete linear time temporal logic for Mazurkiewicz traces. In *Proceedings, Twelth Annual IEEE Symposium on Logic in Computer Science*, pages 183–194, Warsaw, Poland, 29 June–2 July 1997. IEEE Computer Society Press.

[Val91] Antti Valmari. A stubborn attack on state explosion. In Edmund M. Clarke and Robert P. Kurshan, editors, *Proceedings of Computer-Aided Verification (CAV '90)*, volume 531 of *LNCS*, pages 156–165, Berlin, Germany, June 1991. Springer.

[Var96] Moshe Y. Vardi. *An Automata-Theoretic Approach to Linear Temporal Logic*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer-Verlag Inc., New York, NY, USA, 1996.

[VW86] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Symposium on Logic in Computer Science (LICS '86)*, pages 332–345, Washington, D.C., USA, June 1986. IEEE Computer Society Press.

[Wal98] Igor Walukiewicz. Difficult configurations - on the complexity of LTrL. In *ICALP '98*, volume 1443 of *LNCS*, pages 140–151, 1998.

Aachener Informatik-Berichte

This is a list of recent technical reports. To obtain copies of technical reports please consult ftp://ftp.informatik.rwth-aachen.de/pub/reports/index.html or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: biblio@informatik.rwth-aachen.de

98-01 *	Jahresbericht 1997
98-02 *	S. Gruner/ M. Nagel / A. Schürr: Fine-grained and Structure-oriented Integration Tools are Needed for Product Development Processes
98-03 *	S. Gruner: Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr
98-04	O. Kubitz: Mobile Robots in Dynamic Environments
98-05 *	M. Leucker / St. Tobies: Truth — A Verification Platform for Distributed Systems
98-07 *	M. Arnold / M. Erdmann / M. Glinz / P. Haumer / R. Knoll / B. Paech / K. Pohl / J. Ryser / R. Studer / K. Weidenhaupt: Survey on the Scenario Use in Twelve Selected Industrial Projects
98-08	H. Aust: Sprachverstehen und Dialogmodellierung in natürlichsprachlichen Informationssystemen
98-09	Th. Lehmann: Geometrische Ausrichtung medizinischer Bilder am Beispiel intraoraler Radiographien
98-10	M. Nicola / M. Jarke: Performance Modeling of Distributed and Replicated Databases
98-11	A. Schleicher / B. Westfechtel / D. Jäger: Modeling Dynamic Software Processes in UML
98-12	W. Appelt / M. Jarke: Interoperable Tools for Cooperation Support using the World Wide Web
98-13 *	K. Indermark: Semantik rekursiver Funktionsdefinitionen mit Striktheitsinformation
99-01	Jahresbericht 1998
99-02	F. Huch: Verifcation of Erlang Programs using Abstract Interpretation and Model Checking — Extended Version
99-03	R. Gallersdörfer / M. Jarke / M. Nicola: The ADR Replication Manager
99-04 *	M. Alpuente / M. Hanus / S. Lucas / G. Vidal: Specialization of Functional Logic Programs Based on Needed Narrowing
99-07 *	Th. Wilke: CTL+ is exponentially more succinct than CTL
99-08 *	O. Matz: Dot-Depth and Monadic Quantifier Alternation over Pictures
2000-01	Jahresbericht 1999

2000-02 * **Jens Vöge / Marcin Jurdziński: A Discrete Strategy Improvement Algorithm for Solving Parity Games**

2000-04 * **Andreas Becks, Stefan Sklorz, Matthias Jarke: Exploring the Semantic Structure of Technical Document Collections: A Cooperative Systems Approach**

2000-05 **Mareike Schoop: Cooperative Document Management**

2000-06 **Mareike Schoop (Ed.): Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling**

2000-07 **Markus Mohnen / Pieter Koopman (Eds.): Proceedings of the 12th International Workshop of Functional Languages**

2000-08 * **Thomas Arts / Thomas Noll: Verifying Generic Erlang Client-Server Implementations**

* These reports are also available electronically as PostScript-files
via anonymous FTP on `ftp.informatik.rwth-aachen.de` (IP address 137.226.225.3)
or via WWW at URL `ftp://ftp.informatik.rwth-aachen.de/pub/reports/index.html`