Information Processing Letters 114 (2014) 408–413

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

# LTL is closed under topological closure

Grgur Petric Maretić *, Mohammad Torabi Dashti, David Basin

*Department of Computer Science, ETH, Universitätstrasse 6, Zürich, Switzerland*

ARTICLE INFO

*Article history:*
Received 2 May 2013
Received in revised form 28 February 2014
Accepted 1 March 2014
Available online 13 March 2014
Communicated by L. Viganò

*Keywords:*
Specification languages
LTL
Safety
Liveness
Büchi automata

ABSTRACT

We constructively prove that for every LTL formula $\varphi$, the smallest safety property containing the property expressed by $\varphi$ is also expressible in LTL. It immediately follows that LTL admits the safety-liveness decomposition: any property expressed by an LTL formula is equivalent to the intersection of a safety property and a liveness property, both of them expressible in LTL. Our proof is based on constructing a minimal deterministic counter-free Büchi automaton that recognizes the smallest safety property containing the property expressed by $\varphi$.



## 1. Introduction

Executions of reactive systems can be seen as infinite sequences of states, events, assertions, etc. A property, or language, is a set of executions. Lamport [1] introduced two important classes of properties: safety and liveness. Intuitively, a safety property states that something bad never happens, and a liveness property states that something good eventually happens. Many system validation and verification methods are tailored to these classes. For example, safety properties can be finitely falsified and are therefore testable, while liveness properties are not; safety and liveness require different proof techniques too [1,2].

Alpern and Schneider [3] use a topological argument to show that every property is the intersection of a safety property and a liveness property. It is natural to ask whether, for a property expressed in Linear Temporal Logic (LTL) [4], the decomposition can be done within LTL. In this article, we sharpen Alpern and Schneider's result by constructively proving that any property expressed by an LTL formula is equivalent to the intersection of a safety property and a liveness property, both of them expressible by LTL formulas.

### 1.1. Related work

Alpern and Schneider [2] use Büchi automata to prove that any $\omega$-regular property is the intersection of a safety property and a liveness property, themselves being $\omega$-regular. Since any property expressible in LTL is $\omega$-regular [5], it follows that any property expressed by an LTL formula is the intersection of a safety property $S$ and a liveness property $L$, both being $\omega$-regular. However, it does not follow from this that the properties $S$ and $L$ are both expressible by LTL formulas, as LTL formulas are only capable of expressing star-free $\omega$-regular languages, which constitute a strict subset of $\omega$-regular languages [5].

Lichtenstein, Pnueli, and Zuck [6] prove a normal form theorem (LPZ normal form) for LTL that intuitively states that each LTL formula is equivalent to a positive Boolean combination of what they define as safety and liveness formulas. Their definition of safety coincides with Alpern and Schneider's, but their liveness formulas (LPZ liveness) can express properties that are not liveness by Alpern and Schneider's definition. Hence, their results are essentially incomparable to ours. Moreover, the LPZ normal form is a

* Corresponding author.
*E-mail addresses:* pgrgur@inf.ethz.ch (G. Petric Maretić), torabidm@inf.ethz.ch (M. Torabi Dashti), basin@inf.ethz.ch (D. Basin).

http://dx.doi.org/10.1016/j.ipl.2014.03.001
0020-0190/

disjunction of conjunctions, in contrast to the single conjunction of our LTL decomposition. Furthermore, conjunctions in LPZ normal form do not always consist of one safety formula and one LPZ liveness formula: they could both be safety or both be LPZ liveness.

Although the notion of topological closure, safety, and liveness have been studied in temporal logic, see e.g. [7,8], we are not aware of any safety-liveness decomposition results for LTL.

### 1.2. Organization

In Section 2 we review safety and liveness, the safety-liveness decomposition theorem of Alpern and Schneider, LTL, and Büchi automata. We then define reduced and counter-free Büchi automata, and introduce safety automata. In Section 3 we present our main contributions: LTL is closed under topological closure and the safety-liveness decomposition theorem holds in LTL. For any Büchi automaton, we construct a minimal safety automaton that recognizes its safety part. Whenever the original automaton expresses an LTL formula, the resulting safety automaton is counter-free. Afterward, we describe the construction of an LTL formula from the safety automaton.

## 2. Preliminaries

Most definitions in this section are standard.

### 2.1. Safety and liveness

Fix an alphabet $\Sigma$. Let $\Sigma^\omega$ be the set of all countably infinite sequences over $\Sigma$, $\Sigma^+$ be the set of all finite nonempty sequences over $\Sigma$, and $\Sigma^* = \Sigma^+ \cup \{\epsilon\}$, where $\epsilon$ is the empty sequence. An element of $\Sigma^\omega$ is a **path**, and a **property** is any set of paths. A **trace** is an element of $\Sigma^+$. We write $|t|$ for the length of trace $t$. For a path $\pi = p_0 p_1 \cdots$, its **prefix** $\pi_i$ is the trace $p_0 p_1 \cdots p_i$. The **concatenation** of trace $t$ and (trace or) path $\pi$ is denoted $t\pi$. The empty trace $\epsilon$ is the neutral element for concatenation. The concatenation of a set of traces to a property is defined in the standard way. A **family** is a set of properties. Examples of families include the set of properties expressible by LTL formulas (defined below), the set of $\omega$-regular languages, denoted $\Omega$ (defined below), and the set of all properties, denoted $\Pi$. A family $\mathcal{F}$ is **closed under the $n$-ary function** $f : \Pi^n \to \Pi$, with $n \geqslant 0$, if $\forall P_1, \ldots, P_n \in \mathcal{F}.\ f(P_1, \ldots, P_n) \in \mathcal{F}$.

Lamport [1] defines a safety property as one that states that something (usually bad) never happens. For example, the elevator door never opens while the elevator is moving. In contrast, a liveness property states that something (usually good) eventually happens. For example, every process in a multitasking system will eventually be granted CPU time. There are various formalizations of safety and liveness, such as [6], but Alpern and Schneider's formalization [3] has become widely accepted.

**Definition 1.** (See [3].) A set $S$ of paths is a **safety property** if for every $\pi \notin S$ there is a natural number $i \in \mathbb{N}_0$ such that $\pi_i \sigma \notin S$ for all paths $\sigma \in \Sigma^\omega$. Intuitively, every path that is not in $S$ has a finite irremediable ("bad") prefix.

A set $L$ of paths is a **liveness property** if for every $t \in \Sigma^*$ there is a path $\pi$ such that $t\pi \in L$. Intuitively, every trace can be remedied (when the "good thing" occurs). ▲

Safety and liveness can also be defined through the notion of "distance". Given two executions $\pi^1$ and $\pi^2$, one can measure their similarity in terms of the smallest index at which they differ. This can be formalized by the **metric** $d$, defined by $d(\pi, \pi) = 0$ and $d(\pi^1, \pi^2) = 2^{-i}$, where $i = \min\{j \in \mathbb{N}_0 \mid \pi^1_j \neq \pi^2_j\}$. The distance between a path $\pi$ and a property $P$ is then defined as $d(\pi, P) = \inf\{d(\pi, \pi') \mid \pi' \in P\}$. Intuitively, a set $S$ is a safety property if it includes all the paths that are arbitrarily close to it, and a set $L$ is a liveness property if every path is arbitrarily close to $L$. The proposition below summarizes this observation.

**Proposition 2.** *A set $S \subseteq \Sigma^\omega$ is a safety property if $\{\pi \in \Sigma^\omega \mid d(\pi, S) = 0\} \subseteq S$. A set $L \subseteq \Sigma^\omega$ is a liveness property if $\forall \pi \in \Sigma^\omega.\ d(\pi, L) = 0$.* □

These characterizations of safety and liveness correspond to the characterizations of closed and dense sets in metric spaces, respectively. Therefore, in the topology induced by $d$, safety properties correspond to closed sets, and liveness properties correspond to dense sets. It then follows that safety properties are closed under (infinite) set intersection, and liveness properties are closed under set union. Theorem 4 below immediately follows from this topological characterization of safety and liveness.

For a family $\mathcal{F}$, we write $\mathcal{F}^s = \{P \in \mathcal{F} \mid P \text{ is a safety property}\}$ and $\mathcal{F}^\ell = \{P \in \mathcal{F} \mid P \text{ is a liveness property}\}$. Consequently, $\Pi^s$ and $\Pi^\ell$ are families consisting of all safety properties and all liveness properties, respectively.

**Definition 3.** A family $\mathcal{F}$ admits the **safety-liveness decomposition** if

$$\forall P \in \mathcal{F}.\quad \exists S \in \mathcal{F}^s.\quad \exists L \in \mathcal{F}^\ell.\quad P = S \cap L.$$ ▲

**Theorem 4** *($\Pi$ admits the safety-liveness decomposition). (See [3].) For every property $P \in \Pi$, there exist a property $S \in \Pi^s$ and a property $L \in \Pi^\ell$ such that $P = S \cap L$.* □

We define the unary **topological closure** function $\lceil \cdot \rceil : \Pi \to \Pi$ as $\lceil P \rceil = \bigcap\{S \in \Pi^s \mid P \subseteq S\}$. That is, $\lceil P \rceil$ is the smallest safety property that contains the property $P$. We define the **complement** function $\overline{\cdot} : \Pi \to \Pi$ as $\overline{P} = \Sigma^\omega \setminus P$. If a family $\mathcal{F}$ is closed under $\cap$ and under $\overline{\cdot}$, then $\mathcal{F}$ is closed under set union $\cup$. Examples of families that are closed under $\cap$ and $\overline{\cdot}$ are $\Pi$, $\Omega$, and the set of properties expressible in LTL. The following proposition restates Corollary 1.1 of [3], using our notation.

**Proposition 5.** *Let $\mathcal{F}$ be a family closed under $\cap$, $\overline{\cdot}$, and $\lceil \cdot \rceil$. Then $\mathcal{F}$ admits the safety-liveness decomposition. Namely, for any $P \in \mathcal{F}$,*

$$P = \lceil P \rceil \cap \left(\overline{\lceil P \rceil} \cup P\right),$$

*where* $\lceil P \rceil \in \mathcal{F}^s$ *and* $\overline{\lceil P \rceil} \cup P \in \mathcal{F}^\ell$. □

### 2.2. LTL and Büchi automata

Let $AP$ be a finite set of atomic propositions. The syntax of LTL formulas [4] is given by the grammar $\varphi ::= \top \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi \mathcal{U} \varphi$, where $a \in AP$. We write $\varphi_1 \vee \varphi_2$ for $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\Diamond\varphi$ for $\top \mathcal{U} \varphi$, and $\Box\varphi$ for $\neg\Diamond\neg\varphi$. LTL formulas are interpreted over paths of $\Sigma^\omega$, where $\Sigma = 2^{AP}$. The satisfaction relation is defined inductively over the formula structure:

$$\begin{array}{lll}
\pi, i \models \top & & \\
\pi, i \models a & \text{if} & a \in p_i \\
\pi, i \models \neg\varphi & \text{if} & \pi, i \not\models \varphi \\
\pi, i \models \varphi \wedge \psi & \text{if} & \pi, i \models \varphi \text{ and } \pi, i \models \psi \\
\pi, i \models \bigcirc\varphi & \text{if} & \pi, i+1 \models \varphi \\
\pi, i \models \varphi \mathcal{U} \psi & \text{if} & \text{there is a } j \geqslant i \text{ such that } \pi, j \models \psi \\
 & & \text{and } \pi, k \models \varphi \text{ for all } i \leqslant k < j
\end{array}$$

An LTL formula $\varphi$ defines the property $L(\varphi) = \{\pi \in \Sigma^\omega \mid \pi, 0 \models \varphi\}$. An LTL formula $\varphi$ is a safety (liveness) formula if $L(\varphi)$ is a safety (respectively liveness) property. Two LTL formulas $\varphi_1$ and $\varphi_2$ are **equivalent**, denoted $\varphi_1 \equiv \varphi_2$, if $L(\varphi_1) = L(\varphi_2)$.

We assume that the reader is familiar with deterministic, non-deterministic, and counter-free finite state automata on finite words, and star-free regular expressions [9]. We take the definition of Büchi automata from [10].

**Definition 6.** A **Büchi automaton** over the alphabet $\Sigma$ is a tuple $\mathcal{A} = (Q, q_0, \Delta, F)$ with a finite set $Q$ of states, an initial state $q_0 \in Q$, a transition relation $\Delta \subseteq Q \times \Sigma \times Q$, and a set $F \subseteq Q$ of accepting states. A Büchi automaton is **deterministic** if $\Delta : Q \times \Sigma \to Q$ is a partial transition function.

For $t \in \Sigma^+$ we define $(s, t, s') \in \Delta^+$ if there is a sequence of states $s_0, s_1, \ldots, s_{|t|}$ such that $s = s_0$, $s' = s_{|t|}$ and $(s_i, t_i, s_{i+1}) \in \Delta$, for $0 \leqslant i < |t|$. A **run** of $\mathcal{A}$ on a path $\pi = p_0 p_1 \cdots \in \Sigma^\omega$ is a sequence of states $s_0 s_1 \cdots$ such that $s_0 = q_0$ and $(s_i, p_i, s_{i+1}) \in \Delta$, for $i \in \mathbb{N}_0$. The run is **accepting** if some element of $F$ occurs infinitely often in the run. Automaton $\mathcal{A}$ **accepts** $\pi$ if there is an accepting run of $\mathcal{A}$ on $\pi$. The set $L(\mathcal{A}) = \{\pi \in \Sigma^\omega \mid \mathcal{A} \text{ accepts } \pi\}$ is the property recognized by $\mathcal{A}$. ▲

The family of properties recognized by Büchi automata is the set of $\omega$-regular languages $\Omega$. We next define counter-free and reduced Büchi automata.

Note that Büchi automata and non-deterministic finite state automata on finite words (NFA) only differ in their acceptance condition. One may therefore view a Büchi automaton $\mathcal{A}$ as an NFA $\mathcal{A}_F$, and vice versa. In order to avoid confusion, we will use the explicit subscript $_F$ to denote finite state automata on finite words. Let $\mathcal{A} = (Q, q_0, \Delta, F)$ be a Büchi automaton defined using the alphabet $\Sigma$. For states $p, q \in Q$, let $L_{p,q} = \{t \in \Sigma^+ \mid (p, t, q) \in \Delta^+\}$ be the set of all nonempty traces that can reach state $q$ from state $p$. The automaton $\mathcal{A}$, or the NFA $\mathcal{A}_F$, is **aperiodic** if for all states $s \in Q$, all traces $u \in \Sigma^+$, and every $m \geqslant 1$, $u^m \in L_{s,s}$ entails $u \in L_{s,s}$. The Büchi automaton $\mathcal{A}$ is **counter-free** if $\mathcal{A}$ is aperiodic [11]. The NFA $\mathcal{A}_F$ is **counter-free** if its corresponding minimal deterministic automaton is aperiodic [9].

The following proposition is used in the proof of Theorem 13. Note that if $\mathcal{A}_F$ is an aperiodic minimal deterministic finite automaton on finite words (DFA), then both the automata $\mathcal{A}_F$ and $\mathcal{A}$ are counter-free.

**Proposition 7.** *Let $\mathcal{A}_F$ be an aperiodic NFA. Determinizing and minimizing $\mathcal{A}_F$ results in an aperiodic DFA.*

**Proof.** Assume the NFA $\mathcal{A}_F = (Q_A, q_{0A}, \Delta_A, F_A)$ is, using subset construction, determinized to $\mathcal{D}_F = (\mathcal{P}(Q_A), \{q_{0A}\}, \Delta_D, F_D)$. Take any $u \in \Sigma^+$, $P_0 \in \mathcal{P}(Q_A)$, and $m > 1$ such that $(P_0, u^m, P_0) \in \Delta_D^+$. We claim $(P_0, u, P_0) \in \Delta_D^+$. Take any sequence $P_0, P_1, \ldots, P_m$, where $(P_i, u, P_{i+1}) \in \Delta_D^+$, where $1 \leqslant i \leqslant m-1$, and $P_0 = P_m$. Note that the sequence in fact defines a cycle. In particular, $(P_i, u^m, P_i) \in \Delta_D^+$, for every $P_i$ in the sequence. Furthermore, we will show that $P_i \subseteq P_{i+1}$ for every $i \in \{0, \ldots, m-1\}$, which implies, since the sequence defines a cycle, that $P_0 = P_1$, thus proving the claim.

Take any $i \in \{0, \ldots, m-1\}$. For any state $s_1 \in P_i$, we construct the sequence $\zeta = s_1, s_2, \ldots, s_{|P_i|+1}$ by choosing $s_j$ such that $(s_j, u^m, s_{j-1}) \in \Delta_A^+$, for $1 < j \leqslant |P_i| + 1$. Since $(P_i, u^m, P_i) \in \Delta_D^+$, the states $s_1, s_2, \ldots, s_{|P_i|+1}$ are all elements of $P_i$. By the pigeonhole principle, there is an index $l$ such that $s_l$ is repeated in $\zeta$. That is, $(s_l, u^{km}, s_l) \in \Delta_A^+$, for some natural number $k \leqslant |P_i|$. This implies that $(s_l, u, s_l) \in \Delta_A^+$, as $\mathcal{A}_F$ is aperiodic. Since $(P_i, u, P_{i+1}) \in \Delta_D^+$, it follows that $s_l \in P_{i+1}$. Now, $s_1, \ldots, s_l$ of the sequence $\zeta$ all belong to $P_{i+1}$ because $(P_{i+1}, u^m, P_{i+1}) \in \Delta_D^+$. In particular $s_1 \in P_{i+1}$. That is, $P_i \subseteq P_{i+1}$. Since the sequence $P_0, P_1, \ldots, P_m$ defines a cycle, it follows from the transitivity of $\subseteq$ that $P_{i+1} \subseteq P_i$.

Now, suppose that minimizing $\mathcal{D}_F$ results in $\mathcal{M}_F = (Q_M, q_{0M}, \Delta_M, F_M)$. Note that $Q_M$ can be seen as a partitioning of the set of states of $\mathcal{D}_F$. Using an argument similar to the above one, it follows that $\mathcal{M}_F$ is aperiodic. □

Diekert and Gastin [11] show that counter-free Büchi automata correspond to LTL formulas. Their proof is however non-constructive: it does not describe how to construct a counter-free automaton from an LTL formula or vice versa.

**Theorem 8.** *(See [11].) Let $\mathcal{L} \subseteq \Sigma^\omega$ be a property. There is an LTL formula $\varphi$ such that $L(\varphi) = \mathcal{L}$ iff there is a counter-free Büchi automaton $\mathcal{A}$ such that $L(\mathcal{A}) = \mathcal{L}$.* □

In a Büchi automaton $\mathcal{A} = (Q, q_0, \Delta, F)$, the set of states **reachable** from state $s \in Q$ is defined as $\mathit{reach}(s) = \{s' \in Q \mid \exists t \in \Sigma^+.\ (s, t, s') \in \Delta^+\}$. Note that reachability is *not* reflexive in this context. We say $\mathcal{A}$ is **reduced** if an accepting state is reachable from any state $s \in Q$. One can reduce any Büchi automaton without changing

the language it recognizes: Iteratively remove from $Q$ all $s \in F$ where $reach(s) \cap F = \emptyset$ and restrict $\Delta$ correspondingly. Then, remove from $Q$ all $s \in Q \setminus F$ such that $reach(s) \cap F = \emptyset$ and then restrict $\Delta$ correspondingly. The time complexity of this algorithm is cubic in the number of states of the automaton. Note that $\mathcal{A}$ is reduced iff $\mathcal{A}_F$ is reduced.

For a Büchi automaton $\mathcal{A}$, an automaton recognizing $\lceil L(\mathcal{A}) \rceil$ can be constructed by reducing $\mathcal{A}$ and then making all of its states accepting.

**Theorem 9.** *(See [2].) Let $\mathcal{L}$ be a language recognized by a Büchi automaton $\mathcal{A}_\mathcal{L}$, and let $\mathcal{A}_\mathcal{R} = (Q, q_0, \Delta, F)$ be its reduced automaton. The topological closure of $\mathcal{L}$ is recognized by the automaton $(Q, q_0, \Delta, Q)$.* □

The following corollary is a direct consequence of Proposition 5, Theorem 9, and the fact that $\Omega$ is closed under $\cap$ and $\bar{\cdot}$.

**Corollary 10** *($\Omega$ admits the safety-liveness decomposition). (See [2].) For every $P \in \Omega$ there exist $S \in \Omega^s$ and $L \in \Omega^\ell$ such that $P = S \cap L$.* □

To summarize, Alpern and Schneider have proven that both $\Pi$ and $\Omega$ admit the safety-liveness decomposition. We are not aware of any other safety-liveness decomposition results in the literature (cf. Section 1.1).

A **safety automaton** is a reduced Büchi automaton that accepts a safety property. By Theorem 9, marking all the states of a safety automaton as accepting does not affect the language it recognizes. We therefore assume that safety automata have all states accepting. The following theorem intuitively states that we may determinize and minimize a safety automaton as is done for NFA. This obviously does not hold for arbitrary Büchi automata. We will build upon this result in the next section.

**Theorem 11.** *Let $\mathcal{A}$ and $\mathcal{B}$ be safety automata. Then $L(\mathcal{A}) = L(\mathcal{B})$ iff $L(\mathcal{A}_F) = L(\mathcal{B}_F)$.*

**Proof.** Assume $L(\mathcal{A}) = L(\mathcal{B})$, and let $t$ be a trace accepted by $\mathcal{A}_F$. Since $\mathcal{A}$ is reduced, an accepting state is reachable from each of its states. It immediately follows that an accepting run of $\mathcal{A}_F$ on $t$ can be extended to an accepting run of $\mathcal{A}$ on some path $t\pi$. Therefore, $\mathcal{B}$ accepts $t\pi$. Let $s_0 s_1 \cdots s_{|t|} \cdots$ be an accepting run of $\mathcal{B}$ on $t\pi$. The sequence of states $s_0 s_1 \cdots s_{|t|}$ is an accepting run of $\mathcal{B}_F$ on $t$, since all of its states are accepting. This argument symmetrically holds for the case where $t$ is accepted by $\mathcal{B}_F$.

Now assume that $L(\mathcal{A}_F) = L(\mathcal{B}_F)$, and let $\pi$ be a path accepted by $\mathcal{A}$. By way of contradiction, let us assume $\pi \notin L(\mathcal{B})$. Since $L(\mathcal{B})$ is a safety property, there is an index $i$ such that, for every path $\sigma$, $\pi_i \sigma \notin L(\mathcal{B})$. Since $\mathcal{B}$ is reduced, this can only hold if the trace $\pi_i$ is not be accepted by the automaton $\mathcal{B}_F$. However, $\pi_i \in \mathcal{A}_F$ because $\pi \in L(\mathcal{A})$ and all states of $\mathcal{A}_F$ are accepting. This contradicts the assumption $L(\mathcal{A}_F) = L(\mathcal{B}_F)$. This argument symmetrically holds for the case where $\pi$ is accepted by $\mathcal{B}$. □

## 3. Topological closure of LTL properties

We prove that LTL is closed under topological closure: for every LTL formula $\varphi$, there is an LTL formula, denoted by $\lceil \varphi \rceil$, such that $\lceil L(\varphi) \rceil = L(\lceil \varphi \rceil)$. Consequently, LTL admits the safety-liveness decomposition. Note that the symbol $\lceil \cdot \rceil$ is overloaded here.

The following lemma, which is used in the proof of Theorem 13, shows that reducing a counter-free automaton always results in a counter-free automaton. We remark that eliminating an arbitrary state from a counter-free automaton does not in general result in a counter-free automaton.

**Lemma 12.** *Let $\mathcal{A}$ be a counter-free Büchi automaton. The automaton $\mathcal{A}'$, obtained by reducing $\mathcal{A}$, is counter-free.*

**Proof.** If $q_0$ is not a state of $\mathcal{A}'$, then $L(\mathcal{A}) = \emptyset$. Therefore, the set of states of $\mathcal{A}'$ is empty and it is counter-free. Otherwise, by way of contradiction, let us assume that $\mathcal{A}' = (Q', q_0, \Delta', F')$ is not counter-free. Then, there is a trace $u \in \Sigma^+$, an $m > 1$, and a state $p \in Q'$ such that $u^m \in L'_{p,p}$ and $u \notin L'_{p,p}$ in $\mathcal{A}'$. Because $\mathcal{A}$ is counter-free, $u \in L_{p,p}$ in $\mathcal{A}$. Therefore, there is a sequence of states $p = s_0, s_1, \ldots, s_{|u|} = p$ such that $(s_i, u_i, s_{i+1}) \in \Delta$, for $0 \leqslant i < |u|$. Since $u \notin L'_{p,p}$, there must be some state $s_i$ in that sequence that is not a state of the automaton $\mathcal{A}'$. This means that in $\mathcal{A}$ there is no accepting state reachable from $s_i$. Since $p$ is reachable from $s_i$ there is also no accepting state reachable from $p$. If no accepting state is reachable from $p$ in $\mathcal{A}$, it follows that no accepting state is reachable from $p$ in $\mathcal{A}'$ as well. This contradicts the assumption that $\mathcal{A}'$ is reduced. □

The next theorem constitutes the core of our argument. It states that applying Alpern and Schneider's construction, given in Theorem 9, to *any* Büchi automaton that recognizes the language of an LTL formula and then minimizing the resulting safety automaton (which is justified by Theorem 11) gives us a counter-free safety automaton. Note that the requirement that the initial automaton is counter-free is relaxed here in comparison to Lemma 12. This has implications regarding the constructiveness of our proof: Although it is known (see Theorem 8) that for any LTL formula there *exists* a counter-free Büchi automaton that recognizes the language of the formula, we are not aware of any algorithm that constructs such an automaton for a given formula. However, constructing *some* Büchi automaton that recognizes the language of an LTL formula, dropping the counter-freeness condition, is well-understood; see, e.g., [12].

**Theorem 13.** *Let $\varphi$ be an LTL formula, and $\mathcal{A}$ be an arbitrary Büchi automaton that recognizes $L(\varphi)$. Suppose that $\mathcal{S}$ is the automaton obtained by reducing $\mathcal{A}$, and then labeling all its states accepting. Determinizing and then minimizing $\mathcal{S}$ results in a counter-free automaton $\mathcal{M}$.*

**Proof.** From Theorem 9, it follows that $\lceil L(\varphi) \rceil = L(\mathcal{S})$. Let $\mathcal{M}_F$ be the minimal deterministic automaton of the

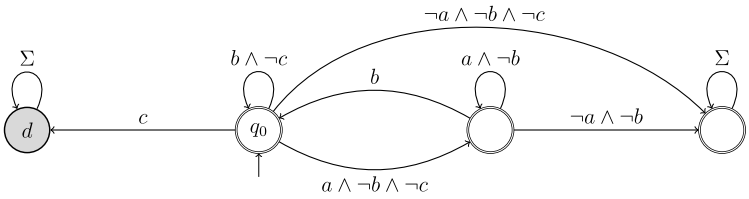


**Fig. 1.** Safety automaton $\mathcal{M}$.

NFA $\mathcal{S}_F$. It immediately follows that $\mathcal{M}_F$ is reduced and all of its states are accepting. From $L(\mathcal{M}_F) = L(\mathcal{S}_F)$, it follows that $\lceil L(\varphi) \rceil = L(\mathcal{M})$. Now, it remains to show that $\mathcal{M}$ is counter-free.

From Theorem 8 and Lemma 12, it follows that there is a reduced counter-free Büchi automaton $\mathcal{C}$, with all its states accepting, that recognizes $\lceil L(\varphi) \rceil$. Let us write $\mathcal{M}'_F$ for the DFA obtained by determinizing and minimizing $\mathcal{C}_F$. By Proposition 7, $\mathcal{M}'_F$ is aperiodic and therefore $\mathcal{M}'$ is counter-free.

Since $L(\mathcal{M}) = L(\mathcal{M}')$ and both are safety automata, by Theorem 11, it follows that $L(\mathcal{M}_F) = L(\mathcal{M}'_F)$. Automata $\mathcal{M}_F$ and $\mathcal{M}'_F$ must therefore be identical because each finite state automaton has, up to an isomorphism, a unique minimal automaton. Consequently, $\mathcal{M}$ and $\mathcal{M}'$ are identical too. Hence $\mathcal{M}$ is counter-free. □

The following corollary follows from Theorem 8 and Theorem 13.

**Corollary 14.** *LTL is closed under topological closure.* □

Since LTL has connectives for conjunction and negation, the family of languages expressible in LTL is closed under intersection and complement. The following corollary follows from Proposition 5 and Corollary 14. It states that for any LTL formula $\varphi$, there exist a safety LTL formula $\sigma$ and a liveness LTL formula $\lambda$ such that $\varphi \equiv \sigma \wedge \lambda$.

**Corollary 15.** *LTL admits the safety-liveness decomposition.* □

To complete the constructive proof of Corollary 14, we need to construct the formula $\lceil \varphi \rceil$ from the minimal safety automaton $\mathcal{M}$ recognizing $L(\lceil \varphi \rceil)$ of Theorem 13. We use a result by Pnueli and Zuck [13] for translating star-free regular expressions to LTL formulas evaluated on *finite* traces. For safety automata, it is simple to extend the result to infinite words.

We give only a brief account of the results we use; for the technical details and precise definitions see [13]. Let us first extend the semantics of LTL to finite traces. For a trace $t$ of length $|t|$, if $i < |t|$ then $t, i \models \varphi$ is defined the same as for infinite paths. However, for $i \geqslant |t|$ we define $t, i \not\models \varphi$, for any formula $\varphi$. Theorem 4.1 from [13] shows how one can, for a star-free regular expression $\alpha$, construct a *protected* LTL formula $\varphi$ such that $L(\alpha) = L(\varphi)$. The definition of protected formulas is beyond the scope of this paper. It is for us however instrumental that protected formulas are evaluated over finite traces, and they may contain an explicit end marker `last` that is only true at the end of a trace. From Theorem 4.4 of [13], it follows that replacing every occurrence of `last` in a protected formula $\varphi_1$ by a formula $\bigcirc \varphi_2$ on infinite, respectively finite, words results in an LTL formula that is satisfied by exactly the infinite, respectively finite, words $L(\varphi_1)L(\varphi_2)$.

We now explain how these results are used. Consider an LTL formula $\varphi$ on infinite paths and the corresponding safety automaton $\mathcal{M}$. We first extend $\mathcal{M}$'s partial transition function by adding a non-accepting dead state $d$ to the set of states. It is immediate that this does not change the language of the automaton. Any path that does not satisfy $\lceil \varphi \rceil$ has a finite prefix that reaches state $d$, and once $d$ is reached no other states of the automaton will be visited. Therefore, if $L$ is the set of all finite words reaching $d$, the set of all infinite paths that do not satisfy $\lceil \varphi \rceil$ is the set $L\Sigma^\omega$. Since $\mathcal{M}_F$ is a counter-free automaton (by Theorem 13), there is a star-free regular expression that captures the set of all finite traces reaching $d$. We use the construction of [13] to find a protected formula $\psi$ on finite words such that $L(\psi)$ is the set of all finite traces that reach state $d$. It is immediate that $L(\neg\lceil \varphi \rceil) = L(\psi)\Sigma^\omega = L(\psi)L(\top)$. Therefore $\lceil \varphi \rceil = \neg\psi^\top$, where $\psi^\top$ is obtained from $\psi$ by replacing every occurrence of `last` by $\top$.

**Example 16.** In this example, we illustrate how an LTL formula is constructed from a safety automaton. Consider the safety automaton $\mathcal{M}$ in Fig. 1, where we have already introduced the dead state $d$. All finite traces that reach the state $d$ can be expressed by the regular expression $(\epsilon + (a \vee b)^* b) c \Sigma^*$. This expression is star-free since $\Sigma^*$ can be replaced by the complement of the empty set, and $(a \vee b)^*$ can be replaced by the complement of $\Sigma^*(\neg a \wedge \neg b)\Sigma^*$. Here we use propositional formulas as syntactic sugar, where a formula $E$ stands for the union of all literals $l \in \Sigma = 2^{AP}$ that represent a truth assignment satisfying $E$. For example, the formula $E = c$ stands for the regular expression $(\{c\} + \{a, c\} + \{b, c\} + \{a, b, c\})$. This regular expression can be written, following the algorithm given in [13], as the protected formula $\psi = (c \wedge (\top \mathbf{U}\, \texttt{last})) \vee ((a \vee b)\, \mathbf{U}\, (b \wedge \bigcirc (c \wedge (\top \mathbf{U}\, \texttt{last}))))$, where $(\varphi_1 \mathbf{U} \varphi_2)$ stands for $\varphi_1 \mathcal{U} (\varphi_1 \wedge \bigcirc \varphi_2)$. The safety automaton recognizes the language of the formula $\neg\psi^\top$; that is, the automaton $\mathcal{M}$ recognizes the language of the LTL formula $\neg(c \vee ((a \vee b) \mathcal{U} (b \wedge \bigcirc c)))$. △

## References

[1] L. Lamport, Proving the correctness of multiprocess programs, IEEE Trans. Softw. Eng. 3 (2) (1977) 125–143.
[2] B. Alpern, F.B. Schneider, Recognizing safety and liveness, Distrib. Comput. 2 (3) (1987) 117–126.
[3] B. Alpern, F.B. Schneider, Defining liveness, Inf. Process. Lett. 21 (4) (1985) 181–185.

[4] A. Pnueli, The temporal logic of programs, in: FOCS, IEEE Computer Society, 1977, pp. 46–57.
[5] P. Wolper, Temporal logic can be more expressive, Inf. Control 56 (1/2) (1983) 72–99.
[6] O. Lichtenstein, A. Pnueli, L.D. Zuck, The glory of the past, in: R. Parikh (Ed.), Logic of Programs, in: Lecture Notes in Computer Science, vol. 193, Springer, 1985, pp. 196–218.
[7] B. Jonsson, Y.-K. Tsay, Assumption/guarantee specifications in linear-time temporal logic, Theor. Comput. Sci. 167 (1&2) (1996) 47–72.
[8] P. Maier, Intuitionistic LTL and a new characterization of safety and liveness, in: J. Marcinkowski, A. Tarlecki (Eds.), CSL, in: Lecture Notes in Computer Science, vol. 3210, Springer, 2004, pp. 295–309.
[9] R. McNaughton, S.A. Papert, Counter-Free Automata, M.I.T. Research Monograph, vol. 65, The MIT Press, 1971.
[10] W. Thomas, Automata on infinite objects, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, vol. B: Formal Models and Sematics (B), Elsevier and MIT Press, 1990, pp. 133–192.
[11] V. Diekert, P. Gastin, First-order definable languages, in: J. Flum, E. Grädel, T. Wilke (Eds.), Logic and Automata, in: Texts in Logic and Games, vol. 2, Amsterdam University Press, 2008, pp. 261–306.
[12] R. Gerth, D. Peled, M.Y. Vardi, P. 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, Chapman & Hall, Ltd., London, UK, 1996, pp. 3–18.
[13] A. Pnueli, L.D. Zuck, In and out of temporal logic, in: LICS, IEEE Computer Society, 1993, pp. 124–135.