Lost in Translation: from Linear Temporal Logic to Büchi Automata

Microsoft Technical Report MSR-TR-2022-26

MARGUS VEANES, Microsoft Research, USA
OLLI SAARIKIVI, Microsoft Research, USA
THOMAS BALL, Microsoft Research, USA

In the automata-theoretic approach to languages, formulas from a domain-specific language (such as regular expressions over finite words or a temporal logic over infinite words) are translated to automata, which come equipped with their own semantics, algebraic properties, and supporting algorithms. In the process of translating from formulas to automata, it is possible to lose track of the algebra that exists in the world of formulas, making it harder to reason about semantics and perform optimizations. Recent work on symbolic derivatives for extended regular expressions shows that it is possible to leverage effective Boolean algebras to represent both infinite spaces of characters as well as transition functions/terms, enabling optimizations that apply simultaneously at the level of formula and automata.

We develop here a *framework* of *transition terms* modulo an effective Boolean algebra $\mathcal{A}$ that works over ω*-languages* and over *infinite alphabets* in an algebraically well-defined and precise manner. Using this framework, we then define symbolic derivatives for linear temporal logic (LTL), and define symbolic alternating Büchi automata, based on a shared semantic representation that makes it simpler to reason about optimizations. We present several new optimizations, including one that allows locally eliminating alternation, which results in non-alternating or even deterministic Büchi automata for some classes of LTL. We believe there is a rich world of LTL rewriting rules for *on-the-fly* optimization of alternating Büchi automata to be discovered.

1 INTRODUCTION

When we define a higher-level language F in terms of a lower-level language A, we expect that the semantics of a program in f in language F is preserved when translated to a program $T_{F,A}(f)$ in language A. At the same time, we always recognize that something will be "lost in translation" in the process. In general, what might have been easy/simple to reason about for programs in language F becomes more difficult for the corresponding programs in language A. Examples abound:

- higher-level languages with structured control-flow constructs are compiled into linear bytecode with jump statements, losing the notion of syntactic nesting, which must be recovered by algorithms over the control-flow graph of the bytecode, as with dominance frontiers in static single assignment form [14];
- given a register r in a low-level representation A that corresponds to a program variable in F, if we don't know if r represents a number or a pointer from F then precise garbage collection becomes more difficult and requires conservative pointer finding [17];
- in languages that allow programmers to concisely describe numerical algorithms using high-level abstractions such as vectors and matrices, and the linear-algebraic operations over them, there is a need to both optimize algebraically at the level of mathematical abstractions as well as to compile the same representation into forms that express iteration over arrays, encode the layout of data, and address other concerns relevant to efficient execution [35].

Of course, something is gained as well as lost in the process of translation from F to A: semantics that were implicit at the level of language F can be made more explicit via a more detailed expansion in language A. In the example of numerical algorithms, lower levels of program representation can represent the layout of data, but are not well-suited to algebraic manipulation of mathematical expressions.

We have chosen the symbols F and A for a purpose: F represents a language of *Formulas*, while A represents a language of *Automata*. Automata can represent many different types of formalisms, ranging from regular expressions to temporal logic formula, such as linear temporal logic (LTL). Furthermore, automata generally form a Boolean algebra over their corresponding languages, which allows for their manipulation, both in theory and practice. Automata-based libraries have been part of the programming and reasoning toolkit for decades.

The rationale for leaving the realm of formulas and entering that of automata as early as possible is made persuasively by Tsay and Vardi in their recent paper describing the automata-theoretic approach to working with LTL through translation to Büchi automata [43]. They propose adhering to an "early and simple" principle: when given multiple paths for reduction (translation) from the application domain (essentially, the world of formulas), leave that world as early as possible to take full advantage of the cornucopia of automata-based algorithms. They further argue that by doing so, the process is made simpler, especially by choosing automata (such as alternating Büchi automata) that admit a very straightforward translation from LTL.

On the other hand, recent work in the domain of extended regular expressions (ERE) shows that it is possible to have a "simultaneous semantics" that unites the worlds of formula and automata, enabling more precise and stronger optimizations [41]. In the case of ERE, maintaining the connection between automata states and the regular expressions through symbolic derivatives enables ERE-level optimizations that are otherwise lost. At the same time, symbolic derivatives remain closed under *all Boolean operations*, maintaining a finite state space, as well as *incrementality* of the derivation process itself. Most striking perhaps is that complementation of automata is avoided by working directly with complemented regexes incrementally through their symbolic derivatives.

While derivatives are mainly known in the application domain of regular expressions over finite words, we show that they have something important to say about languages over infinite words that form the foundation for the semantics of LTL and Büchi automata. In his 1995 paper "An Automata-Theoretic Approach to Linear Temporal Logic" [46], Vardi presents Theorem 22, which relates the semantics of an LTL formula ϕ to that of an alternating Büchi automata A_ϕ constructed from ϕ, which conforms to the following grammar:

$$\varphi \quad \rightarrow \quad p, \quad \neg\varphi, \quad \varphi \wedge \psi, \quad \varphi \vee \psi, \quad \mathbf{X}\psi, \quad \varphi \,\mathbf{U}\, \psi$$

where p is a proposition from P, a finite set of atomic propositions, and $\mathbf{X}$ and $\mathbf{U}$ are the temporal operators referred to as "next" and "until". Most surprising to us is Vardi's formulation of the transition relation ρ of A_ϕ, by induction over the formula ϕ with respect to a given element $a \in \mathbb{D} = 2^P$:

$$\begin{array}{lcl}
\rho(p, a) & = & p \in a \\
\rho(\varphi \wedge \psi, a) & = & \rho(\varphi, a) \wedge \rho(\psi, a) \\
\rho(\neg\varphi, a) & = & \overline{\rho(\varphi, a)} \\
\rho(\mathbf{X}\psi, a) & = & \psi \\
\rho(\varphi \,\mathbf{U}\, \psi, a) & = & \rho(\varphi, a) \vee (\rho(\psi, a) \wedge \varphi \,\mathbf{U}\, \psi)
\end{array}$$

where the dual $\overline{\phi}$ of a formula ϕ is obtained as usual by negating ϕ and pushing negation down to the leaves of ϕ, via de Morgan (leaving temporal operators and their subformula untouched).

All LTL formulas ψ have a *language* semantics $\mathbf{L}(\psi) \subseteq \mathbb{D}^\omega$. The *derivative* of $L \subseteq \mathbb{D}^\omega$ with respect to $a \in \mathbb{D}$ is the set $\mathbf{D}_a(L) \stackrel{\text{DEF}}{=} \{v \mid av \in L\}$. Now, the following property holds for the definition of ρ for all $a \in \mathbb{D}$ and all LTL formulas ϕ:

$$\mathbf{D}_a(\mathbf{L}(\phi)) = \mathbf{L}(\rho(\phi, a)).$$

Therefore, by construction in [46], it follows that for *all states* ψ of A_ϕ: $\mathcal{L}(\psi) = \mathbf{L}(\psi)$ where $\mathcal{L}(\psi)$ is the the language accepted by the alternating Büchi automaton A_ϕ *with respect to the state* ψ – we call this *$\mathcal{L}$L-invariance*.

We therefore name this inductive construction ***Vardi derivatives for LTL*** ("Vardi derivatives" for short) as it has all the desired properties of derivatives, as shown above. To the best of our knowledge, this analogy with regular expression derivatives has so far not been made.

Observe that in the regular expression world the languages of ERE and finite automata coincide, enabling direct optimizations such as approximate *subsumption* at the level of regular expressions [47]. In the world of LTL this is not always the case with $\mathbf{L}(\mathbf{GE}p) \subset \mathbf{L}(\mathbf{E}p)$ being a concrete example because in $A_{\mathbf{GE}p}$ the state $\mathbf{GE}p$ is accepting while $\mathbf{E}p$ is not. Replacing $(\mathbf{E}p) \wedge \mathbf{GE}p$, where $\wedge$ is *conjuction of states* in the autotmaon, with $\mathbf{GE}p$ would be incorrect in the context of alternating Büchi automata while correct in the context of LTL where $\wedge$ is *conjuction of formulas*, despite the fact that formulas $\mathbf{E}p$ and $\mathbf{GE}p$ are states themselves.

We now turn to the second key observation, namely that the *concrete* derivation step, $\rho(p, a) = p \in a$ above can be made into a *symbolic* derivation by lifting the concept of *transition regexes* from [41] to LTL, namely $\rho(p) = \mathbf{ite}(p, \top, \bot)$ where the decision of actually computing the derivative $\rho(p)(a)$ is being deferred. This enables the semantics to be defined without prior knowledge of $\mathbb{D}$. Not only that, $\mathbb{D}$ can now be an arbitrary Boolean algebra $\mathcal{A}$ (even infinite, such as ERE), where p above is a formula in that algebra, and the semantics of $\rho(p)(a)$ above becomes $a \models_{\mathcal{A}} p$, with a being an element in the domain of $\mathcal{A}$. In order to achieve this, we develop here a *framework* of *transition terms* modulo $\mathcal{A}$, $\mathbb{T}_{\mathcal{A}}$, that works over *ω-languages* and over *infinite alphabets* in an algebraically well-defined and precise manner. Our definition of symbolic derivatives for LTL modulo $\mathcal{A}$ is a *conservative extension* of Vardi derivatives that preserves both the structure of the formulas and their semantics precisely.

We present a number of LTL-based optimizations that respect $\mathcal{L}$L-invariance and thus apply in the "simultaneous semantics" of LTL + alternating Büchi automata. While some of our optimizations depend only on the functional properties of transition terms or on the laws of their Boolean algebra, for others establishing $\mathcal{L}$L-invariance requires a deeper look at the semantics of alternating Büchi automata. In particular, we use the concept of *suspendable* formulas [5] to develop Theorem 8.8, which allows reducing alternation by treating formulas such as $(\mathbf{E}p) \wedge \mathbf{GE}p$ as a single state in the automaton being constructed. While requiring a careful proof, the rule itself can be applied purely syntactically by virtue of its $\mathcal{L}$L-invariance maintaining the link between formulas and states.

We believe there is a rich world of $\mathcal{L}$L-invariant optimizations yet to be discovered. Many pure-LTL optimizations are already known – Somenzi and Bloem [39] alone list 20 rules[1] for preprocessing LTL formulas – and the $\mathcal{L}$L-invariant rules we present coincide with some of these. We expect future analyses to uncover many more that either are $\mathcal{L}$L-invariant or can be constrained to be so. The promise of our work is a future where, like with ERE, simple syntactic rules respecting the "simultaneous semantics" of LTL and alternating Büchi automata can be used to apply powerful optimizations *on-the-fly* during automata construction.

Overview. Section 2 presents basic material about languages over infinite words, effective boolean algebras ($\mathcal{A}$), and Boolean closures. Section 3 defines the syntax and semantics of linear temporal logic modulo $\mathcal{A}$ ($LTL_{\mathcal{A}}$) and properties that are critical for the application of symbolic derivatives. Section 4 introduces transition terms modulo $\mathcal{A}$, which lays the foundation for treating both LTL and Büchi automata symbolically, and Section 5 shows a number of optimizations that are enabled by the fact that $\mathcal{A}$ and $\mathbb{T}_{\mathcal{A}}$ are both boolean algebras (these optimizations are independent of the semantics of $LTL_{\mathcal{A}}$ and Büchi automata). Section 6 redefines the semantics of $LTL_{\mathcal{A}}$ via symbolic derivatives, which

[1]The earlier counterexample of $\mathbf{L}(\mathbf{GE}p) \subset \mathbf{L}(\mathbf{E}p)$ not justifying a rewrite of $(\mathbf{E}p) \wedge \mathbf{GE}p$ to $\mathbf{GE}p$ is an instance of the first rule in [39, Section 3].

are based on transition terms, while Section 7 does the same for alternating Büchi automata ($ABA_{\mathcal{A}}$). With these two abstractions defined by the same transition terms, Section 8 gives and proves correct two translations of $LTL_{\mathcal{A}}$ to $ABA_{\mathcal{A}}$, and discusses several classes of optimizations based on the semantics/translation. Section 9 reviews related work and Section 10 concludes the paper.

Proofs. Proofs that are omitted from the main body of the paper can be found in the Appendix.

2 PRELIMINARIES

2.1 Infinite sequences

We work with infinite sequences over a nonempty domain $\mathbb{D}$, denoted by $\mathbb{D}^\omega$. A sequence $v \in \mathbb{D}^\omega$ is formally a function from $\mathbb{N}$ to $\mathbb{D}$. When it is unambiguous we write v_i for $v(i)$ for $i \in \mathbb{N}$. The complement of a subset $L \subseteq \mathbb{D}^\omega$ is defined as $\complement(L) \stackrel{\text{DEF}}{=} \mathbb{D}^\omega \setminus L$. We also define the complement of a subset $S \subseteq \mathbb{D}$ as $\complement(S) \stackrel{\text{DEF}}{=} \mathbb{D} \setminus S$.

If $a \in \mathbb{D}$ and $v \in \mathbb{D}^\omega$ then $a{\cdot}v$ is a shorthand for defining the following sequence for $i \in \mathbb{N}$:

$$(a{\cdot}v)(i) \begin{cases} a, & \text{if } i = 0; \\ v(i-1), & \text{otherwise.} \end{cases}$$

More generally, if $S \subseteq \mathbb{D}$ and $L \subseteq \mathbb{D}^\omega$ then $S{\cdot}L$ denotes the subset $\{a{\cdot}v \mid a \in S, v \in L\}$ of $\mathbb{D}^\omega$. Observe that if S or L is empty then $S{\cdot}L$ is also empty. We write av for $a{\cdot}v$ or SL for $S{\cdot}L$ when this is unambiguous. We use the following additional definition for infinite sequences. If $v \in \mathbb{D}^\omega$ and $n \in \mathbb{N}$ then $v_{n..} \stackrel{\text{DEF}}{=} \lambda i.v(n+i)$ is the *n'th rest* of v. Note that concatenation ($\cdot$) binds stronger than intersection ($\cap$) that binds stronger than union ($\cup$). Also observe also that $\mathbb{D} \cdot \mathbb{D}^\omega = \mathbb{D}^\omega$. If $a \in \mathbb{D}$ then a^ω denotes the infinite sequence such that $a^\omega(i) = a$ for all $i \in \mathbb{N}$.

2.2 Boolean Algebras

Given a nonempty *universe* $\mathbb{D}$, a *Boolean algebra over* $\mathbb{D}$ is a tuple $\mathcal{A} = (\mathbb{D}, \Psi, [\![\_]\!], \bot, \top, \vee, \wedge, \neg)$ where Ψ is a set of *predicates* closed under the Boolean connectives; $[\![\_]\!] : \Psi \to 2^{\mathbb{D}}$ is a *denotation function*; $\bot, \top \in \Psi$; $[\![\bot]\!] = \emptyset$, $[\![\top]\!] = \mathbb{D}$, and for all $\alpha, \beta \in \Psi$, $[\![\alpha \vee \beta]\!] = [\![\alpha]\!] \cup [\![\beta]\!]$, $[\![\alpha \wedge \beta]\!] = [\![\alpha]\!] \cap [\![\beta]\!]$, and $[\![\neg\alpha]\!] = \mathbb{D} \setminus [\![\alpha]\!]$.[2] For $\alpha, \beta \in \Psi$ we write $\alpha \equiv \beta$ to mean $[\![\alpha]\!] = [\![\beta]\!]$. In particular, if $\alpha \equiv \bot$ then α is *unsatisfiable* and if $\alpha \equiv \top$ then α is *valid*. $\mathcal{A}$ is *effective* if all components of $\mathcal{A}$ are recursively enumerable, and satisfiability is decidable in $\mathcal{A}$. We use $\mathcal{A}$ as a subscript to indicate a component of $\mathcal{A}$, e.g., $\Psi_{\mathcal{A}}$ is the set of predicates of $\mathcal{A}$. We often omit the subscript when it follows from the context.

A *minterm* of a finite subset Γ of Ψ is a predicate $(\bigwedge S) \wedge \neg\bigvee(\Gamma \setminus S)$ for some $S \subseteq \Gamma$. *Minterms*(Γ), say Σ, denotes the set of all minterms of Γ. The *core properties* of Σ are that all minterms are satisfiable and mutually disjoint, and that each satisfiable predicate in Γ is equivalent to a disjunction of some minterms. Thus, Σ defines a *finite partition* of $\mathbb{D}$. For example, if $\Gamma = \{\alpha, \beta\}$ then Σ is the set of all satisfiable predicates in $\{\alpha \wedge \beta, \neg\alpha \wedge \beta, \alpha \wedge \neg\beta, \neg\alpha \wedge \neg\beta\}$. If all of them are satisfiable then each one identifies one of the four regions of the *Venn diagram* formed by $[\![\alpha]\!]$ and $[\![\beta]\!]$.

We let $O^{\text{sat}}_{\mathcal{A}}(n)$ denote the computational complexity of checking satisfiability in $\mathcal{A}$ for predicates ψ of size $|\psi| = n$. Here we make the standard assumption that the size of a predicate is the sum of the sizes of its subformulas. Under this assumption it follows that the computation cost of Σ is $O(2^{O^{\text{sat}}_{\mathcal{A}}(n)})$ where $n = \sum_{i<k} |\gamma_i|$ and $k = |\Gamma|$. Observe also that $|\Sigma| \leq 2^k$, i.e., the *number* of minterms is in the worst case exponential in k. This assumption is not always accurate. For example for BDDs the Boolean operations themselves are quadratic while deciding satisfiablity is trivial, but in this

[2]Observe that if $\mathbb{D} = \emptyset$ then $2^{\mathbb{D}} = \{\emptyset\}$ in which case $\top$ and $\bot$ are indistinguishable because then $[\![\top]\!] = [\![\bot]\!] = \emptyset$.

setting it is appropriate to assume that the actual Boolean operations are being postponed (at least in theory) until satisfiability is being checked at which point the operations are actually performed.

Extended Regular Expressions. Let $\mathcal{A} = ERE$ be an effective Boolean algebra (or *ERE*-solver for short) for all the extended regular expressions (*regexes* for short) modulo the set U of all Unicode characters, as defined and used in [41]. In this case $\mathbb{D} = U^*$ is the set of all strings over U, Ψ is the set of all regexes, and for any regex ψ, $s \in [[\psi]]$ means that ψ matches s. In this case, using standard notation, the regex `([A-Z]+)` matches all nonempty strings of capital letters, e.g., `"HELLO"`. The regex `(\d+)` matches all nonempty strings of digits, e.g., `"0123"`. We will use *ERE* in several examples.

2.3 Boolean Closure

Given a nonempty (possibly infinite) set Q of basic elements called *states*, we define the Boolean closure $\mathbb{B}(Q)$ of Q to contain the following expressions. If $q \in Q$ then $q \in \mathbb{B}(Q)$ and if $p, q \in \mathbb{B}(Q)$ then $p \vee q, p \wedge q, \neg q \in \mathbb{B}(Q)$. The Boolean connectives are treated here as *commutative*, *associative*, and *idempotent* operators.

Now consider any nonempty domain D and any given *denotation* function $\mathbf{L} : Q \to 2^D$ associated with states. If there is an element $q \in Q$ such that $\mathbf{L}(q) = D$ then select that element as $q_\top$ else let $q_\top \stackrel{\text{DEF}}{=} q \vee \neg q$ for some fixed $q \in Q$. Analogously, if there is an element $q \in Q$ such that $\mathbf{L}(q) = \emptyset$ then select that element as $q_\perp$ else let $q_\perp \stackrel{\text{DEF}}{=} \neg q_\top$. Extend the definiton of $\mathbf{L}$ to all elements of $\mathbb{B}(Q)$ as usual, giving rise to the following Boolean algebra over D:

$$(D, \mathbb{B}(Q), \mathbf{L}, q_\perp, q_\top, \vee, \wedge, \neg)$$

We refer to such a Boolean algebra as being *induced* by Q and $\mathbf{L}$. Observe that de Morgan's laws and laws of ditributivity hold in this Boolean algebra, independently of $\mathbf{L}$ defined for the basic elements of Q. It is also allowed for Boolean combinations of basic states already occur in Q if they obey the laws of the algebra.

We write $\mathbb{B}^+(Q)$ for the *positive* Boolean closure of Q where the complement $\neg$ is not allowed.

We define the *negation normal form* for elements of $\mathbb{B}(Q)$ as usual where $NNF(q) \stackrel{\text{DEF}}{=} q$ for $q \in Q$:

$$NNF(p \vee q) \stackrel{\text{DEF}}{=} NNF(p) \vee NNF(q) \qquad NNF(p \wedge q) \stackrel{\text{DEF}}{=} NNF(p) \wedge NNF(q) \qquad NNF(\neg\neg q) \stackrel{\text{DEF}}{=} NNF(q)$$
$$NNF(\neg(p \wedge q)) \stackrel{\text{DEF}}{=} NNF(\neg p) \vee NNF(\neg q) \qquad NNF(\neg(p \vee q)) \stackrel{\text{DEF}}{=} NNF(\neg p) \wedge NNF(\neg q)$$

We also use the *disjunctive normal form* $DNF(q)$ of $q \in \mathbb{B}(Q)$ defined as a disjunction of conjunctions of the $NNF(q)$ where all complements are applied to states only. For $q \in \mathbb{B}^+(Q)$ we apply $DNF(q)$ directly because there is no complement. These normal forms follow from de Morgan's laws and laws of distributivity of Boolean operations.

3 LTL MODULO $\mathcal{A}$

The following are the $LTL_\mathcal{A}$ formulas where $\mathcal{A} = (\mathbb{D}, \Psi, [[\_]], \perp, \top, |, \&, \sim)$ is a given (effective) Boolean algebra. We will use $\mathcal{A}$ throughout the rest of the paper as the underlying *element* algebra. We write *LTL* for $LTL_\mathcal{A}$ when $\mathcal{A}$ is clear from the context.

- if $\alpha \in \Psi$ then α is a formula in *LTL*,
- if φ, ψ are *LTL* formulas then $\neg\varphi$, $\varphi \vee \psi$, $\varphi \wedge \psi$, $\mathbf{X}\psi$, $\psi \,\mathbf{R}\, \phi$ are *LTL* formulas.

We let the *true* formula be $\top$ and the *false* formula be $\perp$ from $\mathcal{A}$. We also use the following abbreviations:

- *Logical implication*: $\varphi \to \psi \stackrel{\text{DEF}}{=} \neg\varphi \vee \psi$
- *Until*: $\varphi \,\mathbf{U}\, \psi \stackrel{\text{DEF}}{=} \neg(\neg\varphi \,\mathbf{R}\, \neg\psi)$
- *Eventually*: $\mathbf{E}\psi \stackrel{\text{DEF}}{=} \top \,\mathbf{U}\, \psi$

- *Globally*: $\mathbf{G}\psi \stackrel{\text{DEF}}{=} \bot\ \mathbf{R}\ \psi$

Another common notation for **E** (Eventually) is **F** (Finally). We have the non-standard approach where **U** (Until) is defined as the dual of **R** (Release), rather than the other way around. The main reason for doing so is to treat the formulas $\top$ and $\varphi\ \mathbf{R}\ \psi$ uniformly as being the *positive* formulas treated as accepting states in Section 8.

3.1 Semantics

An infinite sequence $w \in \mathbb{D}^\omega$ is a *model of* $\varphi \in LTL$, denoted by $w \models \varphi$, when the following holds, where $\alpha \in \Psi$:

$$w \models \alpha \quad \stackrel{\text{DEF}}{=} \quad w(0) \in [[\alpha]] \tag{1}$$

$$w \models \varphi \wedge \psi \quad \stackrel{\text{DEF}}{=} \quad w \models \varphi \text{ and } w \models \psi \tag{2}$$

$$w \models \varphi \vee \psi \quad \stackrel{\text{DEF}}{=} \quad w \models \varphi \text{ or } w \models \psi \tag{3}$$

$$w \models \neg\varphi \quad \stackrel{\text{DEF}}{=} \quad w \not\models \varphi \tag{4}$$

$$w \models \mathbf{X}\psi \quad \stackrel{\text{DEF}}{=} \quad w_{1..} \models \psi \tag{5}$$

$$w \models \varphi\ \mathbf{R}\ \psi \quad \stackrel{\text{DEF}}{=} \quad \forall j \in \mathbb{N} : w_{j..} \models \psi \quad \text{or} \quad \exists j \in \mathbb{N} : w_{j..} \models \varphi \text{ and } \forall i \le j : w_{i..} \models \psi \tag{6}$$

$$w \models \varphi\ \mathbf{U}\ \psi \quad \Leftrightarrow \quad \exists j \in \mathbb{N} : w_{j..} \models \psi \text{ and } \forall i < j : w_{i..} \models \varphi \tag{7}$$

The rules (6) and (7) are duals of each other, either one suffices as the main definition, although we treat an **R** formula as being *positive* while an **U** formula (as its dual) is treated as a *negative* formula. In (6) either ψ holds forever in w, or at some step j, φ holds in w and ψ holds until (including) step j. Intuitively either φ "releases" ψ at some step or else ψ has to hold forever.

It follows from the definition above and laws of $\mathcal{A}$ that if $\alpha, \beta \in \Psi$ then $w \models \alpha \wedge \beta$ iff $w \models \alpha\ \&\ \beta$, and $w \models \neg\alpha$ iff $w \models {\sim}\alpha$. In other words, any subformula of an LTL formula that is a Boolean combination of predicates from $\mathcal{A}$ can itself be reduced to a predicate in $\mathcal{A}$. This is a useful simplifying reduction when working with LTL formulas.

In some situations we prefer **U** over **R** because **U** is somewhat easier and more intuitive to work with compared to **R**. The semantics of **U** and **R** obey the following well-known classical properties:

$$w \models \varphi\ \mathbf{U}\ \psi \quad \Leftrightarrow \quad w \models \psi \text{ or } (w \models \varphi \text{ and } w_{1..} \models \varphi\ \mathbf{U}\ \psi) \tag{8}$$

$$w \models \varphi\ \mathbf{R}\ \psi \quad \Leftrightarrow \quad w \models \psi \text{ and } (w \models \varphi \text{ or } w_{1..} \models \varphi\ \mathbf{R}\ \psi) \tag{9}$$

Let the *language* of $\varphi \in LTL$ be defined as $\mathbf{L}(\varphi) \stackrel{\text{DEF}}{=} \{w \in \mathbb{D}^\omega \mid w \models \varphi\}$. It follows that

$$(\mathbb{D}^\omega, LTL, \mathbf{L}, \bot, \top, \vee, \wedge, \neg)$$

is a Boolean algebra over $\mathbb{D}^\omega$. We also write $LTL_\mathcal{A}$ for the Boolean algebra itself.

3.2 Examples

The following examples illustrate some cases of $LTL_\mathcal{A}$ modulo various Boolean algebras $\mathcal{A}$. The first example illustrates – *at a very abstract level* – the well-known connection of integrating SAT solving into *symbolic LTL*. More concretely, BDDs can be used in combination with antichain algorithms in the underlying element algebra $\mathcal{A}$ to support efficient handling of propositional formulas in practice [52].

Example 3.1. Classical LTL over a set of atomic propositions P is $LTL_\mathcal{A}$ where $\mathcal{A}$ can be a *SAT solver* over P with $\mathbb{D} = 2^P$. A formula $\alpha \in \Psi$ is a Boolean combination over P. An element $d \in \mathbb{D}$ such that $d \models \alpha$ defines a *truth assignment* to P that makes α true. For example if $P = \{p_i\}_{i<7}$ and $\alpha = p_6$ `&` p_5 `&` p_4 `&` $(p_3$ `|` $((\text{~}p_2)$ `&` $p_1))$ then if $w \in \mathbb{D}^\omega$ is such that $w(0) = \{p_1, p_4, p_5, p_6\}$ and $w(1) = \{p_1, p_2, p_4, p_5, p_6\}$ then $w(0) \models \alpha$ but $w(1) \not\models \alpha$. Thus, for example $w \not\models \mathbf{G}\alpha$. □

While in the traditional case of LTL, as in Example 3.1, $\mathbb{D}$ may be assumed to be *finite*, in the next two examples $\mathbb{D}$ is necessarily *infinite*.

Example 3.2. Consider the LTL_{ERE} formula $\mathbf{G}((\texttt{[A-Z]+}) \rightarrow \mathbf{X}(\texttt{\d+}))$. Intuitively it says that, any nonempty string of capital letters must immediately be followed by a nonempty string of digits. After we remove all the abbreviations we get that $\psi = \bot\ \mathbf{R}\ (\neg(\texttt{[A-Z]+}) \vee \mathbf{X}(\texttt{\d+}))$. Consider the infinite sequence w such that, for all $i \in \mathbb{N}$, $w(2i) = \texttt{"HI"}$ and $w(2i+1) = \texttt{"2023"}$. We show that $w \models \psi$. Since there exists no $j \in \mathbb{N}$ such that $w_{j..} \models \bot$, in order to establish (6), we must show, for all $j \in \mathbb{N}$, if $w(i)$ is a nonempty string of capital letters then $w(i+1)$ is a nonempty string of digits. This follows directly from the definition of w.

One application of LTL_{ERE} is to monitor logs in network traffic where a log is a stream of messages and each message is a string. For example, a request must eventually be followed by a response, where regular expressions specify what requests and responses are. □

Example 3.3. Consider LTL modulo $\mathcal{A} = SMT_{\mathbb{Q}}$ where $\mathcal{A}$ is an SMT solver restricted to linear rational arithmetic. In this case $\mathbb{D}$ is the set of models for linear arithmetic formulas over rationals as $\Psi_{\mathcal{A}}$. Let α be the predicate $0 < x$ and let β be the predicate $x < 1$. Then $\beta\ \mathbf{R}\ \alpha$ states that x has to remain positive until x is less than 1. Observe that if the same formula is stated modulo $\mathcal{A} = SMT_{\mathbb{Z}}$ over *integer* linear arithmetic, then β can never release α because $0 < x$ `&` $x < 1$ is then unsatisfiable, in which case $\beta\ \mathbf{R}\ \alpha$ becomes equivalent to $\mathbf{G}\alpha$. □

3.3 Properties of $LTL_{\mathcal{A}}$ languages

We get the following characterization of the semantics of $LTL_{\mathcal{A}}$ in terms of languages, that is directly based on the formal definition (1–7) and uses Equation (8). Observe that this *is not a definition* of a language of an $LTL_{\mathcal{A}}$ formula (as (15,16) are not inductive) but a useful characterization of the properties that hold (e.g., used in Theorem 6.1).

$$\mathbf{L}(\alpha) = [[\alpha]] \cdot \mathbb{D}^\omega \tag{10}$$
$$\mathbf{L}(\varphi \wedge \psi) = \mathbf{L}(\varphi) \cap \mathbf{L}(\psi) \tag{11}$$
$$\mathbf{L}(\varphi \vee \psi) = \mathbf{L}(\varphi) \cup \mathbf{L}(\psi) \tag{12}$$
$$\mathbf{L}(\neg\varphi) = \complement(\mathbf{L}(\varphi)) \tag{13}$$
$$\mathbf{L}(\mathbf{X}\varphi) = \mathbb{D} \cdot \mathbf{L}(\varphi) \tag{14}$$
$$\mathbf{L}(\varphi\ \mathbf{U}\ \psi) = \mathbf{L}(\psi) \cup (\mathbf{L}(\varphi) \cap (\mathbb{D} \cdot \mathbf{L}(\varphi\ \mathbf{U}\ \psi))) \tag{15}$$
$$\mathbf{L}(\varphi\ \mathbf{R}\ \psi) = \mathbf{L}(\psi) \cap (\mathbf{L}(\varphi) \cup (\mathbb{D} \cdot \mathbf{L}(\varphi\ \mathbf{R}\ \psi))) \tag{16}$$

Note that (15) follows from (8) because $w_{1..} \models \varphi\ \mathbf{U}\ \psi$ iff $w \in \mathbb{D} \cdot \mathbf{L}(\varphi\ \mathbf{U}\ \psi)$. Analogously, (16) follows from (9).

Example 3.4. Consider the $LTL_{\mathcal{A}}$ formula $\varphi = \alpha \wedge \mathbf{X}\beta$ for some $\alpha, \beta \in \Psi_{\mathcal{A}}$. Then $\mathbf{L}(\varphi) = \mathbf{L}(\alpha) \cap \mathbf{L}(\mathbf{X}\beta) = \mathbf{L}(\alpha) \cap (\mathbb{D}\mathbf{L}(\beta)) = ([[\alpha]]\mathbb{D}^\omega) \cap (\mathbb{D}[[\beta]]\mathbb{D}^\omega) = [[\alpha]]\,[[\beta]]\mathbb{D}^\omega$. Let also $\psi = \neg\alpha \wedge \mathbf{X}\neg\beta$. Analogously, $\mathbf{L}(\psi) = [[\text{~}\alpha]]\,[[\text{~}\beta]]\mathbb{D}^\omega$. Let $\phi = \varphi \vee \psi$. Then $\mathbf{L}(\phi) = [[\alpha]]\,[[\beta]]\mathbb{D}^\omega \cup [[\text{~}\alpha]]\,[[\text{~}\beta]]\mathbb{D}^\omega$. □

3.4 Derivatives

The main reason for using (15) (or (16)) is that it also allows us to describe the semantics of LTL in terms of derivatives. Given $L \subseteq \mathbb{D}^\omega$ and $a \in \mathbb{D}$, the *derivative of* L *with respect to* a, $\mathbf{D}_a(L)$, is defined as follows:

$$\mathbf{D}_a(L) \stackrel{\text{DEF}}{=} \{v \mid av \in L\}$$

It follows that

$$\begin{aligned}
\mathbf{D}_a(L_1 \cup L_2) &= \mathbf{D}_a(L_1) \cup \mathbf{D}_a(L_2) && (17)\\
\mathbf{D}_a(L_1 \cap L_2) &= \mathbf{D}_a(L_1) \cap \mathbf{D}_a(L_2) && (18)\\
\mathbf{D}_a(\complement(L)) &= \complement(\mathbf{D}_a(L)) && (19)
\end{aligned}$$

Proof of (19). $\complement(\mathbf{D}_a(L)) = \complement(\{v \in \mathbb{D}^\omega \mid av \in L\}) = \{v \in \mathbb{D}^\omega \mid av \notin L\} = \{v \in \mathbb{D}^\omega \mid av \in \complement(L)\} = \mathbf{D}_a(\complement(L))$. ◻

Example 3.5. Take ϕ from Example 3.4 and let $a \in \mathbb{D}$. Then $\mathbf{D}_a(\mathbf{L}(\phi)) = \mathbf{D}_a([[\alpha]]\,[[\beta]]\mathbb{D}^\omega) \cup \mathbf{D}_a([[{\sim}\alpha]]\,[[{\sim}\beta]]\mathbb{D}^\omega)$. It follows that if $a \in [[\alpha]]$ then $\mathbf{D}_a(\mathbf{L}(\phi)) = [[\beta]]\mathbb{D}^\omega$ else $\mathbf{D}_a(\mathbf{L}(\phi)) = [[{\sim}\beta]]\mathbb{D}^\omega$. ◻

We connect the semantic definition of derivatives with a *syntactic notion of derivatives* for $LTL_{\mathcal{A}}$ in Section 6. This connection establishes the effectiveness of $LTL_{\mathcal{A}}$ by reduction to alternating Büchi automata modulo $\mathcal{A}$.

Theorem 3.6 (Decidability Theorem). *$LTL_{\mathcal{A}}$ is effective if $\mathcal{A}$ is effective.*

Proof. By applying Theorem 8.1 and Theorem 7.3. ◻

4 TRANSITION TERMS

We define the key concept of *transition terms* over infinite languages ($\mathbb{D}^\omega$) by lifting the notion of transition regexes from [41] over $\mathbb{D}^*$. We later define *symbolic derivatives* for $LTL_{\mathcal{A}}$ in terms of transition terms but at this point the definitions do not depend on $LTL_{\mathcal{A}}$, only on $\mathcal{A}$. Let Q be a nonempty (possibly infinite) set of *states* and consider the Boolean algebra $(\mathbb{D}^\omega, \mathbb{B}(Q), \mathbf{L}, q_\bot, q_\top, \vee, \wedge, \neg)$ induced by Q and some denotation function $\mathbf{L} : Q \to 2^{\mathbb{D}^\omega}$ (recall Section 2.3). Later on we will instantiate $\mathbf{L}$ for both LTL modulo $\mathcal{A}$ as well as states of alternating Büchi automata modulo $\mathcal{A}$, but at this stage the definition of $\mathbf{L}(q)$ for $q \in Q$ does not affect any of the theory developed in this section.

Transition terms $\mathbb{T}_{\mathcal{A},Q}$ (or $\mathbb{T}$ for short) are defined as expressions using the following syntactic rules (reusing the same Boolean connectives as in $\mathbb{B}(Q)$):

- if $q \in Q$ then q in $\mathbb{T}$; q is called a *leaf*;
- if f, g are in $\mathbb{T}$ then $f \vee g$, $f \wedge g$, and $\neg f$ are in $\mathbb{T}$;
- if α is in $\Psi_{\mathcal{A}}$ and f, g are in $\mathbb{T}$ then $\mathbf{ite}(\alpha, f, g)$ is in $\mathbb{T}$ and is called a *conditional*.

Observe in particular that $\mathbb{B}(Q) \subset \mathbb{T}$. We write $\mathbb{T}^+$ for $\mathbb{T}$ where complementation ($\neg f$) is not allowed; f denotes a function from $\mathbb{D}$ to $\mathbb{B}(Q)$. In the case of $\mathbb{T}^+$, f denotes a function from $\mathbb{D}$ to $\mathbb{B}^+(Q)$.

Let $f, g \in \mathcal{T\!T}$, $q \in Q$, $\alpha \in \Psi_{\mathcal{A}}$, and $a \in \mathbb{D}$. The semantics of transition terms is defined as follows:

$$q(a) \stackrel{\text{DEF}}{=} q \tag{20}$$

$$\mathbf{ite}(\alpha, f, g)(a) \stackrel{\text{DEF}}{=} \begin{cases} f(a), & \text{if } a \in [\![\alpha]\!]; \\ g(a), & \text{otherwise.} \end{cases} \tag{21}$$

$$(f \wedge g)(a) \stackrel{\text{DEF}}{=} f(a) \wedge g(a) \tag{22}$$

$$(f \vee g)(a) \stackrel{\text{DEF}}{=} f(a) \vee g(a) \tag{23}$$

$$(\neg f)(a) \stackrel{\text{DEF}}{=} \neg(f(a)) \tag{24}$$

It follows immediately from the definition that for all $\mathbf{q} \in \mathbb{B}(Q)$ and $a \in \mathbb{D}$, when $\mathbf{q}$ is viewed as a transition term then $\mathbf{q}(a) = \mathbf{q}$. We use the notion of the *transition language* of a transition term f, denoted by $\mathbf{T}(f)$:

$$\mathbf{T}(f) = \{av \mid a \in \mathbb{D}, v \in \mathbf{L}(f(a))\}$$

The semantics of the Boolean operators for transition terms is such that $\mathbf{T}(\neg f) = \complement(\mathbf{T}(f))$, $\mathbf{T}(f \vee g) = \mathbf{T}(f) \cup \mathbf{T}(g)$, and $\mathbf{T}(f \wedge g) = \mathbf{T}(f) \cap \mathbf{T}(g)$. Finally,

$$\mathbf{T}(\mathbf{ite}(\alpha, f, g)) = ([\![\alpha]\!] \cdot \mathbb{D}^{\omega} \cap \mathbf{T}(f)) \cup ([\![\neg\alpha]\!] \cdot \mathbb{D}^{\omega} \cap \mathbf{T}(g)).$$

It is critically important to note that $\mathbf{T}(\mathbf{ite}(\alpha, f, g))$ is **NOT** the same as $[\![\alpha]\!] \cdot \mathbf{T}(f) \cup \complement([\![\alpha]\!]) \cdot \mathbf{T}(g)$ because the conditions in the nested transition terms (namely, any occurring in f and g) are evaluated over the *same* input element $a \in \mathbb{D}$ that α is evaluated over.

Transition terms f and g are *equivalent*, denoted $f \equiv g$, when $\mathbf{T}(f) = \mathbf{T}(g)$. It immediately follows from the definitions that any Boolean combination of states $\mathbf{q} \in \mathbb{B}(Q)$ – *as a transition term* – is trivially equivalent to $\mathbf{ite}(\top, \mathbf{q}, q_{\perp})$, i.e.,

$$\mathbf{T}(\mathbf{q}) = \mathbf{T}(\mathbf{ite}(\top, \mathbf{q}, q_{\perp})) = \mathbb{D} \cdot \mathbf{L}(\mathbf{q})$$

The *Negation Normal Form* (*NNF*) of a transition term is computed as follows, where $q \in Q$, $f, g \in \mathcal{T\!T}$ and $\alpha \in \Psi_{\mathcal{A}}$.

$$\begin{aligned}
\mathit{NNF}(q) &= q \\
\mathit{NNF}(\neg f) &= \overline{f} \\
\mathit{NNF}(f \wedge g) &= \mathit{NNF}(f) \wedge \mathit{NNF}(g) \\
\mathit{NNF}(f \vee g) &= \mathit{NNF}(f) \vee \mathit{NNF}(g) \\
\mathit{NNF}(\mathbf{ite}(\alpha, f, g)) &= \mathbf{ite}(\alpha, \mathit{NNF}(f), \mathit{NNF}(g)) \\
\overline{q} &= \neg q \\
\overline{\neg f} &= \mathit{NNF}(f) \\
\overline{f \wedge g} &= \overline{f} \vee \overline{g} \\
\overline{f \vee g} &= \overline{f} \wedge \overline{g} \\
\overline{\mathbf{ite}(\alpha, f, g)} &= \mathbf{ite}(\alpha, \overline{f}, \overline{g})
\end{aligned}$$

NNF is used to propagate complement in a transition term into its leaves. In order to show that such propagation preserves infinite languages, the following theorem is critical:

Theorem 4.1 (Complementation Theorem). *Let $X, Y, S \subseteq \mathbb{D}$ and $L, R \subseteq \mathbb{D}^\omega$. Then the following equations hold:*

$$
\begin{array}{lrcl}
4.1(a) & X{\cdot}L \cap Y{\cdot}R & = & (X \cap Y){\cdot}(L \cap R) \\
4.1(b) & \complement(S{\cdot}L) & = & \complement(S){\cdot}L \cup \mathbb{D}{\cdot}\complement(L) \\
4.1(c) & \complement(\mathbb{D}{\cdot}L) & = & \mathbb{D}{\cdot}\complement(L) \\
4.1(d) & \complement(S{\cdot}\mathbb{D}^\omega) & = & \complement(S){\cdot}\mathbb{D}^\omega \\
4.1(e) & \complement(S{\cdot}\mathbb{D}^\omega \cap L \cup \complement(S){\cdot}\mathbb{D}^\omega \cap R) & = & S{\cdot}\mathbb{D}^\omega \cap \complement(L) \cup \complement(S){\cdot}\mathbb{D}^\omega \cap \complement(R)
\end{array}
$$

Proofs. In the following we write $u = av$ where $a = u(0)$ and $v = \lambda i.u(i+1)$.

4.1(a). For all $av \in \mathbb{D}^\omega$: $av \in X{\cdot}L \cap Y{\cdot}R \Leftrightarrow av \in X{\cdot}L$ and $av \in Y{\cdot}R \Leftrightarrow a \in X \cap Y$ and $v \in L \cap R \Leftrightarrow av \in (X \cap Y){\cdot}(L \cap R)$. □

4.1(b). ($\subseteq$): Let $av \notin S \cdot L$. Then either $a \in \complement(S)$ or else $a \in S$ and $v \notin L$. In either case $av \in (\complement(S) \cdot L) \cup (\mathbb{D} \cdot \complement(L))$.
($\supseteq$): Let $av \in (\complement(S) \cdot L) \cup (\mathbb{D} \cdot \complement(L))$. If $av \in \complement(S) \cdot L$ then clearly $av \notin S \cdot L$ because $a \notin S$. If $av \in \mathbb{D} \cdot \complement(L)$ then $v \in \complement(L)$. So, it cannot be that $av \in S \cdot L$ because then $v \in L$ but $L \cap \complement(L)$ is empty. Thus $av \in \complement(S \cdot L)$. □

4.1(c). 4.1(c) is a special case of 4.1(b) with $S = \mathbb{D}$ since $\complement(\mathbb{D}) = \emptyset$. □

4.1(d). 4.1(d) is a special case of 4.1(b) with $L = \mathbb{D}^\omega$ since $\complement(\mathbb{D}^\omega) = \emptyset$. □

4.1(e). By using 4.1(a), de Morgan's laws, and Boolean laws of distributivity.

$$
\begin{array}{rcl}
\complement(S{\cdot}\mathbb{D}^\omega \cap L \cup \complement(S){\cdot}\mathbb{D}^\omega \cap R) & = & \complement(S{\cdot}\mathbb{D}^\omega \cap L) \cap \complement(\complement(S){\cdot}\mathbb{D}^\omega \cap R) \\
& = & (\complement(S{\cdot}\mathbb{D}^\omega) \cup \complement(L)) \cap (\complement(\complement(S){\cdot}\mathbb{D}^\omega) \cup \complement(R)) \\
& \stackrel{(4.1(d))}{=} & (\complement(S){\cdot}\mathbb{D}^\omega \cup \complement(L)) \cap (S{\cdot}\mathbb{D}^\omega \cup \complement(R)) \\
& = & \begin{array}{l}(\complement(S){\cdot}\mathbb{D}^\omega \cap S{\cdot}\mathbb{D}^\omega) \cup (\complement(S){\cdot}\mathbb{D}^\omega \cap \complement(R)) \cup \\ (S{\cdot}\mathbb{D}^\omega \cap \complement(L)) \cup (\complement(L) \cap \complement(R))\end{array} \\
& \stackrel{(4.1(a))}{=} & (S{\cdot}\mathbb{D}^\omega \cap \complement(L)) \cup (\complement(S){\cdot}\mathbb{D}^\omega \cap \complement(R)) \cup (\complement(L) \cap \complement(R)) \\
& \stackrel{(*)}{=} & (S{\cdot}\mathbb{D}^\omega \cap \complement(L)) \cup (\complement(S){\cdot}\mathbb{D}^\omega \cap \complement(R))
\end{array}
$$

(*) Let $v \in \complement(L) \cap \complement(R)$. If $v(0) \in S$ then $v \in S{\cdot}\mathbb{D}^\omega \cap \complement(L)$ else $v(0) \in \complement(S)$ and so $v \in \complement(S){\cdot}\mathbb{D}^\omega \cap \complement(R)$. □

Equation 4.1(e) plays a key role in Theorem 4.2, where it is the basis for the complementation of the **ite** terms, proved by induction over the structure of transition terms. The corollaries reflect that one can always linearly transform $\neg f$ into an equivalent *dual* $\overline{f}$ of f where all complements have been propagated into the leaves.

Theorem 4.2 (NNF Theorem). *For all f in $\mathbb{T}$: (1) $f \equiv NNF(f)$ and (2) $\overline{f} \equiv \neg f$.*

Corollary 4.3. $\neg\mathbf{ite}(\alpha, f, g) \equiv \mathbf{ite}(\alpha, \neg f, \neg g)$

Proof. $\mathbf{T}(\neg\mathbf{ite}(\alpha, f, g)) \stackrel{(\text{Thm } 4.2(1))}{=} \mathbf{T}(NNF(\neg\mathbf{ite}(\alpha, f, g))) = \mathbf{T}(\mathbf{ite}(\alpha, \overline{f}, \overline{g})) \stackrel{(\text{Thm } 4.2(2))}{=} \mathbf{T}(\mathbf{ite}(\alpha, \neg f, \neg g))$. □

Corollary 4.4. *For all $a \in \mathbb{D}$, $\mathbf{L}((\neg f)(a)) = \mathbf{L}(\overline{f}(a))$.*

The following example illustrates a fundamental aspect of conditional transition terms, and is a "sneak peek" into the following sections, where transition terms are used for constructing *symbolic derivatives*.

$$\textsc{CondElim1}\ \frac{\mathbf{ite}(\alpha, f, f)}{f} \qquad \textsc{CondElim2}\ \frac{\mathbf{ite}(\alpha, f, g)}{g}\ \text{IF } \alpha \equiv \bot \qquad \textsc{CondElim3}\ \frac{\mathbf{ite}(\alpha, f, g)}{f}\ \text{IF } {\sim}\alpha \equiv \bot \qquad \textsc{ConjProp1}\ \frac{\mathbf{ite}(\alpha, f_1, g_1) \wedge \mathbf{ite}(\beta, f_2, g_2)}{\mathbf{ite}(\alpha, f_1 \wedge f_2, g_1 \wedge g_2)}\ \text{IF } \alpha \equiv \beta$$

$$\textsc{ConjProp2}\ \frac{\mathbf{ite}(\alpha, f, g) \wedge h}{\mathbf{ite}(\alpha, f \wedge h, g \wedge h)} \qquad \textsc{DeadEnd1}\ \frac{\mathbf{ite}(\alpha, \mathbf{ite}(\beta, f, g), h)}{\mathbf{ite}(\alpha, g, h)}\ \text{IF } \alpha \,\&\, \beta \equiv \bot \qquad \textsc{DeadEnd2}\ \frac{\mathbf{ite}(\alpha, \mathbf{ite}(\beta, f, g), h)}{\mathbf{ite}(\alpha, f, h)}\ \text{IF } \alpha \,\&\, {\sim}\beta \equiv \bot$$

$$\textsc{LocalDet1}\ \frac{\mathbf{ite}(\alpha, f, g) \vee \mathbf{ite}(\beta, f, g)}{\mathbf{ite}(\alpha \mid \beta, f, g)} \qquad \textsc{LocalDet2}\ \frac{\mathbf{ite}(\alpha, f_1, g_1) \vee \mathbf{ite}(\beta, f_2, g_2)}{\mathbf{ite}(\alpha, f_1 \vee f_2, g_1 \vee g_2)}\ \text{IF } \alpha \equiv \beta$$

$$\textsc{LocalDet3}\ \frac{\mathbf{ite}(\alpha, f_1, g_1) \vee \mathbf{ite}(\beta, f_2, g_2)}{\mathbf{ite}(\alpha, f_1 \vee g_2, \mathbf{ite}(\beta, g_1 \vee f_2, g_1 \vee g_2))}\ \text{IF } \alpha \,\&\, \beta \equiv \bot \qquad \textsc{Reorder}\ \frac{\mathbf{ite}(\alpha, f, g)}{\mathbf{ite}({\sim}\alpha, g, f)}$$

Fig. 1. Functional optimizations: *CondElim* = conditional elimination; *ConjProp* = conjunction propagation; *DeadEnd* = elimination of infeasible paths (dead ends); *LocalDet* = local determinization; *Reorder* = branch reordering (see text for more detail).

Example 4.5. Consider the formula ϕ from Examples 3.4 and 3.5 and suppose that $Q = LTL_{\mathcal{A}}$. By using a conditional transition term we can express the derivative of $\mathbf{L}(\phi)$ syntactically by the term $f = \mathbf{ite}(\alpha, \beta, \neg\beta)$. It follows, by using the definitions (20–24), that for all $a \in \mathbb{D}$, if $a \in [[\alpha]]$ then $f(a) = \beta$ else $f(a) = \neg\beta$. Therefore, $\mathbf{L}(f(a)) = \mathbf{D}_a(\mathbf{L}(\phi))$ because $\mathbf{L}(\beta) = [[\beta]]\mathbb{D}^\omega$ and $\mathbf{L}(\neg\beta) = [[{\sim}\beta]]\mathbb{D}^\omega$.

To illustrate Theorem 4.2, observe that $\overline{f} = \mathbf{ite}(\alpha, \neg\beta, \beta)$ and it follows that, for any $a \in \mathbb{D}$, $\mathbf{L}(\overline{f}(a)) = \complement(\mathbf{L}(f(a)))$ that is – by using 4.1(e) – if $a \in [[\alpha]]$ then $\mathbf{L}(\overline{f}(a)) = [[{\sim}\beta]]\mathbb{D}^\omega$ else $\mathbf{L}(\overline{f}(a)) = [[\beta]]\mathbb{D}^\omega$. □

5 FUNCTIONAL AND BOOLEAN TRANSITION TERM OPTIMIZATIONS

Here we focus on equivalence preserving optimizations of transition terms in $\mathcal{T}_{\mathcal{A},Q}$. We describe these optimization rules as a set of rewrite rules that simplify terms based on their logical structure as well as properties of $\mathcal{A}$. Recall that $\mathcal{A} = (\mathbb{D}, \Psi, [[\_]], \bot, \top, |, \&, {\sim})$. The core principles behind all of these rules discussed here are: to propagate operations into $\mathcal{A}$ whenever possible; to maintain satisfiability of predicates in Ψ that occur in the terms; to eliminate infeasible paths in nested conditionals; and to simplify Boolean combinations of states.

The rewrite rules are divided roughly into *two types* of rules: *functional* and *boolean* optimizations. Some of these rules are inspired by and have analogues in [39, Section 3] and some of the rules are adaptations of rules briefly discussed in [41, Section 4]. Use of further Boolean optimizations can be traced back to [29] and are discussed in [39] but are subject to restrictions (see Section 8.1).

5.1 Functional optimizations

Any simplifications that preserve the semantics (20–24) of $f \in \mathcal{T}_{\mathcal{A},Q}$, as a function from $\mathbb{D}$ to $\mathbb{B}(Q)$ are applicable to f. Recall from Section 2.3 that conjunction and disjunction are treated as commutative, associative, and idempotent operators, i.e., as *sets*. Thus, reordering of arguments or nested applications of the operator in conjunctions or disjunctions is immaterial and automatically maintains the semantics of transition terms.

$$\textsc{UnitAndCancellation}\ \frac{\top \vee f}{\top} \qquad \frac{\top \wedge f}{f} \qquad \frac{\bot \vee f}{f} \qquad \frac{\bot \wedge f}{\bot}$$

$$\textsc{Subsumption}\ \frac{f \vee (f \wedge g)}{f} \qquad \frac{f \wedge (f \vee g)}{f} \qquad\qquad \textsc{ExcludedMiddle}\ \frac{f \vee g}{\top}\ \text{if}\ f = \overline{g}$$

$$\textsc{Contradiction}\ \frac{f \wedge g}{\bot}\ \text{if}\ f = \overline{g} \qquad\qquad \textsc{ConjunctionDistribution}\ \frac{(f \vee g) \wedge h}{(f \wedge h) \vee (g \wedge h)}$$

Fig. 2. Boolean optimization. In the above rules, let $f, g \in \mathit{NNF}(\mathcal{T})$. Recall that $\overline{g}$ is the dual of g and that $\mathbf{T}(\overline{g}) = \complement(\mathbf{T}(g))$.

The optimizations make use of Boolean operations and *satisfiability* of predicates in $\Psi_{\mathcal{A}}$. Figure 1 shows the main such rewrite rules (with some symmetrical cases missing). The overall aim is to both simplify the transition function by eliminating unreachable Boolean combinations of states as well as to reduce the branching factor of transition terms (number of target states). The use of branch reordering (rule *Reorder*) obviously is only relevant if it enables subsequent use of other simplification rules, such as local determinization. Moreover, its applicability in practice also depends on the *cost of complementation* in $\mathcal{A}$.

Example 5.1. Consider $f = \mathbf{ite}(\alpha, \beta, \bot) \vee \mathbf{ite}({\sim}\alpha, \neg\beta, \bot)$. First, the second conditional branches are reordered: $f = \mathbf{ite}(\alpha, \beta, \bot) \vee \mathbf{ite}(\alpha, \bot, \neg\beta)$. Second, the choice is locally eliminated: $f = \mathbf{ite}(\alpha, \beta \vee \bot, \bot \vee \neg\beta)$. Third, $\bot$ is eliminated as the unit element of disjunction, so the final simplified transition term is $f = \mathbf{ite}(\alpha, \beta, \neg\beta)$. ◻

5.2 Boolean optimizations

Figure 2 presents simplifications that preserve the semantics of $f \in \mathcal{T}_{\mathcal{A},Q}$, as a function respecting the Boolean laws of the induced Boolean algebra $(\mathbb{D}^\omega, \mathcal{T}, \mathbf{T}, \bot, \top, \vee, \wedge, \neg)$ that are generic for any denotation $\mathbf{T}(f) \subseteq \mathbb{D}^\omega$ for $f \in \mathcal{T}$, and thus solely depend on the laws of the Boolean algebra and not on any *particular* denotation $\mathbf{T}(f)$. It will be clear in Section 8.1 why this limitation here of *not allowing particular denotations is important.*

Distribution of conjunction over disjunction enables the other rewrite rules (e.g. propagation of conjunction into conditionals and deadend elimination) to locally eliminate unreachable cases. The end result of applying the rewrites is a form of DNF of $\mathcal{T}$ where all conditionals are in the top layer and, if the notion of *leaves* is extended to $\mathbb{B}(Q)$, then all the leaves belong to $\mathbb{B}(Q)$. Top level may still include disjunctions of conditionals.

Some of the rules in Figure 2 are special cases of the rewrite rules in [39]. In particular, the rewrite rule $\varphi \leq \psi \Rightarrow (\varphi \wedge \psi) \equiv \varphi$, where $\varphi \leq \psi$ stands for $\mathbf{L}(\varphi) \subseteq \mathbf{L}(\psi)$, holds in particular for the subsumption rule, but is not admitted here as a simplification rewrite rule in its full generality (see Section 8.1).

Example 5.2. Consider the states $\varphi = \alpha\ \mathbf{R}\ \beta$ and $\psi = {\sim}\alpha\ \mathbf{U}\ {\sim}\beta$. Then the disjunction $\varphi \vee \psi$ falls under the rule of excluded middle in the context of LTL because we know that φ is the dual of ψ. So we know, without having to take into consideration what the precise denotation of those states is, that $\mathbf{L}(\varphi) \cup \mathbf{L}(\psi) = \mathbb{D}^\omega$ because they are mutually dual. ◻

6 SYMBOLIC DERIVATIVES OF $LTL_{\mathcal{A}}$

In this section, we show how the semantics of $LTL_{\mathcal{A}}$ can be realized via transition terms. In particular, the *symbolic derivative* of an $LTL_{\mathcal{A}}$ formula is defined as the following transition term in $\mathbb{T}_{\mathcal{A},LTL_{\mathcal{A}}}$. Let $\alpha \in \Psi_{\mathcal{A}}$, and $\varphi, \psi \in LTL_{\mathcal{A}}$:

$$\delta(\alpha) \stackrel{\text{DEF}}{=} \mathbf{ite}(\alpha, \top, \bot) \tag{25}$$

$$\delta(\varphi \wedge \psi) \stackrel{\text{DEF}}{=} \delta(\varphi) \wedge \delta(\psi) \tag{26}$$

$$\delta(\varphi \vee \psi) \stackrel{\text{DEF}}{=} \delta(\varphi) \vee \delta(\psi) \tag{27}$$

$$\delta(\neg\varphi) \stackrel{\text{DEF}}{=} \neg\delta(\varphi) \tag{28}$$

$$\delta(\mathbf{X}\psi) \stackrel{\text{DEF}}{=} \psi \tag{29}$$

$$\delta(\varphi \,\mathbf{U}\, \psi) \stackrel{\text{DEF}}{=} \delta(\psi) \vee (\delta(\varphi) \wedge (\varphi \,\mathbf{U}\, \psi)) \tag{30}$$

$$\delta(\varphi \,\mathbf{R}\, \psi) \stackrel{\text{DEF}}{=} \delta(\psi) \wedge (\delta(\varphi) \vee (\varphi \,\mathbf{R}\, \psi)) \tag{31}$$

We also let $\delta(\top) \stackrel{\text{DEF}}{=} \top$ and $\delta(\bot) \stackrel{\text{DEF}}{=} \bot$ and we let $\neg\top \stackrel{\text{DEF}}{=} \bot$ and $\neg\bot \stackrel{\text{DEF}}{=} \top$. We will mostly make use of the NNF of transition terms and therefore introduce the shorthand

$$\hat{\delta}(\psi) \stackrel{\text{DEF}}{=} \mathit{NNF}(\delta(\psi))$$

In this context we make use of the additional classical rule over the leaves of $\hat{\delta}(\psi)$ that

$$\mathit{NNF}(\neg\mathbf{X}\psi) \stackrel{\text{DEF}}{=} \mathbf{X}(\mathit{NNF}(\neg\psi))$$

where complement is propagated over the $\mathbf{X}$ operator. The correctness of this rule can also be seen from equations 4.1(c) and (14).

By viewing both $\mathbf{U}$ and $\mathbf{R}$ as built-in operators, we have the duality $\overline{\varphi \,\mathbf{R}\, \psi} = \overline{\varphi} \,\mathbf{U}\, \overline{\psi}$. So all *leaves* of a transition term $\hat{\delta}(\psi)$ have the form α, $\mathbf{X}\psi$, $\varphi \,\mathbf{U}\, \psi$, or $\varphi \,\mathbf{R}\, \psi$, where $\alpha \in \Psi_{\mathcal{A}}$. Observe that $\mathit{NNF}(\neg\mathbf{ite}(\alpha, \top, \bot)) = \mathbf{ite}(\alpha, \bot, \top)$ that is equivalent to $\mathbf{ite}({\sim}\alpha, \top, \bot)$. In other words, complement over atomic predicates is always propagated into $\mathcal{A}$.

The following theorem lays the foundation for the derivative based view of $LTL_{\mathcal{A}}$. The proof is by induction over the size of ϕ by case analysis over rules (25–30) coupled with the semantics of transition terms (20–24) as well as the corresponding semantics of $LTL_{\mathcal{A}}$ (10–15).

Theorem 6.1 (Derivation Theorem). *For all $\phi \in LTL_{\mathcal{A}}$ and $a \in \mathbb{D}$: $\mathbf{D}_a(\mathbf{L}(\phi)) = \mathbf{L}(\delta(\phi)(a))$.*

Example 6.2. We revisit the LTL_{ERE} formula $\psi = \mathbf{G}($`([A-Z]+)` $\rightarrow \mathbf{X}($`\d+`$))$ and illustrate the resulting derivatives together with some of the simplification rules that are being applied, while we always skip many simplifications (such as unit and cancellation laws from Figure 2) as well as trivial derivation steps such as $\delta(\top) = \top$ and $\delta(\bot) = \bot$ (as the

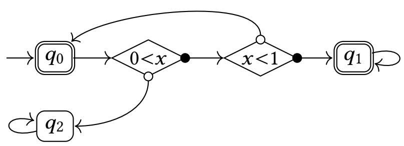

Fig. 3. A Büchi automaton $\mathcal{B}$ modulo linear arithmetic. Branching is shown as in [37] with conditions in diamonds and "then" and "else" branches starting from filled and empty circles, respectively.

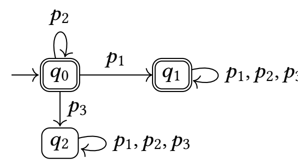

Fig. 4. Translation $\widehat{\mathcal{B}}$ of $\mathcal{B}$ to the classical setting (see Example 7.1)

special case of (25) immediately followed by condition elimination from Figure 1).

$$
\begin{array}{rcl}
\delta(\psi) & = & \delta(\bot\ \mathbf{R}\ (\neg(\texttt{[A-Z]+}) \lor \mathbf{X}(\texttt{\textbackslash d+}))) \\
& \stackrel{(31)}{=} & \delta(\neg(\texttt{[A-Z]+}) \lor \mathbf{X}(\texttt{\textbackslash d+})) \land \psi \\
& = & (\delta(\texttt{\~{}([A-Z]+)}) \lor \delta(\mathbf{X}(\texttt{\textbackslash d+}))) \land \psi \\
& = & (\mathbf{ite}(\texttt{\~{}([A-Z]+)}, \top, \bot) \lor (\texttt{\textbackslash d+})) \land \psi \\
& = & \mathbf{ite}(\texttt{\~{}([A-Z]+)}, \top, (\texttt{\textbackslash d+})) \land \psi \\
& = & \mathbf{ite}(\texttt{\~{}([A-Z]+)}, \psi, (\texttt{\textbackslash d+}) \land \psi) \\
\delta((\texttt{\textbackslash d+})) & = & \mathbf{ite}((\texttt{\textbackslash d+}), \top, \bot)
\end{array}
$$

This exhausts the derivation cases with four relevant formulas: $\{\psi, (\texttt{\textbackslash d+}), \top, \bot\}$. $\square$

7 ALTERNATING BÜCHI AUTOMATA MODULO $\mathcal{A}$ OR $ABA_{\mathcal{A}}$

Now that we have presented the symbolic approach to $LTL_{\mathcal{A}}$ via transition terms and symbolic derivatives, we turn our attention to alternating Büchi automata, show how they can be generalized using transition terms so as to be modulo $\mathcal{A}$, demonstrate a reduction to classical alternating Büchi automata, and show that nondeterministic Büchi automata modulo $\mathcal{A}$ arise as a special case. We also define deterministic Büchi automata modulo $\mathcal{A}$ as a special case that we revisit later.

An *alternating Büchi automaton modulo* $\mathcal{A}$ is a tuple $\mathcal{B} = (\mathcal{A}, Q, q^0, \rho, F)$ where Q is a finite nonempty set of *states*, $q^0 \in \mathbb{B}^+(Q)$ is an *initial state combination*, $F \subseteq Q$ is a set of *accepting states*, and $\rho : Q \mapsto \mathbb{T}^+_{\mathcal{A},Q}$ is a *transition function* [3] that maps each state into a positive transition term as defined in Section 4.

If there is a state $q \in F$ such that $\rho(q) = q$ then we fix such a state and name it $q_\top$. Analogously, if there is a state $q \in Q \setminus F$ such that $\rho(q) = q$ then we fix such a state and name it $q_\bot$. Otherwise we add such states into Q and will from here on assume that $q_\top, q_\bot \in Q$. For example in Figure 7.1(a), we have that $q_\top = q_1$ and $q_\bot = q_2$. This will simplify our formal treatment of semantics. Let $k = |Q|$.

Example 7.1. We revisit the earlier LTL modulo $\mathcal{A}$ formula $\psi = (x < 1)\ \mathbf{R}\ (0 < x)$ from Example 3.3 where $\mathcal{A}$ is a linear rational arithmetic solver. In this case $\mathbb{D}$ is the set all possible valuations for variables, for example $(x{=}51) \in \mathbb{D}$ and $(x{=}51) \in [\![(0 < x)]\!]$. The classical analogy is that classically $\mathbb{D} = 2^P$ for some finite set P of propositions and the traditional formulation $p \in a$ for $p \in P$ and $a \in \mathbb{D}$, means that a is a truth assignment that makes p true, in other words

[3] In the classical case ρ is allowed to be *partial* but here ρ must be total because $\mathbb{D}$ can be infinite, and maps elements whose transitions would otherwise not be defined to a *false* sink state $q_\bot \notin F$ such that $\rho(q_\bot) = q_\bot$.

$a \in [[p]]$ is saying the same thing in the modulo $\mathcal{A}$ context, where $[[p]]$ is the set of all models (truth assignments for p). Thus, the statement $(x{=}51) \in [[(0 < x)]]_{\mathcal{A}}$ should be clear, typically also written $(x{=}51) \models_{\mathcal{A}} (0 < x)$.

Without going into the details of the construction itself, we illustrate the equivalent Büchi automaton modulo $\mathcal{A}$ for ψ in Figure 3 the automaton is:

$$\mathcal{B} = (\mathcal{A}, \{q_0, q_1, q_2\}, q_0, \{q_0 \mapsto \mathbf{ite}((0 < x), \mathbf{ite}((x < 1), q_1, q_0), q_2), q_1 \mapsto q_1, q_2 \mapsto q_2\}, \{q_0, q_1\})$$

Then $w \in \mathbb{D}^\omega$ such as $w = (x{=}2)\cdot(x{=}\frac{1}{2})\cdot(x{=}0)^\omega$ is accepted by the automaton because the state q_1 is reached and visited infinitely often after $x = \frac{1}{2}$ and the valuation for x may remain forever 0 after that.

Let us consider the translation of $\mathcal{B}$ to the classical case $\widehat{\mathcal{B}}$ (see proof of Theorem 7.3). Consider P as the set of all minterms of predicates in ρ, i.e., all the predicates in $\{(0{<}x)\&(x{<}1), (0{<}x)\&(x{\geq}1), (0{\geq}x)\&(x{<}1), (0{\geq}x)\&(x{\geq}1)\}$ that are *satisfiable* in $\mathcal{A}$. These are $p_1 = (0{<}x)\&(x{<}1)$, $p_2 = (0{<}x)\&(x{\geq}1)$, and $p_3 = (0{\geq}x)\&(x{<}1)$. In this case, by coincidence, they also happen be the combined paths in the conditional, making the comparison easy. Here $\mathcal{A}$ is detached from the semantics of $\widehat{\mathcal{B}}$ and one would consider the classical view where for $a \in 2^P$ and $p \in P$, then $p \in a$ semantically (in terms of $\mathcal{A}$) means here that p *implies* $\bigvee_{r \in a} r$. See Figure 4. □

Before continuing with the formal development we need additional background on infinite trees that is only used in this section.

7.1 Infinite trees

Let $I = \{i \in \mathbb{N} | i < k\}$ be a set of *indices*. Elements in I^* (the Kleene closure of I) are called *nodes* where the empty sequence ϵ is called the *root* and if $x \in I^*$ and $i \in I$ then $xi \in I^*$ is the *i'th child* of x. A *Q-labeled infinite k-tree*, or *tree* for short, is a function τ from I^* to Q.

Let $q \in Q$ and let $(\tau_i)_{i<k}$ be a given sequence of k trees $\tau_i : I^* \to Q$. We use the *shorthand notation* $\langle q, (\tau_i)_{i<k}\rangle$ below to denote the following tree:

$$\langle q, (\tau_i)_{i<k}\rangle = \lambda x. \begin{cases} q, & \text{if } x = \epsilon; \\ \tau_i(u), & \text{else where } x = iu. \end{cases}$$

This is *not a representation* for a tree but a way of succinctly denoting the function. Let $\tau_\top$ be the tree $\lambda x.q_\top$ that maps all nodes to $q_\top$. In particular $\langle q_\top, (\tau_\top)_{i<k}\rangle = \tau_\top$. If we view $\mathbb{N}$ as unary numbers in $\{0\}^*$ and $k = 1$ then Q-labeled infinite 1-trees are just infinite sequences in Q^ω.

7.2 Runs and Languages

In the following it is useful to view a *disjunction* $\bigvee_{i<n} \psi_i$ as the set $\{\psi_i\}_{i<n}$. Let $aw \in \mathbb{D}^\omega$ and $q \in Q$. A *run from q for aw* is a tree $\langle q, (\tau_i)_{i<k}\rangle$ – an infinite unwinding of the transition function – where for some $\bigwedge_{i<n} q_i \in DNF(\rho(q)(a))$ and all $i < n$, τ_i is a run from q_i for w, and $\tau_i = \tau_\top$ for $n \leq i < k$. Let $\mathbf{R}_w(q)$ be the set of all runs from q for w. Then we have:

$$\mathbf{R}_{aw}(q) \quad = \quad \begin{array}{ll}\{\langle q, (\tau_0, \ldots, \tau_n, \tau_\top, \ldots, \tau_\top)\rangle \mid & (q_0 \wedge \cdots \wedge q_n) \in DNF(\rho(q)(a)), \\ & \tau_0 \in \mathbf{R}_w(q_0), \ldots, \tau_n \in \mathbf{R}_w(q_n)\}\end{array} \qquad (32)$$

Observe that taking the disjunctive normal form of $\rho(q)(a)$ *does not* affect the semantics because the semantics is invariant under *any* equivalent Boolean combination of states (e.g. by taking the *conjuctive normal form* instead), but it simplifies the statement of (32) that would alternatively have to reason about all the truth assignments to $\rho(q)(a)$ as is done in [45].[4] For example, $(q_1 \vee q_2) \wedge q_3$ is equivalent to $(q_1 \wedge q_3) \vee (q_2 \wedge q_3)$.

For example, for all w, $\mathbf{R}_w(q_\top) = \{\tau_\top\}$ and for $\alpha \in \Psi$, if $a \in [\![\alpha]\!]$, then $\mathbf{R}_{aw}(\alpha) = \{\langle \alpha, (\tau_\top)_{i<k} \rangle\}$.

A *branch* is an infinite sequence $\beta \in (I^*)^\omega$ such that $\beta(0) = \epsilon$ and, for all $n \in \mathbb{N}$, $\beta(n+1) = \beta(n)i$ for some $i \in I$. For all $q \in Q$, the *language of* q in $\mathcal{B}$ is the set of all words $w \in \mathbb{D}^\omega$ such that $\mathbf{R}_w(q)$ contains an *accepting* run τ, meaning that *all branches* of τ visit F infinitely often, formally:

$$Accepting(\tau) \stackrel{\text{DEF}}{=} \forall \beta \in Branches : \forall n \in \mathbb{N} : \exists m \geq n : \tau(\beta(m)) \in F \tag{33}$$

$$\mathcal{L}_{\mathcal{B}}(q) \stackrel{\text{DEF}}{=} \{w \in \mathbb{D}^\omega \mid \exists \tau \in \mathbf{R}_w(q) : Accepting(\tau)\} \tag{34}$$

We now lift the definition of $\mathbf{L}$ to $\mathbb{B}(Q)$ as in Section 2.3. It follows in particular for all $\mathbf{p}, \mathbf{q} \in \mathbb{B}^+(Q)$:

$$\mathcal{L}_{\mathcal{B}}(\mathbf{p} \vee \mathbf{q}) \stackrel{\text{DEF}}{=} \mathcal{L}_{\mathcal{B}}(\mathbf{p}) \cup \mathcal{L}_{\mathcal{B}}(\mathbf{q}) \tag{35}$$

$$\mathcal{L}_{\mathcal{B}}(\mathbf{p} \wedge \mathbf{q}) \stackrel{\text{DEF}}{=} \mathcal{L}_{\mathcal{B}}(\mathbf{p}) \cap \mathcal{L}_{\mathcal{B}}(\mathbf{q}) \tag{36}$$

$$\mathcal{L}(\mathcal{B}) \stackrel{\text{DEF}}{=} \mathcal{L}_{\mathcal{B}}(q^0) \tag{37}$$

Next, we prove the *transition* theorem of $\mathcal{B}$. This theorem plays a key role in many formal arguments and proofs going forward. It is the analogue of the *derivation* Theorem 6.1 of $LTL_{\mathcal{A}}$.

Theorem 7.2 (Transition Theorem). *For all $q \in Q$, $\mathcal{L}_{\mathcal{B}}(q) = \bigcup_{a \in \mathbb{D}} a \cdot \mathcal{L}_{\mathcal{B}}(\rho(q)(a))$.*

Proof. $\subseteq$: Let $aw \in \mathcal{L}_{\mathcal{B}}(q)$. From (34) it follows that there exists a run $\tau \in \mathbf{R}_{aw}(q)$ that is accepting, so all branches of τ visit F infinitely often. From (32) it follows that there exists $(q_i)_{i<n} \in \rho(q)(a)$ such that $\tau = \langle q, (\tau_i)_{i<n} \rangle$ and $\tau_i \in \mathbf{R}_w(q_i)$. So all branches of each τ_i visit F infinitely often. It follows by (34) that $w \in \mathcal{L}_{\mathcal{B}}(q_i)$ for all $i < n$ and thus by (36) that $w \in \mathcal{L}_{\mathcal{B}}(\bigwedge_{i<n} q_i)$ and finally by (35) that $w \in \mathcal{L}_{\mathcal{B}}(\rho(q)(a))$.

$\supseteq$: Let $w \in \mathcal{L}_{\mathcal{B}}(\rho(q)(a))$. It follows by the DNF assumption of $\rho(q)(a)$ and by using (35,36) that there exists $(q_i)_{i<n} \in \rho(q)(a)$ such that $w \in \mathcal{L}_{\mathcal{B}}(q_i)$ for all $i < n$. So there are accepting runs $\tau_i \in \mathbf{R}_w(q_i)$ for all $i < n$. Therefore the run $\tau = \langle q, (\tau_i)_{i<n} \rangle \in \mathbf{R}_{aw}(q)$ is also accepting because if all branches of all the τ_i visit F infinitely often then so do all branches of τ. Thus, $aw \in \mathcal{L}_{\mathcal{B}}(q)$ by using (34). □

7.3 Reduction to classical alternating Büchi automata

We make use of an effective encoding of $\mathcal{B}$ into a classical alternating Büchi automaton, that is formally defined, following [45], as a tuple $A = (\Sigma, Q, q^0, \rho, F)$ where Σ is a nonempty finite alphabet, Q is a finite set of states with $q^0 \in \mathbb{B}^+(Q)$ as an initial state combination[5], $F \subseteq Q$ as a set of accepting states, and $\rho : Q \times \Sigma \to \mathbb{B}^+(Q)$ is a transition function. Then $w \in \mathcal{L}(A)$ is defined as in (34,37) except that $\mathbb{D} = \Sigma$, so $\mathcal{L}(A) \subseteq \Sigma^\omega$.

We say that $\mathcal{B}$ is *nonempty* iff $\mathcal{L}(\mathcal{B}) \neq \emptyset$.

Theorem 7.3. *Nonemptiness of $\mathcal{B}$ is decidable if $\mathcal{A}$ is effective.*

Proof. Let $\mathcal{B} = (\mathcal{A}, Q, q^0, \rho, F)$ and let $\Gamma = \{\gamma_i\}_{i<k}$ be a finite set of all $\alpha \in \Psi$ that occur in ρ. Let $\Sigma = Minterms(\Gamma)$ (see Section 2.2). So Σ is computable because $\mathcal{A}$ is effective.

So Σ is a finite set of size at most 2^k that we now treat as a finite alphabet. For each minterm $\alpha \in \Sigma$ let $\check{\alpha}$ represent some fixed member of $[\![\alpha]\!]$, and conversely for all $a \in \mathbb{D}$, let $\widehat{a}$ denote the unique minterm $\alpha \in \Sigma$ such that $a \in [\![\alpha]\!]$. Now

[4] This is more of a matter of *style* of presentation rather than anything else, and in particular has no affect on the semantics.
[5] Here using the standard generalization that $q^0 \in \mathbb{B}^+(Q)$ instead of $q^0 \in Q$.

let $\widehat{\mathcal{B}} \stackrel{\text{DEF}}{=} (\Sigma, Q, q^0, \widehat{\rho}, F)$ where $\widehat{\rho}(q, \alpha) \stackrel{\text{DEF}}{=} \rho(q)(\breve{\alpha})$ for all $\alpha \in \Sigma$ and $q \in Q$. It follows that for all $a \in \mathbb{D}$, $\widehat{\rho}(q, \widehat{a}) = \rho(q)(a)$ because for any $f = \mathbf{ite}(\gamma, g, h)$ that occurs in ρ, we have $\gamma \in \Gamma$ so $f(a) = f(b)$ if $\widehat{a} = \widehat{b}$ because, by definition of minterms, $[[\alpha \wedge \gamma]] \neq \emptyset$ iff $[[\alpha]] \subseteq [[\gamma]]$ for all $\alpha \in \Sigma$ and $\gamma \in \Gamma$.

It follows that, for all $w \in \mathbb{D}^\omega$: $w \in \mathcal{L}(\mathcal{B})$ iff $\widehat{w} \in \mathcal{L}(\widehat{\mathcal{B}})$ where, for all $i \in \mathbb{N}$, $\widehat{w}(i) \stackrel{\text{DEF}}{=} \widehat{w_i}$. So $\mathcal{L}(\widehat{\mathcal{B}}) \neq \emptyset$ iff $\mathcal{L}(\mathcal{B}) \neq \emptyset$. The theorem follows now from reduction of nonemptiness from alternating Büchi automata to nonemptiness of nondeterministic Büchi automata [30] and decidability of nonemptiness of nondeterministic Büchi automata [34] as the special case of nondeterministic Büchi $\{1\}$-*tree* automata. ◻

An immediate question that arises here concerns the cost and implications of reducing $\mathcal{B}$ to $\widehat{\mathcal{B}}$, as defined above, and then using the algorithms developed for classical (alternating) Büchi automata also for the symbolic case. For many decision problems this would introduce an upfront worst-case exponential cost, recall that the cost of computing Σ is $O(2^{O^{\text{sat}}_{\mathcal{A}}(n)})$ where n is the size of $\mathcal{B}$, rendering the translation impractical in general, while working directly with $\mathcal{B}$ could avoid that cost completely. (The example in Figures 3 and 4 is too small to illustrate that aspect, but can very easily be made larger involving tens of predicates or even more.) The next section illustrates a particular case.

7.4 Nondeterministic Büchi automata modulo $\mathcal{A}$

We say that an alternating Büchi automaton modulo $\mathcal{A}$, $\mathcal{B} = (\mathcal{A}, Q, q^0, \rho, F)$ is a *nondeterministic Büchi automaton modulo* $\mathcal{A}$ if conjunction does not occur in ρ. In other words, when $\rho(q)(a)$ is a *disjunction* of states in Q for all $q \in Q$ and $a \in \mathbb{D}$. We then view $\rho(q)(a)$ as a *nonempty subset* of Q. In this setting the use of infinite trees becomes unnecessary because only one branch of the tree ever matters. The simplified definition of acceptance of runs in $\mathcal{A}$ is as follows. For $aw \in \mathbb{D}^\omega$ the set of all runs $\mathbf{R}_{aw}(q) \subseteq Q^\omega$ has the property (38), and where accepting runs are defined as follows:

$$\mathbf{R}_{aw}(q) \quad = \quad \{q\tau \mid p \in \rho(q)(a), \tau \in \mathbf{R}_w(p)\} \tag{38}$$

$$\mathit{Accepting}(\tau) \quad \stackrel{\text{DEF}}{=} \quad \forall n \in \mathbb{N} : \exists m \geq n : \tau(m) \in F \tag{39}$$

The definitions (34,37) remain otherwise the same, but equivalently, rely on (38,39) instead.

Example 7.4. Consider the automaton in Example 7.1 and recall it is modulo rational (not integer) arithemetic. Let $w = (x{=}2)\cdot(x{=}\frac{1}{2})\cdot(x{=}0)^\omega$. Then the run from q_0 for w is the infinite sequence r such that $r(0) = q_0$ then $r(1) = q_0$ because $\rho(q_0) = \mathbf{ite}((0{<}x), \mathbf{ite}((x{<}1), q_1, q_0), q_2)$ where $(x{=}2) \in [[0{<}x]]$ and $(x{=}2) \notin [[x{<}1]]$. Then $r(2) = q_1$ because $(x{=}\frac{1}{2}) \in [[0{<}x]]$ and $(x{=}\frac{1}{2}) \in [[x{<}1]]$. The rest $r_{3..} = q_1^\omega$. So this is an accepting run. ◻

The following theorem follows by adapting the corresponding results from [19, 20] that nonemptiness of nondeterministic Büchi Automata is decidable in linear time. Here checking satisfiability of predicates of $\mathcal{A}$ is needed to ensure feasibilty of transitions. Here, the *size* $|\mathcal{B}|$ of $\mathcal{B}$ is the total size of the representation, which also depends, not only on the number of states, but ultimately on the size of the transition function, that depends on the size of the representation of predicates in $\mathcal{A}$.

Theorem 7.5. *Nonemptiness of a nondeterministic Büchi Automata $\mathcal{B}$ modulo $\mathcal{A}$ is decidable in time $O(|\mathcal{B}|^2 + O^{sat}_{\mathcal{A}}(|\mathcal{B}|))$.*

This is one of many other decision problems, such as product, showing that first constructing $\widehat{\mathcal{B}}$ from $\mathcal{B}$ adds an upfront exponential cost $O(2^{O^{\text{sat}}_{\mathcal{A}}(|\mathcal{B}|)})$ of the alphabet Σ, as in the proof of Theorem 7.3, that may not be needed.

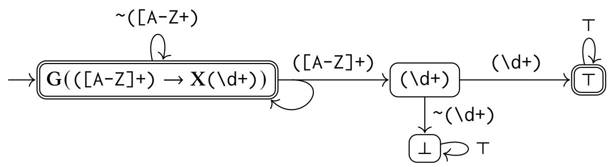

Fig. 5. Alternating Büchi automaton modulo *ERE* for $\mathbf{G}(\texttt{([A-Z]+)} \rightarrow \mathbf{X}\texttt{(\textbackslash d+)})$

7.5 Deterministic Büchi automata modulo $\mathcal{A}$

We say that an alternating Büchi automaton modulo $\mathcal{A}$, $\mathcal{B} = (\mathcal{A}, Q, q^0, \rho, F)$ is a *deterministic Büchi automaton modulo* $\mathcal{A}$ if $q_0 \in Q$ and each $\rho(q)$ is a (nested) conditional whose leaves are individual states. Complementation of classical Büchi automata is by itself a problem area that has been studied, with an exponential algorithm that works for for the nondeterministic case [2] in general, to a polynomial time algorithm for the deterministic case [28]. The question here is: *Can one make use of the NNF Theorem by dualizing states and their transition terms so that* $\overline{q} \stackrel{\text{DEF}}{=} \overline{\rho(q)}$ *for* $q \in Q$ *in order to complement* $\mathcal{B}$*?* The main problem is that the notion of *accepting states* gets lost in this translation. We will revisit this question in Section 8.3 when looking at deterministic Büchi automata modulo $\mathcal{A}$ that can arise from $LTL_{\mathcal{A}}$, and where this connection between states q and their duals $\overline{q}$ is *built-in*. For example the automaton in Figure 3 is deterministic. If we dualize the transition terms as well as the states, we will in this case obtain a deterministic Büchi automaton modulo linear arithmetic for the formula $(x \geq 1)\ \mathbf{U}\ (0 \geq x)$, that is the correct.

8 FROM $LTL_{\mathcal{A}}$ TO $ABA_{\mathcal{A}}$

We adapt the classical translation [46] using symbolic derivatives as follows. Given a start formula $\phi \in LTL_{\mathcal{A}}$ that is in NNF, let Q consist *exhaustively* of all subformulas of ϕ *and their duals* as well as $\top$ and $\bot$. For all $q \in Q$ let $\rho(q) = \delta(q)$. Observe that all leaves of $\delta(q)$ also belong to Q. Hence $|Q|$ is linear in the size of ϕ. Also, $\rho(q) \in \mathcal{T}^+_{\mathcal{A},Q}$ for all $q \in Q$, when both $\mathbf{R}$ and $\mathbf{U}$ are *built-in* operators, in which case complement never occurs and so ρ is well-defined as a transition function. Let the set of accepting states F contain $\top$ as well as all release-formulas $\varphi\ \mathbf{R}\ \psi$ in Q. The resulting alternating Büchi automaton modulo $\mathcal{A}$ for ϕ is then

$$\mathcal{B}_\phi \stackrel{\text{DEF}}{=} (\mathcal{A}, Q, \phi, \rho, F)$$

The $\mathcal{L}\mathbf{L}$ theorem of the construction shows that the intended language semantics is preserved.

Theorem 8.1 ($\mathcal{L}\mathbf{L}$ Theorem). *For all* $\phi \in LTL_{\mathcal{A}}$ *and* $q \in Q_{\mathcal{B}_\phi}$: $\mathcal{L}_{\mathcal{B}_\phi}(q) = \mathbf{L}(q)$.

The following corollary makes it explicit that *complement* is built into the automaton itself through dual states using their dual transition terms because $\rho(\overline{q}) = \overline{\rho(q)}$.

Corollary 8.2 (Duality Theorem). *Let* $\mathcal{B} = \mathcal{B}_\phi$. *For all* $q \in Q_{\mathcal{B}}$, $\mathcal{L}_{\mathcal{B}}(\overline{q}) = \complement(\mathcal{L}_{\mathcal{B}}(q))$.

Proof. $\mathcal{L}_{\mathcal{B}}(\overline{q}) \stackrel{\text{(Thm 8.1)}}{=} \mathbf{L}(\overline{q}) = \mathbf{L}(\neg q) \stackrel{(13)}{=} \complement(\mathbf{L}(q)) \stackrel{\text{(Thm 8.1)}}{=} \complement(\mathcal{L}_{\mathcal{B}}(q))$. ◻

Example 8.3. Recall the derivatives starting from the LTL_{ERE} formula $\psi = \mathbf{G}(\texttt{([A-Z]+)} \rightarrow \mathbf{X}\texttt{(\textbackslash d+)})$ from Example 6.2. The ABA_{ERE} automaton for ψ (see Figure 5) is:

$$\mathcal{B}_\psi = (ERE, \{\psi, \texttt{(\textbackslash d+)}, \top, \bot\}, \psi, \rho, \{\psi, \top\})$$

where $\rho(\top) = \top$, $\rho(\bot) = \bot$, $\rho(\psi) = \mathbf{ite}(\texttt{\textasciitilde([A-Z]+)}, \psi, \texttt{(\textbackslash d+)} \wedge \psi)$, and $\rho(\texttt{(\textbackslash d+)}) = \mathbf{ite}(\texttt{(\textbackslash d+)}, \top, \bot)$. ◻

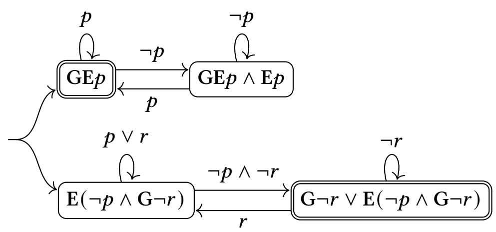

Fig. 6. Alternating Büchi automaton for $\mathbf{E}(\neg p \wedge \mathbf{G}\neg r) \wedge \mathbf{GE}p$ equivalent to the one in [52, Figure 1].

8.1 $\mathcal{L}\mathbf{L}$-invariance

Here we investigate rewrite rules that can be applied to further optimize transition terms while *simultaneously* maintaining their semantics in both worlds of $LTL_{\mathcal{A}}$ as well as $ABA_{\mathcal{A}}$. The rewrite rules will maintain the following invariant, given $\mathcal{B} = (\mathcal{A}, Q, \phi, \rho, F)$ that has been translated from an $LTL_{\mathcal{A}}$ formula,

$\mathcal{L}\mathbf{L}$-invariant: $\forall q \in Q : \mathcal{L}_{\mathcal{B}}(q) = \mathbf{L}(q)$

We believe maintaining this invariant is critical to developing a powerful system of rewrite rules for LTL that can be freely composed and applied in any context. We also want to make precise the following corollary of the $\mathcal{L}\mathbf{L}$ Theorem and [46, Theorem 22].

Corollary 8.4 (Vardi Derivative). *The LTL to ABA construction in [46] preserves $\mathcal{L}\mathbf{L}$-invariance.*

Our rewrite rules will be used to optimize transition terms $\rho(q)$ *on-the-fly*, thus transforming $\mathcal{B}$ as it is being constructed. The main purpose of the rewrite rules is to simplify and minimize the representation of the transition terms so as to reduce the number of states, then avoid *alternation*, and finally, to avoid *nondeterminism*.

While Vardi's construction [46] itself guarantees this invariant, subsequent work on simplifications has focused on optimizations on the automata level while throwing away the LTL view [22, 52]. In our world, the semantics of $\mathcal{B}_\phi$ are *defined in terms of derivatives of* ϕ instead of being just *constructed from* ϕ, which means that the only way to modify the semantics of $\mathcal{B}$ is to rewrite the transition terms.

Let us consider some rewrite rules from [39]. The rule $\varphi \leq \psi \Rightarrow (\varphi \wedge \psi) \equiv \varphi$ where $\varphi \leq \psi$ (ψ *subsumes* φ) means that if $\mathbf{L}(\varphi) \subseteq \mathbf{L}(\psi)$ then $\mathbf{L}(\varphi \wedge \psi) = \mathbf{L}(\varphi) \cap \mathbf{L}(\psi) = \mathbf{L}(\varphi)$. Subsumption can often be detected syntactically, as for example in the Boolean subsumption rule in Section 5. Another example is $(\mathbf{GE}\psi) \wedge \mathbf{E}\psi$ that is LTL-equivalent to $\mathbf{GE}\psi$. However,

in $\mathcal{B}$ the state $\mathrm{GE}\psi$ *is **accepting** while* $\mathrm{E}\psi$ *is **not accepting**.*

Therefore, applying general subsumption that relies on LTL semantics alone, to replace $(\mathbf{GE}\psi) \wedge \mathbf{E}\psi$ by $\mathbf{GE}\psi$ would in fact violate $\mathcal{L}\mathbf{L}$-invariance (see Example 8.9). Some rules that preserve $\mathcal{L}\mathbf{L}$-invariance are: $\mathbf{X}\psi \wedge \mathbf{X}\varphi = \mathbf{X}(\psi \wedge \varphi)$ and $\mathbf{EX}\varphi = \mathbf{XE}\varphi$. A full analysis of which rules in [39] preserve $\mathcal{L}\mathbf{L}$-invariance and which don't is beyond the scope of this work, but we identify a new class of rules next.

Example 8.5. To give a taste of what these rewrites enable, consider the LTL formula in [52, Example], which in NNF is $\mathbf{E}(\neg p \wedge \mathbf{G}\neg r) \wedge \mathbf{GE}p$. Figure 6 shows the corresponding alternating Büchi automaton with the rules from Sections 8.2 and 8.3. Notice that the automaton is *co-deterministic* – its complement is deterministic – apart from the conjunctive initial state the remaining parts are deterministic, while the one in [52, Figure 1] has both alternation and nondeterminism in the transitions themselves. Observe that deciding nonuniversality of the co-deterministic case can be achieved by deciding nonemptiness of its complement. □

8.2 Suspension

We take the definition of *suspendable* formulas as a practically important *syntactic* subclass of LTL formulas from [5] (originally called *alternating* formulas in [6]) to $LTL_{\mathcal{A}}$. We make use of the following lemma that carries over to $LTL_{\mathcal{A}}$, as suspendability does not depend on any properties of $\mathcal{A}$:

Lemma 8.6 ([6, Lemma 2]). *If ξ is suspendable then $w \models \xi \Leftrightarrow w_{n..} \models \xi$ for all $n \in \mathbb{N}$ and $w \in \mathbb{D}^{\omega}$.*

In particular, it follows that any suspendable formula ξ is equivalent to $\mathbf{X}\xi$, which interacts well with the derivative of ξ that can now also be suspended because $\delta(\mathbf{X}\xi) = \xi$.

We use Theorem 8.8 below to reduce alternation in $\mathcal{B}$, using the concept of suspendable formulas. We introduce a variant of δ that applies the following rules when it encounters the cases when the first argument of $\diamond \in \{\wedge, \vee\}$ is any $\mathbf{U}$-formula, $\mathbf{R}$-formula, or $\mathbf{X}$-formula, and the second argument ξ is suspendable:

$$\begin{aligned}
\delta((\varphi \,\mathbf{U}\, \psi) \diamond \xi) &\stackrel{\text{DEF}}{=} \delta(\psi \diamond \xi) \vee (\delta(\varphi \diamond \xi) \wedge ((\varphi \,\mathbf{U}\, \psi) \diamond \xi))\\
\delta((\varphi \,\mathbf{R}\, \psi) \diamond \xi) &\stackrel{\text{DEF}}{=} \delta(\psi \diamond \xi) \wedge (\delta(\varphi \diamond \xi) \vee ((\varphi \,\mathbf{R}\, \psi) \diamond \xi))\\
\delta((\mathbf{X}\varphi) \diamond \xi) &\stackrel{\text{DEF}}{=} \varphi \diamond \xi
\end{aligned}$$

Correctness of the rule for the $\mathbf{X}$-formula follows immediately from suspendability of ξ by Lemma 8.6 that essentially states that $\mathbf{L}(\xi) = \mathbb{D}{\cdot}\mathbf{L}(\xi)$ in a slightly more generalized form. For $\mathbf{U}$ and $\mathbf{R}$ we use the following lemma.

Lemma 8.7. *If ξ is suspendable, $\diamond \in \{\wedge, \vee\}$, and $\blacklozenge \in \{\mathbf{U}, \mathbf{R}\}$ then $\mathbf{L}((\varphi \blacklozenge \psi) \diamond \xi) = \mathbf{L}((\varphi \diamond \xi) \blacklozenge (\psi \diamond \xi))$.*

In terms of alternating Büchi automata this implies that we can treat $(\varphi \blacklozenge \psi) \diamond \xi$ consisting of two states the same way as the single state $(\varphi \diamond \xi) \blacklozenge (\psi \diamond \xi)$, which is precisely the effect of the variant of δ when used to define ρ.

Theorem 8.8 (Suspension Theorem). *If ϕ is a state then for any suspendable state ξ and $\diamond \in \{\wedge, \vee\}$, if $q = \phi \diamond \xi$ is included as a state of $\mathcal{B}$ with the transition term $\rho(q) = \delta(q)$ with the variant of δ then $\mathcal{L}_{\mathcal{B}}(q) = \mathbf{L}(q)$.*

When we apply the suspension theorem then the corresponding Boolean combinations $q = \phi \diamond \xi$ are *elevated* to the status of being *states*, meaning that $\rho(q)$ is now *defined*, and in this way we reduce the correponding alternation of Boolean operations between states. Elevation of states is related to promotion of formulas to states [43] but it is natural here to treat the states themselves as formulas in order to maintain reuse of all the $\mathit{TT}_{\mathcal{A}}$ algebra simplification rules. In general, certain positive Boolean combinations of states can now themselves be elevated to states. Therefore, we need to be clear about the *accepting condition* of an elevated state in $\mathbb{B}^{+}(Q)$. We consider states in *normalized* form where in particular $(\varphi \,\mathbf{R}\, \psi) \diamond \xi$ is normalized to $(\varphi \diamond \xi) \,\mathbf{R}\, (\psi \diamond \xi)$ and is therefore an *accepting state*. All other states besides $\top$ that in normalized form are not $\mathbf{R}$-states are nonaccepting.

We illustrate suspension in the following example, where $p \in \Psi$ and both $\mathbf{GE}p$ as well as $(\mathbf{E}p) \wedge \mathbf{GE}p$ are suspendable formulas (cf [5]).

Example 8.9 (Infinitely often p). Consider $\phi = \mathbf{GE}p$ for some $p \in \Psi$. The derivation steps of ϕ are as follows:

$$\begin{aligned}
\delta(\mathbf{E}p) &= \delta(\top \,\mathbf{U}\, p) = \delta(p) \vee (\delta(\top) \wedge \mathbf{E}p) = \mathbf{ite}(p, \top, \bot) \vee \mathbf{E}p = \mathbf{ite}(p, \top, \mathbf{E}p)\\
\delta(\phi) &= \delta(\bot \,\mathbf{R}\, \mathbf{E}p) = \delta(\mathbf{E}p) \wedge (\delta(\bot) \vee \phi) = \mathbf{ite}(p, \top, \mathbf{E}p) \wedge \phi = \mathbf{ite}(p, \phi, (\mathbf{E}p) \wedge \phi)
\end{aligned}$$

Observe here that rewriting $(\mathbf{E}p) \wedge \phi$ into ϕ, by using the LTL semantics that $\mathbf{L}((\mathbf{E}p) \wedge \mathbf{GE}p) = \mathbf{L}(\mathbf{GE}p)$, would collapse the transition term $\delta(\phi)$ into an *incorrect* self-loop.

(a) Alternating (b) Deterministic

Fig. 7. Büchi automata for $\mathbf{GE}p$

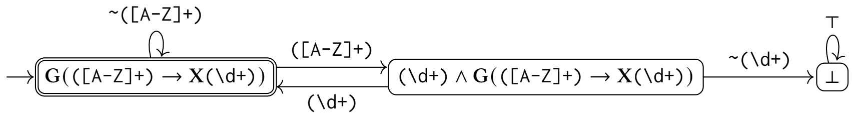

Fig. 8. Deterministic Büchi automaton modulo *ERE* for $\mathbf{G}($`([A-Z]+)` $\rightarrow \mathbf{X}$`(\d+)`$)$

See Figure 7a where $\mathbf{GE}p$ is illustrated as $ABA_{\mathcal{A}}$. Here ϕ is suspendable, so we can treat $\psi = (\mathbf{E}p) \wedge \phi$ as a (non-accepting) state. We get the following derivation steps for ψ by using the updated variant of δ and Boolean simplifications:

$$\begin{aligned}
\delta(\psi) &= \delta((\top \ \mathbf{U}\ p) \wedge \phi) = \delta(p \wedge \phi) \vee (\delta(\top \wedge \phi) \wedge \psi) \\
&= (\delta(p) \wedge \delta(\phi)) \vee (\delta(\phi) \wedge \psi) \\
&= (\mathbf{ite}(p, \top, \bot) \wedge \mathbf{ite}(p, \phi, \psi)) \vee (\mathbf{ite}(p, \phi, \psi) \wedge \psi) \\
&= \mathbf{ite}(p, \phi, \bot) \vee \mathbf{ite}(p, \phi \wedge \psi, \psi) \\
&= \mathbf{ite}(p, \phi \vee (\phi \wedge \psi), \psi) = \mathbf{ite}(p, \phi, \psi)
\end{aligned}$$

We have *eliminated* alternation fully and have obtained a *deterministic* automaton with initial state ϕ, states $Q = \{\phi, \psi\}$, final states $F = \{\phi\}$ and transition function ρ such that $\rho(\phi) = \mathbf{ite}(p, \phi, \psi)$ and $\rho(\psi) = \mathbf{ite}(p, \phi, \psi)$. See Figure 7b. □

In the following we illustrate use of a specific class of a non-suspendable formulas: $\psi = \mathbf{G}(\alpha \rightarrow \mathbf{X}\beta)$ where $\alpha, \beta \in \Psi_{\mathcal{A}}$ and $\alpha \ \&\ \beta$ is *unsatisfiable* then $\beta \wedge \psi$ can also be elevated to a state and maintains the $\mathcal{L}\mathbf{L}$-invariant. This illustrates that one can develop domain specific rules for state elevation that go beyond the suspendable case.

Example 8.10. We revisit the automaton constructed in Example 8.3 (Figure 5) and observe that the conjunction `(\d+)` $\wedge\ \phi$, where $\phi = \mathbf{G}($`([A-Z]+)` $\rightarrow \mathbf{X}$`(\d+)`$)$, can be elevated as it fits the pattern above. We get the following composed transition term for `(\d+)` $\wedge\ \phi$:

$$\begin{aligned}
\delta(\texttt{(\textbackslash d+)} \wedge \phi) &= \delta(\texttt{(\textbackslash d+)}) \wedge \delta(\phi) \\
&= \mathbf{ite}(\texttt{(\textbackslash d+)}, \top, \bot) \wedge \mathbf{ite}(\texttt{\~{}([A-Z]+)}, \phi, \texttt{(\textbackslash d+)} \wedge \phi) \\
&= \mathbf{ite}(\texttt{(\textbackslash d+)}, \mathbf{ite}(\texttt{\~{}([A-Z]+)}, \phi, \texttt{(\textbackslash d+)} \wedge \phi), \bot) \\
&= \mathbf{ite}(\texttt{(\textbackslash d+)}, \phi, \bot)
\end{aligned}$$

The last step uses the DeadEnd2 rule and the step before uses the ConjProp2 rule (see Figure 1) in combination with basic Boolean rewrite rules from Figure 2. We have now reached a fixpoint and the resulting *deterministic* Büchi automaton modulo *ERE* is shown in Figure 8 with `(\d+)` $\wedge\ \phi$ as the middle state. □

(a) Nondeterministic (b) Deterministic

Fig. 9. Büchi automata for $\mathbf{EG}r$

Theoretically, deciding nonemptiness in *ERE* is nonelementary [15, 42], which implies that LTL_{ERE} is also nonelementary. However, we believe that typical use cases fall in the much smaller sublass $\mathbb{B}(RE)$ where *RE* is the class of standard regular expressions (without complement or intersection). The complexity of $\mathbb{B}(RE)$ it is not nonelementary but it is PSPACE-hard [26, 27].

8.3 Elevated states

Here some of the states of $\mathcal{B} = (\mathcal{A}, Q, q_0, \rho, F)$ can in general be *elevated* formulas and thus in general $Q \subset NNF(LTL_{\mathcal{A}})$ with the definition of accepting states as defined above, thus F is defined *as the accepting states in* Q (as opposed to an arbitrary subset of Q). Let the *dual of* $\mathcal{B}$ be the automaton $\overline{\mathcal{B}}$:

$$\begin{array}{rrcl} & \overline{Q} & \stackrel{\text{DEF}}{=} & \{\overline{q} \mid q \in Q\} \\ \text{for } q \in Q: & \overline{\rho}(\overline{q}) & \stackrel{\text{DEF}}{=} & \overline{\rho(q)} \\ & \overline{\mathcal{B}} & \stackrel{\text{DEF}}{=} & (\mathcal{A}, \overline{Q}, \overline{q_0}, \overline{\rho}, \{p \in \overline{Q} \mid p \text{ is accepting}\}) \end{array}$$

It follows from the $\mathcal{L}\mathbf{L}$-invariant of $\mathcal{B}$ and elvation of states that for all elevated states $q \vee p$ and $q \wedge p$, also p and q are states and that $\mathcal{L}_{\mathcal{B}}(p \vee q) = \mathcal{L}_{\mathcal{B}}(p) \cup \mathcal{L}_{\mathcal{B}}(q)$ and $\mathcal{L}_{\mathcal{B}}(p \wedge q) = \mathcal{L}_{\mathcal{B}}(p) \cap \mathcal{L}_{\mathcal{B}}(q)$. Moreover, for all $q \in Q$, $\mathcal{L}_{\mathcal{B}}(q) = \mathbf{L}(q)$.

Recall from Section 7.5 that $\mathcal{B}$ being *deterministic* means that the transition term $\rho(q)$ for each state is a nested conditional whose leaves are again states, here considering leaves as elevated states, i.e., viewing ρ as a function from $\mathbb{B}^{+}(Q)$ to $\mathcal{T}_{\mathcal{A},\mathbb{B}^{+}(Q)}$ and if $\rho(q)$ is defined then q is considered as a state.

Theorem 8.11 (Elevated Duality Theorem). *If $\mathcal{B} = (\mathcal{A}, Q, q_0, \rho, F)$ is a Büchi automaton for $LTL_{\mathcal{A}}$ then $\overline{\mathcal{B}}$ is a Büchi automaton for $LTL_{\mathcal{A}}$ such that, for all $q \in Q$, $\mathcal{L}_{\overline{\mathcal{B}}}(\overline{q}) = \complement(\mathcal{L}_{\mathcal{B}}(q))$. If $\mathcal{B}$ is deterministic then $\overline{\mathcal{B}}$ is deterministic.*

Proof. $\overline{\mathcal{B}}$ is clearly well-defined. The main argument, based on the invariant above, is that $\rho(p_1 \vee q_2) \equiv \rho(p_1) \vee \rho(q_2)$ and $\rho(p_1 \wedge q_2) \equiv \rho(p_1) \wedge \rho(q_2)$. We can therefore lower $\mathcal{B}$ into an equivalent alternating Büchi automaton whose only accepting states are $\top$ and $\mathbf{R}$-formulas. We then apply the Duality Theorem (Corollary 8.2) to dualize all these terms, effectively flipping $\wedge$ and $\vee$, $\top$ and $\bot$, $\mathbf{R}$ and $\mathbf{U}$ in that process. We then symmetrically elevate the dualized transition terms back into $\overline{\rho}$ and observe also that the dualization is captured precisely by the definition of accepting states in $\overline{\mathcal{B}}$.

Determinism is preserved because $\overline{\mathbf{ite}(\alpha, f, g)} = \mathbf{ite}(\alpha, \overline{f}, \overline{g})$, i.e., the dual of any nested conditional remains a nested conditional, essentially with an identical branching structure. The statement follows. ◻

As an immediate consequence we get a *linear*, essentially trivial, complementation algorithm that, moreover, preserves determinism.

Example 8.12 (Stable r). Consider $\mathbf{EG}r$ for some $r \in \Psi$. We here make use of the duality that $\mathbf{EG}r = \overline{\mathbf{GE}p}$ when $p = {\sim}r$. We get, reusing Example 8.9, that

$$\begin{array}{rcl} \delta(\mathbf{EG}r) & = & \overline{\delta(\mathbf{GE}p)} = \overline{\mathbf{ite}(p, \mathbf{GE}p, (\mathbf{E}p) \wedge \mathbf{GE}p)} = \mathbf{ite}(p, \overline{\mathbf{GE}p}, \overline{(\mathbf{E}p) \wedge \mathbf{GE}p}) = \mathbf{ite}({\sim}r, \mathbf{EG}r, (\mathbf{G}r) \vee \mathbf{EG}r) \\ \delta((\mathbf{G}r) \vee \mathbf{EG}r) & = & \delta(\overline{(\mathbf{E}p) \wedge \mathbf{GE}p}) = \overline{\mathbf{ite}(p, \mathbf{GE}p, (\mathbf{E}p) \wedge \mathbf{GE}p)} = \mathbf{ite}({\sim}r, \mathbf{EG}r, (\mathbf{G}r) \vee \mathbf{EG}r) \end{array}$$

where the formula $(\mathbf{G}r) \vee \mathbf{EG}r$ is treated as an accepting state (the dual state of $(\mathbf{E}p) \wedge \mathbf{GE}p$) rather than *two* states. ◻

Example 8.12 makes use of Theorem 8.11 to essentially trivially complement the deterministic automaton in Figure 7b to obtain the deterministic automaton in Figure 9b, that can now be contrasted against the equivalent nondeterministic automaton in Figure 9a.

Observe that Theorem 8.11 can potentially also be applied in the context of classical symbolic LTL as $LTL_{\mathcal{A}}$ where $\mathcal{A}$ is a SAT solver (or BDD solver) over a set P of atomic propositions, where $\Psi_{\mathcal{A}} = \mathbb{B}(P)$ and $\mathbb{D} = 2^P$.

> *This assumes that no other transformation has broken $\mathcal{L}$**L**-invariant, which would cause the link between states and formulas to be **lost in translation**.*

In particular, for each state q, each single transition $\rho(q, r) \in \mathbb{B}^+(Q)$ for $r \in P$ becomes the conditional $\mathbf{ite}(r, \rho(q, r), \bot)$ and the disjunction of all such conditionals can be rewritten into a single nested conditional, e.g. by applying the simplification rules from Section 5.1, that is then used as the definition of $\rho(q)$ in $\mathcal{B}_{\mathcal{A}}$.

9 RELATED WORK

9.1 Derivatives

Transition regexes for extended regular expressions [41], a symbolic generalization of *Brzozowski derivatives* [9], is one of the inspirations behind our work here. In analogy we view transition terms for LTL as a symbolic generalization of *Vardi's derivatives for LTL* (see Section 8). However, there are fundamental differences between finite and infinite sequences, which carry over to the theory here. First, as an example, the complement law in Theorem 4.1(c) that is used frequently in many proofs, does not hold for L over finite sequences (because ϵ is missing from the right side). Second, one of the key derivation steps for regular expressions is for loops, $\delta(R^*) = \delta(R){\cdot}R^*$, and introduces concatenation and special *lift* rules to propagate concatenation, which must now interact correctly with complement also. Here concatenation does not arise, while the derivation rules for **U** and **R** give rise to loops (analogously to R^*) but with very different semantic properties.

The main theorems regarding infinite sequences are Theorem 4.1(e) and the negation-normal-form (NNF) Theorem 4.2 for transition terms. For transition regexes [41, Lemma 4.2], negation-normal-form comes from a simple observation that any negated ite-term $\neg\mathbf{ite}(\alpha, r_1, r_2)$ in SMT whose type is a regular expression has (by definition) the same interpretation as $\mathbf{ite}(\alpha, \neg r_1, \neg r_2)$. Many simplification laws in regular expressions are also easier to apply because the language semantics of regular expressions and automata go hand-in-hand, so there is no need for any special treatment as in Section 8.1 (in particular any form of *subsumption* can always be applied).

Here $\mathcal{L}$**L**-invariance must be maintained by rewrite rules. Our DNF form of transition terms is on the surface similar to the DNF of transition regexes in [41], but with the crucial difference that alternation (if any) has been incrementally eliminated in the latter and often amounts to dealing with incremental unfolding into NFAs, as a symbolic generalization of *Antimirov derivatives* [3]. Such incremental unfolding is now integrated into the core of the regex decision procedure in Z3 [16]. Whether an analogous alternation elimination procedure exists for $LTL_{\mathcal{A}}$ based $ABA_{\mathcal{A}}$

is an active research topic for us, and we are currently investigating possible generalizations from works on classical alternation elimination algorithms [8, 30] that would ideally also integrate incrementally with derivation rules for $LTL_{\mathcal{A}}$ and maintain $\mathcal{L}\mathbf{L}$-invariance.

9.2 Monitoring

LTL based monitoring is studied in [38] that introduces a *coalgebraic* method where monitoring-equivalence of LTL formulas is based on experimental indistinguishability and coinduction is used for proving monitoring-equivalence. The method is further used to generate monitors in the form of deterministic finite automata. Moreover, in its core, the work also uses *derivatives* of LTL [24], although not in the generalized form of symbolic derivatives modulo $\mathcal{A}$, but for a finite set of atomic propositions. This work uses the decision procedure from [25] to canonicalize propositional formulas through a rewrite-system that is Church-Rosser and terminating (modulo associativity and commutativity of Boolean operators, including exclusive-OR).

9.3 Comparison with Standard Construction

Vardi [44, Theorem 14, Proof] is the first LTL to alternating Büchi automata construction defined in terms of a step-wise unwinding with a similar structure to our derivatives. This construction is not symbolic, as it uses the next element to directly compute a Boolean combination of successor states.

Gastin and Oddoux [22] modify Vardi's construction to produce alternating co-Büchi automata instead, and they take a step towards a symbolic representation by representing transitions as relations, although treatment of the alphabet is still non-symbolic. This representation allows them to develop simplifications based on relational reasoning for eliminating implied transitions and equivalent states *on-the-fly*. In the context of our work, since these simplifications operate directly on the representation of the transition relation, the direct correspondence between formulas and states is lost. Even if $\mathcal{L}\mathbf{L}$-invariance (see Section 8.1) would be maintained in some form, it is unclear how these rules would compose with the syntactic LTL-level simplifications presented in this paper.

Wulf et al. [52] define *symbolic alternating Büchi automata* (sABW) where transition relations are Boolean combinations of literals and successor states. They further develop incremental satisfiability and model checking methods using BDDs both as the alphabet theory and to represent sets of states. They do not give the construction from LTL to sABW, but instead refer to the presentations in [22, 46].

Tsay and Vardi [43] give a full construction of LTL to symbolic alternating co-Büchi automata. To compare their style of construction with the one in this paper, we show the following example of LTL to sABW for $\varphi = \mathbf{GE}a$. The example uses the style of [43], but uses the acceptance of the construction in [44] to produce a Büchi instead of a co-Büchi automaton.

Example 9.1. Transition semantics are defined using a *one-step expansion* function *exp* [43, Definition 12], which plays a similar role as Equations 25-31, factoring out requirements for the current input element. The relevant expansions for φ are $exp(\mathbf{GE}a) = (a \vee \mathbf{XE}a) \wedge \mathbf{XGE}a$ and $exp(\mathbf{E}a) = a \vee \mathbf{XE}a$.

States are labeled by *elementary subformulae*, i.e., literals and temporal operators, but not Boolean combinations thereof. For φ the states are "$\mathbf{GE}a$" (the initial state) and "$\mathbf{E}a$". The transition relation is then formed by replacing $\mathbf{X}$ operators with states in the one-step expansions:

$$\delta(\text{"}\mathbf{GE}a\text{"}) = (a \vee \text{"}\mathbf{E}a\text{"}) \wedge \text{"}\mathbf{GE}a\text{"} \qquad \delta(\text{"}\mathbf{E}a\text{"}) = a \vee \text{"}\mathbf{E}a\text{"}$$

$"\mathbf{GE}a"$ is the only final state. The resulting automaton is similar to the one Figure 7a, although less deterministic as $\delta("\mathbf{GE}a")$ includes $"\mathbf{E}a" \wedge "\mathbf{GE}a"$ as a successor for all input elements. An explicit $"\top"$ state is also not present, as sABW are allowed to be non-total.

While the construction above is quite similar to the one in our work, there is a key difference in how the transitions are represented: total functions built on **ite**-terms cleanly separate evaluation of transitions the target state formulas. We believe that our use of **ite**-terms makes developing simplifications based on a system of syntactic rewrite rules natural. For example, Theorem 8.8 does not directly apply to $(a \vee "\mathbf{E}a") \wedge "\mathbf{GE}a"$. While the opportunity could be exposed in DNF form, we find imposing that cumbersome. ◻

Muller, Saoudi and Schupp [31] were the first to state and prove the theorem that LTL can be translated to Büchi automata. The proof, however, does not use an inductive unwinding of LTL formula but composes automata from subformulas instead, which makes it quite different from our derivatives. Moreover, the construction uses weak alternating automata over trees and a further result from [32] to then reduce weak alternating automata to Büchi automata. As far as we know, derivatives have not been studied for finite or infinite tree languages, so the concept of what a transition term would mean in the context of tree languages or tree automata is unclear.

9.4 Duality

A general question that arises is how transition terms and their built-in duality principle can be taken advantage of or shed new light on other areas of reasoning with classical Büchi automata when cast in terms of modulo $\mathcal{A}$. This could impact algorithms for deterministic Büchi automata [7, 21], deterministic generalized Büchi automata [4], as well as generalized Büchi automata [36] and transition-based generalized Büchi automata [13] where the latter arise naturally from LTL [18] and could thus potentially directly benefit from duality. Importance of duality is also studied in the context of weak alternating automata in [31] with key relationships established to Büchi automata. The latter work is also related to study of duality in the context of infinite *tree* automata [33].

9.5 Tableau

Tableau based techniques for LTL were initially studied by Wolper [49–51]. A further extension of tableau based technique for LTL is introduced in [12] using on-the-fly expansion of *transition Büchi automata*. The key technique there is also rooted in what we call here Vardi derivatives, that are called *fundamental identities* of Boolean variables in that context, that reflect how the variables for the subformulas are created inductively. The if-the-else aspect of transition terms is not present there.

$LTL_{\mathcal{A}}$ is fundamentally an automata-based based technique assisted by $\mathcal{A}$ as a solver. We do not believe it is possible in general to take an arbitrary effective Boolean algebra $\mathcal{A}$ and view it as part of a generic tableau procedure modulo $\mathcal{A}$. It is an open and active research area, part of a general effort to combine first-order deduction with modulo theories with many open challenges of its own [10].

9.6 General

One question that has eluded us is if Theorem 4.1(e) can be deduced from the general theory of ω-languages [40, 48]. The recent work in [23] studies LTL modulo theories but over *finite* strings, so we believe that the work in [41] could be a possible link to transition terms over $\mathbb{D}^*$ in that study. First-Order LTL is introduced in [1] that is in general *undecidable*. LTL-EF [11] is a recent extension of First-Order LTL with *event freezing* functions operators, the logic can be interpreted with different models of time using SMT techniques. In contrast, $LTL_{\mathcal{A}}$ is not first-order LTL, in particular, predicates from $\mathcal{A}$ can not be related at the level of individual variables accross state boundaries.

10 CONCLUSION

We have shown how the concept of symbolic derivatives can be used to define a symbolic semantics for linear temporal logic (LTL) and alternating Büchi automata, via a shared representation of transition terms. The semantics is parameterized by an effective Boolean algebra for the base alphabetic domain, which enables it to apply to ω-languages and infinite alphabets in an algebraically well-defined and precise manner. This framework allows syntactic rewrite rules for LTL to be applied *on-the-fly* during automata construction when they simultaneously respect semantics of LTL formulas and their alternating Büchi automata. We present several of these rules and believe there is a rich landscape of optimization to be discovered.

REFERENCES

[1] 1992. *The Temporal Logic of Reactive and Concurrent Systems - Specification.* Springer.

[2] P. Wopler A. P. Sistla, M. Y. Vardi. 1985. The complementation problem for Büchi automata, with applications to temporal logic. In *Proc. 12th Internat. Coil. on Automata, Languages and Programming (LNCS).* Springer.

[3] Valentin Antimirov. 1995. Partial Derivatives of Regular Expressions and Finite Automata Constructions. *Theoretical Computer Science* 155 (1995), 291–319.

[4] Souheib Baarir and Alexandre Duret-Lutz. 2014. Mechanizing the Minimization of Deterministic Generalized Büchi Automata. In *Proceedings of the 34th IFIP International Conference on Formal Techniques for Distributed Objects, Components and Systems (FORTE'14) (Lecture Notes in Computer Science, Vol. 8461).* Springer, 266–283. https://doi.org/10.1007/978-3-662-43613-4_17

[5] Tomáš Babiak, Thomas Badie, Alexandre Duret-Lutz, Mojmír Křetínský, and Jan Strejček. 2013. Compositional Approach to Suspension and Other Improvements to LTL Translation. In *Proceedings of the 20th International SPIN Symposium on Model Checking of Software (SPIN'13) (Lecture Notes in Computer Science, Vol. 7976).* Springer, 81–98. https://doi.org/10.1007/978-3-642-39176-7_6

[6] Tomás Babiak, Mojmír Kretínský, Vojtech Rehák, and Jan Strejcek. 2012. LTL to Büchi Automata Translation: Fast and More Deterministic. In *Tools and Algorithms for the Construction and Analysis of Systems - 18th International Conference, TACAS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings (Lecture Notes in Computer Science, Vol. 7214)*, Cormac Flanagan and Barbara König (Eds.). Springer, 95–109. https://doi.org/10.1007/978-3-642-28756-5_8

[7] C. Baier and J.-P. Katoen. 2008. *Principles of Model Checking.* MIT Press.

[8] Udi Boker, Orna Kupferman, and Adin Rosenberg. 2010. Alternation Removal in Büchi Automata. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 6199)*, Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis (Eds.). Springer, 76–87. https://doi.org/10.1007/978-3-642-14162-1_7

[9] Janusz A. Brzozowski. 1964. Derivatives of regular expressions. *JACM* 11 (1964), 481–494.

[10] Guillaume Burel, Guillaume Bury, Raphaël Cauderlier, David Delahaye, Pierre Halmagrand, and Olivier Hermant. 2020. First-Order Automated Reasoning with Theories: When Deduction Modulo Theory Meets Practice. *Journal of Automated Reasoning* 64 (2020), 1001–1050.

[11] Alessandro Cimatti, Alberto Griggio, Enrico Magnago, Marco Roveri, and Stefano Tonetta. 2020. SMT-based satisfiability of first-order LTL with event freezing functions and metric operators. *Information and Computation* 272 (2020), 104502. https://doi.org/10.1016/j.ic.2019.104502 GandALF 2017.

[12] Jean-Michel Couvreur. 1999. On-the-Fly Verification of Linear Temporal Logic. In *FM'99 - Formal Methods, World Congress on Formal Methods in the Development of Computing Systems, Toulouse, France, September 20-24, 1999, Proceedings, Volume I (Lecture Notes in Computer Science, Vol. 1708)*, Jeannette M. Wing, Jim Woodcock, and Jim Davies (Eds.). Springer, 253–271. https://doi.org/10.1007/3-540-48119-2_16

[13] J.-M. Couvreur, A. Duret-Lutz, and D. Poitrenaud. 2005. On-the-fly emptiness checks for generalized Büchi automata. In *SPIN'05 (LNCS, Vol. 3639).* Springer, 143–158.

[14] Ron Cytron, Jeanne Ferrante, Barry K. Rosen, Mark N. Wegman, and F. Kenneth Zadeck. 1989. An Efficient Method of Computing Static Single Assignment Form. In *Conference Record of the Sixteenth Annual ACM Symposium on Principles of Programming Languages, Austin, Texas, USA, January 11-13, 1989.* ACM Press, 25–35. https://doi.org/10.1145/75277.75280

[15] Z. R. Dang. 1973. On the complexity of a finite automaton corresponding to a generalized regular expression. *Dokl. Akad. Nauk SSSR.* (1973).

[16] Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *TACAS'08 (LNCS).* Springer, 337–340.

[17] Alan J. Demers, Mark D. Weiser, Barry Hayes, Hans-Juergen Boehm, Daniel G. Bobrow, and Scott Shenker. 1990. Combining Generational and Conservative Garbage Collection: Framework and Implementations. In *Conference Record of the Seventeenth Annual ACM Symposium on Principles of Programming Languages, San Francisco, California, USA, January 1990*, Frances E. Allen (Ed.). ACM Press, 261–269. https://doi.org/10.1145/96709.96735

[18] Alexandre Duret-Lutz. 2011. LTL Translation Improvements in Spot. In *Proceedings of the 5th International Workshop on Verification and Evaluation of Computer and Communication Systems (VECoS'11) (Electronic Workshops in Computing)*. British Computer Society, Tunis, Tunisia.

[19] E.A. Emerson and C.-L. Lei. 1985. Modalities for model checking: Branching time logic strikes back. In *POPL*. ACM, 84–96.

[20] E.A. Emerson and C.-L. Lei. 1985. Temporal model checking under generalized fairness constraints. In *18th Hawaii International Conference on System Sciences*. 277–288.

[21] B. Finkbeiner and S. Schewe. 2005. Uniform distributed synthesis. In *LICS*. 321–330.

[22] Paul Gastin and Denis Oddoux. 2001. Fast LTL to Büchi Automata Translation. In *Computer Aided Verification, 13th International Conference, CAV 2001, Paris, France, July 18-22, 2001, Proceedings (Lecture Notes in Computer Science, Vol. 2102)*, Gérard Berry, Hubert Comon, and Alain Finkel (Eds.). Springer, 53–65. https://doi.org/10.1007/3-540-44585-4_6

[23] Luca Geatti, Alessandro Gianola, and Nicola Gigante. 2022. Linear Temporal Logic Modulo Theories over Finite Traces (Extended Version). https://doi.org/10.48550/ARXIV.2204.13693 Extended version of paper at the 31st International Joint Conference on Artificial Intelligence (IJCAI-ECAI 2022).

[24] Klaus Havelund and Grigore Roşu. 2001. Monitoring Programs using Rewriting. In *International Conference on Automated Software Engineering (ASE'01)*. IEEE.

[25] Jieh Hsiang. 1985. Refutational theorem proving using term-rewriting systems. *Artificial Intelligence* 25, 3 (1985), 255–300.

[26] H. B. Hunt III. 1973. *The equivalence problem for regular expressions with intersections is not polynomial in tape.* TR 73-161. Department of Computer Science, Cornell University, Ithaca, New York.

[27] Dexter Kozen. 1977. Lower bounds for natural proof systems. In *18th Annual Symposium on Foundations of Computer Science (SFCS 1977)*. 254–266. https://doi.org/10.1109/SFCS.1977.16

[28] R. P. Kurshan. 1987. Complementing Deterministic Büchi Automata in Polynomial Time. *J. Comput. System Sci.* 35 (1987), 59–71.

[29] E. J. McCluskey Jr. 1956. Minimization of boolean functions. *Bell Syst. Technical Journal* 35 (Nov 1956), 1417–1444.

[30] S. Miyano and T. Hayashi. 1984. Alternating finite automata on ω-words. *Theoretical Computer Science* 32 (1984), 321–330.

[31] D.E. Muller, A. Saoudi, and P.E. Schupp. 1988. Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In *[1988] Proceedings. Third Annual Symposium on Logic in Computer Science*. 422–427. https://doi.org/10.1109/LICS.1988.5139

[32] D. E. Muller, A. Saoudi, and P. E. Schupp. 1986. Alternating automata, the weak monadic theory of the tree and its complexity. In *ICALP*.

[33] D. E. Muller and P. E. Schupp. 1987. Alternating automata on infinite trees. *Theoretical Computer Science* 54 (1987), 267–276.

[34] M.O. Rabin. 1970. Weakly definable relations and special automata. In *Proc. Symp. Math. Logic and Foundations of Set Theory*, Y. Bar-Hilel (Ed.). North Holland, 1–23.

[35] Tiark Rompf, Arvind K. Sujeeth, Nada Amin, Kevin J. Brown, Vojin Jovanovic, HyoukJoong Lee, Manohar Jonnalagedda, Kunle Olukotun, and Martin Odersky. 2013. Optimizing data structures in high-level programs: new directions for extensible compilers based on staging. In *The 40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '13, Rome, Italy - January 23 - 25, 2013*, Roberto Giacobazzi and Radhia Cousot (Eds.). ACM, 497–510. https://doi.org/10.1145/2429069.2429128

[36] Kristin Y. Rozier and Moshe Y. Vardi. 2011. A Multi-encoding Approach for LTL Symbolic Satisfiability Checking. In *FM'11*. 417–431. https://doi.org/10.1007/978-3-642-21437-0_31

[37] Olli Saarikivi. 2017. *Symbolic Methods for Transducers and Testing*. Doctoral thesis. Aalto University School of Science. http://urn.fi/URN:ISBN:978-952-60-7787-1

[38] Koushik Sen, Grigore Roşu, and Gul Agha. 2003. Generating Optimal Linear Temporal Logic Monitors by Coinduction. In *Advances in Computing Science – ASIAN 2003 (LNCS, Vol. 2896)*, Vijay A. Saraswat (Ed.). Springer, 260–275.

[39] Fabio Somenzi and Roderick Bloem. 2000. Efficient Büchi Automata from LTL Formulae. In *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings (Lecture Notes in Computer Science, Vol. 1855)*, E. Allen Emerson and A. Prasad Sistla (Eds.). Springer, 248–263. https://doi.org/10.1007/10722167_21

[40] Ludwig Staiger. 1997. ω-Languages. In *Handbook of formal languages*, Grzegorz Rozenberg and Arto Salomaa (Eds.). Vol. 3. Springer, Chapter 6, 339–387.

[41] Caleb Stanford, Margus Veanes, and Nikolaj S. Bjørner. 2021. Symbolic Boolean derivatives for efficiently solving extended regular expression constraints. In *PLDI '21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20-25, 2021*, Stephen N. Freund and Eran Yahav (Eds.). ACM, 620–635. https://doi.org/10.1145/3453483.3454066

[42] L. J. Stockmeyer and A. R. Meyer. 1973. Word Problems Requiring Exponential Time(Preliminary Report). In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing, STOC'73*. ACM, 1–9. https://doi.org/10.1145/800125.804029

[43] Yih-Kuen Tsay and Moshe Y. Vardi. 2021. From Linear Temporal Logics to Büchi Automata: The Early and Simple Principle. In *Model Checking, Synthesis, and Learning - Essays Dedicated to Bengt Jonsson on The Occasion of His 60th Birthday (Lecture Notes in Computer Science, Vol. 13030)*, Ernst-Rüdiger Olderog, Bernhard Steffen, and Wang Yi (Eds.). Springer, 8–40. https://doi.org/10.1007/978-3-030-91384-7_2

[44] Moshe Y. Vardi. 1994. Nontraditional applications of automata theory. In *Theoretical Aspects of Computer Software*, Masami Hagiya and John C. Mitchell (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 575–597.

[45] Moshe Y. Vardi. 1995. *Alternating automata and program verification*. Springer Berlin Heidelberg, Berlin, Heidelberg, 471–485. https://doi.org/10.1007/BFb0015261

[46] Moshe Y. Vardi. 1995. An Automata-Theoretic Approach to Linear Temporal Logic. In *Logics for Concurrency - Structure versus Automata (8th Banff Higher Order Workshop, Banff, Canada, August 27 - September 3, 1995, Proceedings) (Lecture Notes in Computer Science, Vol. 1043)*, Faron Moller and Graham M. Birtwistle (Eds.). Springer, 238–266. https://doi.org/10.1007/3-540-60915-6_6

[47] Margus Veanes, Olli Saarikivi, Tiki Wan, and Eric Xu. 2019. Symbolic Regex Matcher. In *TACAS'19 (LNCS)*. Springer.

[48] Klaus Wagner. 1979. On ω-Regular Sets. *Information and Control* 43 (1979), 123–177.

[49] Pierre Wolper. 1981. Temporal Logic Can Be More Expressive. In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*. IEEE Computer Society, 340–348. https://doi.org/10.1109/SFCS.1981.44

[50] Pierre Wolper. 1983. Temporal logic can be more expressive. *Information and Control* 56, 1–2 (1983), 72–99.

[51] P. Wolper. 1985. The tableau method for temporal logic: An overview. *Logique et Analyse* 110–111 (1985), 119–136.

[52] Martin De Wulf, Laurent Doyen, Nicolas Maquet, and Jean-François Raskin. 2008. Antichains: Alternative Algorithms for LTL Satisfiability and Model-Checking. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings (Lecture Notes in Computer Science, Vol. 4963)*, C. R. Ramakrishnan and Jakob Rehof (Eds.). Springer, 63–77. https://doi.org/10.1007/978-3-540-78800-3_6

A PROOFS

Proofs of all theorems and properties that have been omitted from the main body of the paper.

Proofs of Section 3

Equations (8,9) are well-known properties of LTL, we include a proof of (8) here for clarity.

Proof of (8). ($\Rightarrow$): Let $w \models \varphi \mathbf{U} \psi$. Fix $j \in \mathbb{N}$ such that (7) holds. If $j = 0$ then $w_{0..} = w \models \psi$ and we are done. If $j > 0$ then, for all $i < j$, $w_{i..} \models \varphi$ and $w_{j..} = (w_{1..})_{j-1..} \models \psi$. In partcular $w_{0..} = w \models \varphi$ and there exists $k = j - 1 \in \mathbb{N}$ such that $(w_{1..})_{k..} \models \psi$ and for all $i < k$, $(w_{1..})_{i..} \models \varphi$. So, by (7), $w_{1..} \models \varphi \mathbf{U} \psi$.
($\Leftarrow$): If $w \models \psi$ then $w \models \varphi \mathbf{U} \psi$ follows immediately from (7). If $w \models \varphi$ and $w_{1..} \models \varphi \mathbf{U} \psi$ then there exists $k \in \mathbb{N}$ such that, by (7), $(w_{1..})_{k..} \models \psi$ and for all $i < k$, $(w_{1..})_{i..} \models \varphi$. It follows that for $j = k + 1$, $w_{j..} \models \psi$ and for all $i < j$, $w_{i..} \models \varphi$. So $w \models \varphi \mathbf{U} \psi$ by (7). □

Proofs of Section 4

Let $|f|$ stand for the number of $\mathbb{TT}$ *constructors* in a transition term f. For any $q \in Q$, $|q| = 0$ and $|\overline{q}| = 0$. Otherwise $|\neg f| = 1 + |f|$ and $|f \wedge g| = |f \vee g| = 1 + |f| + |g|$. Finally, $|\mathbf{ite}(\alpha, f, g)| = 1 + |f| + |g|$.

Proof of Theorem 4.2. We prove the statements

(i) $\mathbf{T}(f) = \mathbf{T}(\mathit{NNF}(f))$
(ii) $\mathbf{T}(\overline{f}) = \complement(\mathbf{T}(f))$

by induction over $|f|$. It clearly follows from the definitions that $|\mathit{NNF}(f)| \leq |f|$ and $|\overline{f}| \leq |f|$ that allows us to use the IH accordingly.

Base case Q: From $\mathbf{L}(\mathit{NNF}(q)) = \mathbf{L}(q)$ it follows that

$$\mathbf{T}(q) = \mathbb{D} \cdot \mathbf{L}(q) = \mathbb{D} \cdot \mathbf{L}(\mathit{NNF}(q)) = \mathbf{T}(\mathit{NNF}(q)).$$

We also have that $\mathbf{L}(\overline{q}) = \mathbf{L}(\neg q) = \complement(\mathbf{L}(q))$. Hence, by Equation 4.1(c) ,

$$\mathbf{T}(\overline{q}) = \mathbb{D}\cdot\mathbf{L}(\overline{q}) = \mathbb{D}\cdot\complement(\mathbf{L}(q)) = \complement(\mathbb{D}\cdot\mathbf{L}(q)) = \complement(\mathbf{T}(q))$$

Induction case $\vee$: By using the definitions and the IH,

$$\mathbf{T}(f \vee g) = \mathbf{T}(f) \cup \mathbf{T}(g) \overset{\text{(IH)}}{=} \mathbf{T}(\mathit{NNF}(f)) \cup \mathbf{T}(\mathit{NNF}(g)) = \mathbf{T}(\mathit{NNF}(f) \vee \mathit{NNF}(g)) = \mathbf{T}(\mathit{NNF}(f \vee g)).$$

And we also get, by using the IH and de Morgan's laws, that

$$\mathbf{T}(\overline{f \vee g}) = \mathbf{T}(\overline{f} \wedge \overline{g}) = \mathbf{T}(\overline{f}) \cap \mathbf{T}(\overline{g}) \overset{\text{(IH)}}{=} \complement(\mathbf{T}(f)) \cap \complement(\mathbf{T}(g)) = \complement(\mathbf{T}(f) \cup \mathbf{T}(g)) = \complement(\mathbf{T}(f \vee g))\,.$$

Induction case $\wedge$: Analogous to the case of $\vee$.

Induction case $\neg$: For (i) we get, by using the definitions and the IH for (ii), because $|\overline{f}| \leq |f| < |\neg f|$,

$$\mathbf{T}(\mathit{NNF}(\neg f)) = \mathbf{T}(\overline{f}) \overset{\text{(IH)}}{=} \complement(\mathbf{T}(f)) = \mathbf{T}(\neg f)$$

For (ii) we get, by using the IH for (i), that

$$\mathbf{T}(\overline{\neg f}) = \mathbf{T}(\mathit{NNF}(f)) \overset{\text{(IH)}}{=} \mathbf{T}(f) = \complement\big(\complement(\mathbf{T}(f))\big) = \complement(\mathbf{T}(\neg f))\,.$$

Induction case $\boldsymbol{ite}(\alpha, g, h)$*:* By using the definitions and the IH, let $[\![\alpha]\!] = A$,

$$\begin{array}{rcl}
\mathbf{T}(\mathit{NNF}(\mathbf{ite}(\alpha,g,h))) & = & \mathbf{T}(\mathbf{ite}(\alpha, \mathit{NNF}(g), \mathit{NNF}(h))) \\
& = & ((A\cdot\mathbb{D}^{\omega}) \cap \mathbf{T}(\mathit{NNF}(g))) \cup ((\complement(A)\cdot\mathbb{D}^{\omega}) \cap \mathbf{T}(\mathit{NNF}(h))) \\
& \overset{\text{(IH)}}{=} & ((A\cdot\mathbb{D}^{\omega}) \cap \mathbf{T}(g)) \cup ((\complement(A)\cdot\mathbb{D}^{\omega}) \cap \mathbf{T}(h)) \\
& = & \mathbf{T}(\mathbf{ite}(\alpha,g,h))
\end{array}$$

Finally, by using the IH and 4.1(e) ,

$$\begin{array}{rcl}
\mathbf{T}(\overline{\mathbf{ite}(\alpha,g,h)}) & = & \mathbf{T}(\mathbf{ite}(\alpha,\overline{g},\overline{h})) \\
& = & (A\cdot\mathbb{D}^{\omega} \cap \mathbf{T}(\overline{g})) \cup (\complement(A)\cdot\mathbb{D}^{\omega} \cap \mathbf{T}(\overline{h})) \\
& \overset{\text{(IH)}}{=} & (A\cdot\mathbb{D}^{\omega} \cap \complement(\mathbf{T}(g))) \cup (\complement(A)\cdot\mathbb{D}^{\omega} \cap \complement(\mathbf{T}(h))) \\
& \overset{\text{(4.1(e))}}{=} & \complement\big((A\cdot\mathbb{D}^{\omega} \cap \mathbf{T}(g)) \cup (\complement(A)\cdot\mathbb{D}^{\omega} \cap \mathbf{T}(h))\big) \\
& = & \complement(\mathbf{T}(\mathbf{ite}(\alpha,g,h)))
\end{array}$$

The Theorem follows by the induction principle. ◻

Proofs of Section 6

Proof of Theorem 6.1. We prove the following stement for all $a \in \mathbb{D}$ and by induction over $\phi \in LTL_{\mathcal{A}}$:

$$\mathbf{D}_a(\mathbf{L}(\phi)) = \mathbf{L}(\delta(\phi)(a)).$$

We need only to consider the following core constructs. The first base case is:

$$\mathbf{D}_a(\mathbf{L}(\alpha)) \overset{(10)}{=} (\text{if } a \in [\![\alpha]\!] \text{ then } \mathbb{D}^{\omega} \text{ else } \emptyset) \overset{(21,20)}{=} \mathbf{L}(\mathbf{ite}(\alpha,\top,\bot)(a)) \overset{(25)}{=} \mathbf{L}(\delta(\alpha)(a))$$

The second base case is for $\mathbf{X}$ because of the simple nature of the derivation rule:

$$\mathbf{D}_a(\mathbf{L}(\mathbf{X}\varphi)) \overset{(14)}{=} \mathbf{D}_a(\mathbb{D}\cdot\mathbf{L}(\varphi)) = \mathbf{L}(\varphi) \overset{(20)}{=} \mathbf{L}(\varphi(a)) \overset{(29)}{=} \mathbf{L}(\delta(\mathbf{X}\varphi)(a))$$

We have the following induction cases. For $\phi = \varphi \vee \psi$:

$$\mathbf{D}_a(\mathbf{L}(\phi)) \overset{(11)}{=} \mathbf{D}_a(\mathbf{L}(\varphi)) \cup \mathbf{D}_a(\mathbf{L}(\psi)) \overset{\text{(IH)}}{=} \mathbf{L}(\delta(\varphi)(a)) \cup \mathbf{L}(\delta(\psi)(a)) \overset{(12)}{=} \mathbf{L}(\delta(\varphi)(a) \vee \delta(\psi)(a)) \overset{(23,27)}{=} \mathbf{L}(\delta(\phi)(a))$$

For $\phi = \varphi \wedge \psi$: Analogous to the case of $\vee$.
For $\phi = \neg\varphi$:

$$\mathbf{D}_a(\mathbf{L}(\phi)) \overset{(13)}{=} \mathbf{D}_a(\complement(\mathbf{L}(\varphi))) \overset{(19)}{=} \complement(\mathbf{D}_a(\mathbf{L}(\varphi))) \overset{\text{(IH)}}{=} \complement(\mathbf{L}(\delta(\varphi)(a))) \overset{(13)}{=} \mathbf{L}(\neg(\delta(\varphi)(a))) \overset{(24)}{=} \mathbf{L}((\neg\delta(\varphi))(a)) \overset{(28)}{=} \mathbf{L}(\delta(\phi)(a))$$

For $\phi = \varphi \mathbf{U} \psi$:

$$\begin{aligned}
\mathbf{D}_a(\mathbf{L}(\phi)) &\overset{(15)}{=} \mathbf{D}_a(\mathbf{L}(\psi) \cup (\mathbf{L}(\varphi) \cap (\mathbb{D} \cdot \mathbf{L}(\phi)))) \\
&= \mathbf{D}_a(\mathbf{L}(\psi)) \cup (\mathbf{D}_a(\mathbf{L}(\varphi)) \cap \mathbf{D}_a(\mathbb{D} \cdot \mathbf{L}(\phi))) \\
&= \mathbf{D}_a(\mathbf{L}(\psi)) \cup (\mathbf{D}_a(\mathbf{L}(\varphi)) \cap \mathbf{L}(\phi)) \\
&\overset{(\mathrm{IH})}{=} \mathbf{L}(\delta(\psi)(a)) \cup (\mathbf{L}(\delta(\varphi)(a)) \cap \mathbf{L}(\phi)) \\
&\overset{(11,12)}{=} \mathbf{L}(\delta(\psi)(a) \vee (\delta(\varphi)(a) \wedge \phi)) \\
&\overset{(20)}{=} \mathbf{L}(\delta(\psi)(a) \vee (\delta(\varphi)(a) \wedge \phi(a))) \\
&\overset{(22,23)}{=} \mathbf{L}((\delta(\psi) \vee (\delta(\varphi) \wedge \phi))(a)) \\
&\overset{(30)}{=} \mathbf{L}(\delta(\phi)(a))
\end{aligned}$$

The statement follows by the induction principle. □

Proofs of Section 7

Recall Theorem 7.5: Nonemptiness of a nondeterministic Büchi Automata $\mathcal{B}$ modulo $\mathcal{A}$ is decidable in time $O(|\mathcal{B}|^2 + O^{\mathrm{sat}}_{\mathcal{A}}(|\mathcal{B}|))$.

Proof. Let $\mathcal{B} = (\mathcal{A}, Q, q^0, \rho, F)$. For each $q \in Q$ consider the transition term $\rho(q)$. Replace all the nested conditionals $\mathbf{ite}(\alpha, f, \mathbf{ite}(\beta, g, h))$ in $\rho(q)$ equivalently by $\mathbf{ite}(\alpha, f, \bot) \vee \mathbf{ite}({\sim}\alpha \mathbin{\&} \beta, g, \bot) \vee \mathbf{ite}({\sim}\alpha \mathbin{\&} {\sim}\beta, h, \bot)$. Analogously for $\mathbf{ite}(\alpha, \mathbf{ite}(\beta, g, h), f)$. Then remove all the disjuncts $\mathbf{ite}(\alpha, q, \bot)$ such that α is unsatisfiable in $\mathcal{A}$. The resulting transition function, say ρ' is such that $\rho'(q) \equiv \rho(q)$ for all $q \in Q$ and the size of ρ' is still linear in the size of ρ. Moreover, without loss of generality, for all $q \in Q$, $\rho'(q) = \bigvee_{i<n} \mathbf{ite}(\alpha_i, q_i, \bot)$ for some $n \leq |Q|$ and where each α_i is satisfiable so that all the target states are indeed reachable from q. Let $n = |\mathcal{B}|$.

It follows that the overall cost of deciding satisfiability of the predicates above in $\mathcal{A}$ is $O^{\mathrm{sat}}_{\mathcal{A}}(n)$ because there are *linearly* many branches in any nested conditional.

Now treat all $\alpha \in \Psi$ that occur in ρ' as a finite alphabet Σ. It follows that the size of Σ is also linear in n. Let $A = (\Sigma, Q, q^0, \sigma, F)$ be the classical nondeterministic Büchi automaton such that, for all $q \in Q$ and $\alpha \in \Sigma$, $\sigma(q, \alpha) = \{p \mid \mathbf{ite}(\alpha, p, \bot) \in \rho'(q)\}$. One can show that A is nonempty iff $\mathcal{B}$ is nonempty.

We know from [19, 20] that nonemptiness of A is decidable in time $O(|\Sigma||Q|)$ thus $O(n^2)$, that is, although linear in the size of Q, but where the size of the alphabet in [19, 20] is treated as a constant whereas here it depends linearly on the size of $\mathcal{B}$. In summary, nonemptiness of $\mathcal{B}$ is decidable in time $O(n^2 + O^{\mathrm{sat}}_{\mathcal{A}}(n))$. □

Proofs of Section 8

Recall Theorem 8.1: For all $\phi \in LTL_{\mathcal{A}}$ and $q \in Q_{\mathcal{B}^{\mathrm{EX}}_{\phi}}$: $\mathcal{L}_{\mathcal{B}^{\mathrm{EX}}_{\phi}}(q) = \mathbf{L}(q)$.

Proof. Let $\mathcal{B}^{\mathrm{EX}}_{\phi} = (\mathcal{A}, Q, \phi, \rho, F)$. Proof is by induction over the size $|q|$ of $q \in Q$. Observe that for $\alpha \in \Psi_{\mathcal{A}}$, $|\alpha| = |{\sim}\alpha| = 1$, i.e., all predicates of $\mathcal{A}$ are treated as atomic units. We abbreviate $\mathcal{L}_{\mathcal{B}^{\mathrm{EX}}_{\phi}}(\psi)$ by $\mathcal{L}(\psi)$.

Base case $q = \alpha \in \Psi_{\mathcal{A}}$: Observe that $\mathcal{L}(\top) = \mathbb{D}^{\omega}$ and $\mathcal{L}(\bot) = \emptyset$ because $\top \in F$ and $\bot \notin F$ and $\rho(\top) = \top$ and $\rho(\bot) = \bot$, so $\top$ is visited infinitely often in $\mathcal{B}^{\text{ex}}_{\phi}$. Then

$$
\begin{aligned}
w \in \mathcal{L}(q) &\overset{(\text{Thm } 7.2)}{\Leftrightarrow} w_{1..} \in \mathcal{L}(\rho(q)(w_0))\\
&\overset{(25)}{\Leftrightarrow} w_{1..} \in \mathcal{L}(\mathbf{ite(}\alpha, \top, \bot\mathbf{)}(w_0))\\
&\overset{(21)}{\Leftrightarrow} w_{1..} \in \mathcal{L}(\text{if } w_0 \in [[\alpha]] \text{ then } \top \text{ else } \bot) \Leftrightarrow w \in [[\alpha]]\cdot\mathbb{D}^{\omega} \Leftrightarrow w \in \mathbf{L}(q)
\end{aligned}
$$

Induction case $q = \mathbf{X}\psi$: Then

$$
w \in \mathcal{L}(q) \overset{(\text{Thm } 7.2)}{\Leftrightarrow} w_{1..} \in \mathcal{L}(\rho(q)(w_0)) \overset{(29)}{\Leftrightarrow} w_{1..} \in \mathcal{L}(\psi(w_0)) \overset{(20)}{\Leftrightarrow} w_{1..} \in \mathcal{L}(\psi) \overset{(\text{IH})}{\Leftrightarrow} w_{1..} \in \mathbf{L}(\psi) \overset{(5)}{\Leftrightarrow} w \in \mathbf{L}(q)
$$

Induction case $q = \varphi \wedge \psi$: Then $w \in \mathcal{L}(q) \overset{(36)}{\Leftrightarrow} w \in \mathcal{L}(\varphi) \cap \mathcal{L}(\psi) \overset{(\text{IH})}{\Leftrightarrow} w \in \mathbf{L}(\varphi) \cap \mathbf{L}(\psi) \overset{(2)}{\Leftrightarrow} w \in \mathbf{L}(q)$.

Induction case $q = \varphi \vee \psi$: Then $w \in \mathcal{L}(q) \overset{(35)}{\Leftrightarrow} w \in \mathcal{L}(\varphi) \cup \mathcal{L}(\psi) \overset{(\text{IH})}{\Leftrightarrow} w \in \mathbf{L}(\varphi) \cup \mathbf{L}(\psi) \overset{(3)}{\Leftrightarrow} w \in \mathbf{L}(q)$.

Induction case $q = \varphi \mathbf{U} \psi$: Then $\rho(q) = \hat{\delta}(q) = \hat{\delta}(\psi) \vee (\hat{\delta}(\varphi) \wedge q) = \rho(\psi) \vee (\rho(\varphi) \wedge q)$.

Direction $\subseteq$: Let $w \in \mathcal{L}(q)$ and let $\tau \in \mathbf{R}_w(q)$ be accepting; τ cannot visit q infinitely often in any branch of τ or else it would keep choosing the right disjunct $(\rho(\varphi) \wedge q)$, that would create one branch β labeled by q that would not be accepting because $q \notin F$. Therefore, τ can only follow $(\rho(\varphi) \wedge q)$ along β a finite number of steps j and then choose the left disjunct $\rho(\psi)$. Along that branch, simultaneously in each step i for $i < j$, $w_{i+1..} \in \mathcal{L}(\rho(\varphi)(w_i))$ must hold, implying, by Theorem 7.2, that $w_{i..} \in \mathcal{L}(\varphi)$. Finally, at step j, $w_{j+1..} \in \mathcal{L}(\rho(\psi)(w_j))$ must hold, implying, by Theorem 7.2, that $w_{j..} \in \mathcal{L}(\psi)$. By using the IH it follows that for all $i < j$, $w_{i..} \in \mathbf{L}(\varphi)$ and $w_{j..} \in \mathbf{L}(\psi)$. In other words, there exists $j \in \mathbb{N}$ such that $w_{j..} \models \psi$ and $\forall i < j : w_{i..} \models \varphi$. Therefore, by (7), $w \models \varphi \mathbf{U} \psi$, and so $w \in \mathbf{L}(q)$.

Direction $\supseteq$: Let $w \in \mathbf{L}(q)$. By reversing the steps in the above argument by using the IH and Theorem 7.2 one can construct an accepting run for w in $\mathbf{R}_w(q)$ showing that $w \in \mathcal{L}(q)$.

Induction case $q = \varphi \mathbf{R} \psi$: Then $\rho(q) = \hat{\delta}(q) = \hat{\delta}(\psi) \wedge (\hat{\delta}(\varphi) \vee q) = \rho(\psi) \wedge (\rho(\varphi) \vee q)$.

Direction $\subseteq$: Let $w \in \mathcal{L}(q)$ and let $\tau \in \mathbf{R}_w(q)$ be accepting. There are two cases.

(1) τ never chooses the left disjunct $\rho(\varphi)$ and thus has the label $q \in F$ in all the branches of the run occuring infinitely often. Then simultaneously, for all $j \in \mathbb{N}$, $w_{j+1..} \in \mathcal{L}(\rho(\psi)(w_j))$, which by Theorem 7.2, implies that $w_{j..} \in \mathcal{L}(\psi)$ and by the IH that $w_{j..} \in \mathbf{L}(\psi)$. We now have that $\forall j \in \mathbb{N} : w_{j..} \models \psi$.

(2) τ chooses the left disjunct $\rho(\varphi)$ at some step j. Then $\forall i < j : w_{i+1..} \in \mathcal{L}(\rho(\psi)(w_i))$, that by Theorem 7.2, implies that $w_{i..} \in \mathcal{L}(\psi)$, and by the IH, that $w_{i..} \in \mathbf{L}(\psi)$. We also have that, at step j

$$
\begin{aligned}
w_{j+1..} \in \mathcal{L}((\rho(\psi) \wedge \rho(\varphi))(w_j)) &\overset{(22)}{\Leftrightarrow} w_{j+1..} \in \mathcal{L}(\rho(\psi)(w_j) \wedge \rho(\varphi)(w_j))\\
&\overset{(36)}{\Leftrightarrow} w_{j+1..} \in \mathcal{L}(\rho(\psi)(w_j)) \cap \mathcal{L}(\rho(\varphi)(w_j))\\
&\overset{(\text{Thm } 7.2)}{\Leftrightarrow} w_{j..} \in \mathcal{L}(\psi) \cap \mathcal{L}(\varphi)\\
&\overset{(\text{IH})}{\Leftrightarrow} w_{j..} \in \mathbf{L}(\psi) \cap \mathbf{L}(\varphi)
\end{aligned}
$$

It follows that there exists j such that for all $i \leq j$, $w_{i..} \models \psi$ and $w_{j..} \models \varphi$.

Both cases imply, by using (6), that $w \models \varphi \mathbf{R} \psi$ and so $w \in \mathbf{L}(q)$.

Direction $\supseteq$: By reversing the steps in the above argument, using the IH and Theorem 7.2.

The statement follows by the induction principle. □

Proofs of Section 8.1

Recall Lemma 8.7: *If ξ is suspendable, $\diamond \in \{\wedge, \vee\}$, and $\blacklozenge \in \{\mathbf{U}, \mathbf{R}\}$ then* $\mathbf{L}((\varphi \blacklozenge \psi) \diamond \xi) = \mathbf{L}((\varphi \diamond \xi) \blacklozenge (\psi \diamond \xi))$.

Proof. The cases are proved separately. The proof also makes use of Boolean laws of distributivity, and the properties $\mathbb{D}{\cdot}X \cup \mathbb{D}{\cdot}Y = \mathbb{D}{\cdot}(X \cup Y)$ and $\mathbb{D}{\cdot}X \cap \mathbb{D}{\cdot}Y = \mathbb{D}{\cdot}(X \cap Y)$ for all $X, Y \subseteq \mathbb{D}^\omega$.

Case $\blacklozenge = \mathbf{U}$ *and* $\diamond = \wedge$. Let $\phi = (\varphi \mathbf{U} \psi) \wedge \xi$, we get, by using (15) and (11) and $\mathbf{L}(\xi) = \mathbb{D}{\cdot}\mathbf{L}(\xi)$, that

$$\begin{aligned}\mathbf{L}(\phi) &= (\mathbf{L}(\psi) \cup (\mathbf{L}(\varphi) \cap \mathbb{D}{\cdot}\mathbf{L}(\varphi \mathbf{U} \psi))) \cap \mathbf{L}(\xi) = (\mathbf{L}(\psi) \cap \mathbf{L}(\xi)) \cup (\mathbf{L}(\varphi) \cap \mathbf{L}(\xi) \cap \mathbb{D}{\cdot}\mathbf{L}(\xi) \cap \mathbb{D}{\cdot}\mathbf{L}(\varphi \mathbf{U} \psi))\\ &= \mathbf{L}(\psi \wedge \xi) \cup (\mathbf{L}(\varphi \wedge \xi) \cap \mathbb{D}{\cdot}(\mathbf{L}(\xi) \cap \mathbf{L}(\varphi \mathbf{U} \psi))) = \mathbf{L}(\psi \wedge \xi) \cup (\mathbf{L}(\varphi \wedge \xi) \cap \mathbb{D}{\cdot}\mathbf{L}(\phi))\end{aligned}$$

Case $\blacklozenge = \mathbf{R}$ *and* $\diamond = \wedge$. Let $\phi = (\varphi \mathbf{R} \psi) \wedge \xi$, we get, by (16), (11), and $\mathbf{L}(\xi) = \mathbb{D}{\cdot}\mathbf{L}(\xi)$, that

$$\begin{aligned}\mathbf{L}(\phi) &= (\mathbf{L}(\psi) \cap (\mathbf{L}(\varphi) \cup \mathbb{D}{\cdot}\mathbf{L}(\varphi \mathbf{R} \psi))) \cap \mathbf{L}(\xi) = \mathbf{L}(\psi) \cap \mathbf{L}(\xi) \cap ((\mathbf{L}(\varphi) \cap \mathbf{L}(\xi)) \cup (\mathbb{D}{\cdot}\mathbf{L}(\varphi \mathbf{R} \psi) \cap \mathbf{L}(\xi)))\\ &= \mathbf{L}(\psi \wedge \xi) \cap (\mathbf{L}(\varphi \wedge \xi) \cup (\mathbb{D}{\cdot}\mathbf{L}(\varphi \mathbf{R} \psi) \cap \mathbb{D}{\cdot}\mathbf{L}(\xi))) = \mathbf{L}(\psi \wedge \xi) \cap (\mathbf{L}(\varphi \wedge \xi) \cup \mathbb{D}{\cdot}\mathbf{L}(\phi))\end{aligned}$$

Case $\blacklozenge = \mathbf{U}$ *and* $\diamond = \vee$. Let $\phi = (\varphi \mathbf{U} \psi) \vee \xi$, we get, by using (15) and (12) and $\mathbf{L}(\xi) = \mathbb{D}{\cdot}\mathbf{L}(\xi)$, that

$$\begin{aligned}\mathbf{L}(\phi) &= (\mathbf{L}(\psi) \cup (\mathbf{L}(\varphi) \cap \mathbb{D}{\cdot}\mathbf{L}(\varphi \mathbf{U} \psi))) \cup \mathbf{L}(\xi) = (\mathbf{L}(\psi) \cup \mathbf{L}(\xi)) \cup ((\mathbf{L}(\varphi) \cup \mathbf{L}(\xi)) \cap (\mathbb{D}{\cdot}\mathbf{L}(\varphi \mathbf{U} \psi) \cup \mathbf{L}(\xi)))\\ &= \mathbf{L}(\psi \vee \xi) \cup (\mathbf{L}(\varphi \vee \xi) \cap (\mathbb{D}{\cdot}\mathbf{L}(\varphi \mathbf{U} \psi) \cup \mathbb{D}{\cdot}\mathbf{L}(\xi))) = \mathbf{L}(\psi \vee \xi) \cup (\mathbf{L}(\varphi \vee \xi) \cap \mathbb{D}{\cdot}\mathbf{L}(\phi))\end{aligned}$$

Case $\blacklozenge = \mathbf{R}$ *and* $\diamond = \vee$. Let $\phi = (\varphi \mathbf{R} \psi) \wedge \xi$, we get, by (16), (12), and $\mathbf{L}(\xi) = \mathbb{D}{\cdot}\mathbf{L}(\xi)$, that

$$\begin{aligned}\mathbf{L}(\phi) &= (\mathbf{L}(\psi) \cap (\mathbf{L}(\varphi) \cup \mathbb{D}{\cdot}\mathbf{L}(\varphi \mathbf{R} \psi))) \cup \mathbf{L}(\xi) = (\mathbf{L}(\psi) \cup \mathbf{L}(\xi)) \cap ((\mathbf{L}(\varphi) \cup \mathbf{L}(\xi)) \cup (\mathbb{D}{\cdot}\mathbf{L}(\varphi \mathbf{R} \psi) \cup \mathbb{D}{\cdot}\mathbf{L}(\xi)))\\ &= \mathbf{L}(\psi \vee \xi) \cap (\mathbf{L}(\varphi \vee \xi) \cup \mathbb{D}{\cdot}(\mathbf{L}(\varphi \mathbf{R} \psi) \cup \mathbf{L}(\xi))) = \mathbf{L}(\psi \vee \xi) \cap (\mathbf{L}(\varphi \vee \xi) \cup \mathbb{D}{\cdot}\mathbf{L}(\phi))\end{aligned}$$

Now consider $\phi' = (\varphi \diamond \xi) \mathbf{U} (\psi \diamond \xi)$ in the $\mathbf{U}$-cases above. Then by Equation (15) we get that

$$\mathbf{L}(\phi') = (\mathbf{L}(\psi \diamond \xi) \cup (\mathbf{L}(\varphi \diamond \xi) \cap \mathbb{D}{\cdot}\mathbf{L}(\phi'))).$$

Next consider $\phi' = (\varphi \diamond \xi) \mathbf{R} (\psi \diamond \xi)$ in the $\mathbf{R}$-cases above. Then by Equation (16) we get that

$$\mathbf{L}(\phi') = (\mathbf{L}(\psi \diamond \xi) \cap (\mathbf{L}(\varphi \diamond \xi) \cup \mathbb{D}{\cdot}\mathbf{L}(\phi'))).$$

The equations for ϕ and ϕ' are semantically identical in all cases, implying that $\mathbf{L}(\phi) = \mathbf{L}(\phi')$ in all cases. □

Recall Theorem 8.8: *If ϕ is a state then for any suspendable state ξ and $\diamond \in \{\wedge, \vee\}$, if $q = \phi \diamond \xi$ is included as a state of $\mathcal{B}$ with the transition term $\rho(q) = \delta(q)$ with the variant of δ then $\mathcal{L}_{\mathcal{B}}(q) = \mathbf{L}(q)$.*

Proof. By induction over ϕ, using Theorem 7.2, $\mathcal{L}\mathbf{L}$-invariance of the existing states, and Lemma 8.7.

Base case $\phi = \alpha \in \Psi$. Then $aw \in \mathcal{L}(\alpha \wedge \xi)$ iff $w \in \mathcal{L}(\rho(\alpha \wedge \xi)(a))$ iff $a \in [[\alpha]]$ and $aw \in \mathcal{L}(\xi)$ iff $a \in [[\alpha]]$ and $aw \in \mathbf{L}(\xi)$ iff $aw \in \mathbf{L}(\alpha \wedge \xi)$. The case for $\diamond = \vee$ is analogous.

Induction case $\phi = \mathbf{X}\psi$. Then $aw \in \mathcal{L}((\mathbf{X}\psi) \wedge \xi)$ iff $w \in \mathcal{L}(\rho((\mathbf{X}\psi) \wedge \xi)(a))$ iff $w \in \mathcal{L}((\psi \wedge \xi)(a))$ iff $w \in \mathcal{L}(\psi \wedge \xi)$ iff (by IH) $w \in \mathbf{L}(\psi \wedge \xi)$ iff (by ξ suspendable) $aw \in \mathbf{L}((\mathbf{X}\psi) \wedge \xi)$. The case for $\diamond = \vee$ is analogous.

Induction case $\phi = \varphi \vee \psi$. Then $w \in \mathcal{L}((\varphi \vee \psi) \wedge \xi)$ iff $w \in \mathcal{L}(\varphi \wedge \xi) \cup \mathcal{L}(\psi \wedge \xi)$ iff (by IH) $w \in \mathbf{L}(\varphi \wedge \xi) \cup \mathbf{L}(\psi \wedge \xi)$ iff $w \in \mathbf{L}((\varphi \vee \psi) \wedge \xi)$. The case for $\phi = \varphi \wedge \psi$ as well as the case for $\diamond = \vee$ are analogous.

Induction case $\phi = \varphi \ \mathbf{U} \ \psi$. Let $q = \phi \diamond \xi$. Then $\rho(q) = \rho(\psi \diamond \xi) \vee (\rho(\varphi \diamond \xi) \wedge q)$. We have that $\mathcal{L}(q) = \mathcal{L}((\varphi \diamond \xi) \ \mathbf{U} \ (\psi \diamond \xi))$ and by IH $\mathcal{L}(\psi \diamond \xi) = \mathbf{L}(\psi \diamond \xi)$ and $\mathcal{L}(\varphi \diamond \xi) = \mathbf{L}(\varphi \diamond \xi)$. It follows that $\mathcal{L}(q) = \mathbf{L}((\varphi \diamond \xi) \ \mathbf{U} \ (\psi \diamond \xi))$, but we also know that $\mathbf{L}(q) = \mathbf{L}((\varphi \diamond \xi) \ \mathbf{U} \ (\psi \diamond \xi))$ by Lemma 8.7. The state q is nonaccepting, essentially q is $(\varphi \diamond \xi) \ \mathbf{U} \ (\psi \diamond \xi)$.

Induction case $\phi = \varphi \ \mathbf{R} \ \psi$. Let $q = \phi \diamond \xi$. Then $\rho(q) = \rho(\psi \diamond \xi) \wedge (\rho(\varphi \diamond \xi) \vee q)$. Analogous to the case of $\mathbf{U}$ by using Lemma 8.7 but here q is *accepting*, essentially q is $(\varphi \diamond \xi) \ \mathbf{R} \ (\psi \diamond \xi)$.
The statement now follows by the induction principle. □