Regular Vacuity

Doron Bustan[2], Alon Flaisher[1], Orna Grumberg[1],
Orna Kupferman[3], and Moshe Y. Vardi[2]

[1] Technion Haifa
[2] Rice University
[3] Hebrew University †

Abstract. The application of model-checking tools to complex systems involves a non-trivial step of modelling the system by a finite-state model and a translation of the desired properties into a formal specification. While a positive answer of the model checker guarantees that the model satisfies the specification, correctness of the modelling is not checked. Vacuity detection is a successful approach for finding modelling errors that cause the satisfaction of the specification to be trivial. For example, the specification "every request is eventually followed by a grant" is satisfied vacuously in models in which requests are never sent. In general, a specification φ is satisfied vacuously in a model M if φ has a subformula ψ that does not affect the satisfaction of φ in M, where "does not affect" means we can replace ψ by a universally quantified proposition. Previous works focus on temporal logics such as LTL, CTL, and CTL*, and reduce vacuity detection to standard model checking.

A major feature of recent industrial property-specification languages is their regular layer, which includes regular expressions and formulas constructed from regular expressions. Our goal in this work is to extend vacuity detection to such a regular layer of linear-temporal logics. We focus here on RELTL, which is the extension of LTL with a regular layer. We define when a regular expression does not affect the satisfaction of an RELTL formula by means of universally quantified intervals. Thus, the transition to regular vacuity takes us from monadic quantification to dyadic quantification. We argue for the generality of our definition and show that regular-vacuity detection is decidable, but involves an exponential blow-up (in addition to the standard exponential blow-up for LTL model checking). This suggests that, in practice, one may need to work with weaker definitions of vacuity or restrict attention to specifications in which the usage of regular events is constrained. We discuss such weaker definitions, and show that their detection is not harder than standard model checking. We also show that, under certain polarity constraints, even general regular-vacuity detection can be reduced to standard model checking.

† For a full version with proofs, see `www.cs.rice.edu/~vardi/papers`.

1 Introduction

Model-checking tools are successfully used for checking whether systems have desired properties [9]. The application of model-checking tools to complex systems involves a nontrivial step of modelling the system by a finite-state mathematical model, and translation of the desired properties into a formal specification. When the model does not satisfy the specification, model-checking tools accompany a negative answer with a counterexample, which may point to a real error in the system [8]. It is often the case, however, that there is an error in the modelling of the system and/or in the formal specification. Such errors may not be detected when the answer of the model-checking tool is positive: while a positive answer does guarantee that the model satisfies the specification, the answer to the real question, namely, whether the system has the desired properties, may be different.

The realization of this unfortunate situation have led to the development of several *sanity checks* for formal verification. The goal of these checks is to detect errors in the modelling of the system and the properties. Sanity checks in industrial tools are typically simple, often ad hoc, tests, such as checking for enabling conditions that are never enabled [14]. A more systematic approach is based on *vacuity detection*. Intuitively, a specification is satisfied vacuously in a model if it is satisfied in some non-interesting way. For example, the LTL specification $\theta = \mathsf{globally}\,(req \rightarrow \mathsf{eventually}\,grant)$ ("every request is eventually followed by a grant") is satisfied vacuously in a model with no requests.

The first formal treatment of vacuity is described in [5]. Consider a model M satisfying a specification φ. A subformula ψ of φ *does not affect* (the satisfaction of) φ in M if M also satisfies all formulas obtained by modifying ψ. In the example above, the subformula $grant$ does not affect θ in a model with no requests. Now, M satisfies φ vacuously if φ has a subformula that does not affect φ in M. A general method for vacuity detection was presented in [13], who showed that when all the occurrences of ψ in φ are of a *pure polarity* (that is, they are either all under an even number of negations (positive polarity) , or all under an odd number of negations (negative polarity)), then ψ does not affect φ iff M satisfies the formula obtained from φ by the single extreme modification of ψ (to $\mathbf{true}$ in case ψ has a negative polarity, and to $\mathbf{false}$ otherwise). This observation reduces vacuity detection to model checking. The usefulness of vacuity analysis is also demonstrated via several case studies in [16].

As shown in [13], their method does not apply when subformulas of φ are of a *mixed polarity*. In practice, however, one often needs to cope with mixed polarity. For example, the subformula ψ has a mixed polarity in formulas of the (commonly seen) form $\mathsf{globally}\,(\psi \rightarrow \theta) \wedge \mathsf{eventually}\,\psi$. In fact, industrial-strength property-specification languages such as Sugar [4], ForSpec [2], and the recent standard PSL 1.01 [1] contain operators in which even a single occurrence of ψ may not have a pure polarity (e.g., ψ XOR θ or $\psi \leftrightarrow \theta$).

Once we allow subformulas of a mixed polarity, there is a need to re-examine the definition of when ψ does not affect φ in M. Indeed, it is only in the pure-polarity case that the various modifications of ψ may be restricted to the single extreme modification. Such a re-examination was done in [3], who considered vacuity detection for LTL specifications. While the modifications to ψ in [5] are *syntactic*; i.e., M has to satisfy all formulas $\varphi[\psi \leftarrow \psi']$, namely formulas obtained from φ by substituting

ψ by an LTL formula ψ', Armoni et al. argued that a right definition is one in which the modifications to ψ are *semantic*; i.e., M has to satisfy the formula $(\forall x)\varphi[\psi \leftarrow x]$, obtained by substituting ψ by a universally quantified proposition[1]. It is shown in [3] that, under such a semantic interpretation, vacuity detection of LTL formulas can still be reduced to LTL model checking.

As mentioned earlier, the work in [3] was motivated by the need to extend vacuity detection to recent industrial property-specification languages, which are significantly richer syntactically and semantically than LTL. A major feature of these languages, which does not exist in LTL, is a *regular layer*, which includes regular expressions and formulas constructed from regular expressions. For example, the ForSpec formula e seq θ, where e is a regular expression and θ is a formula, asserts that some e sequence is followed by θ, and the ForSpec formula e triggers θ, asserts that all e sequences are followed by θ. Our goal in this paper is to extend vacuity detection to such a regular layer of linear-temporal logics. Rather than treat the full complexity of industrial languages, we focus here on RELTL, which is the extension of LTL with a regular layer. Thus, we need to define, and then check, the notion of "does not affect," not only for subformulas but also for regular expressions. We refer to the latter as *regular vacuity*. As an example, consider the property $\varphi =$ globally $((req \cdot (\neg ack)^* \cdot ack)$ triggers $grant)$, which says that a grant is given exactly one cycle after the cycle in which a request is acknowledged. Note that if $(\neg ack)^* \cdot ack$ does not affect the satisfaction of φ in M (that is, replacing $(\neg ack)^* \cdot ack$ by any other sequence of events does not cause M to violate φ), we can learn that acknowledgments are actually ignored: grants are given, and stay on forever, immediately after a request. Such a behavior is not referred to in the specification, but can be detected by regular vacuity.

In order to understand our definition for regular vacuity, consider a formula φ over a set AP of atomic propositions. Let Σ be the set of Boolean functions over AP, and let e be a regular expression over Σ appearing in φ. The regular expression e induces a language – a set of finite words over Σ. For a word $w \in \Sigma^\omega$, the regular expression e induces a set of intervals [2]: these intervals define subwords of w that are members in the language of e. By saying that e does not affect φ in M, we want to capture the fact that we could modify e, replace it with any other regular expression, and still φ would be satisfied in M. As has been the case with propositional vacuity, there is no known algorithmic approach to handle such syntactic modifications in the presence of regular expressions of mixed polarity. Accordingly, as in [3], we follow a semantic approach to modifications of e, where "does not affect" is captured by means of universal quantification. Thus, in RELTL vacuity there are two types of elements we need to universally quantify to check vacuity. First, as in LTL, in order to check whether an RELTL subformula ψ, which is not a regular expression, affects the satisfaction of φ, we quantify universally over a proposition that replaces ψ. In addition, checking whether a regular expression e that appears in φ affects its satisfaction, we need to quantify universally over intervals. Thus, while LTL vacuity involved only *monadic* quantification (over the sets of points in which a subformula may hold), regular vacuity involves *dyadic* quan-

[1] A model M satisfies a formula $(\forall x)\varphi(x)$ if φ is satisfied in all computations π that differ from a computation of M only in the label of the proposition x. Note that different occurrences of a state in π may have different x labels.

tification (over intervals – sets of pairs of points, in which a regular expression may hold). In Section 3.2, we discuss two weaker alternative definitions: a restriction of the universally quantified intervals to intervals of the same duration as e (in case such a duration is well defined), and an approximation of the dyadic quantification over intervals by monadic quantification over the Boolean events referred to in the regular expressions. As discussed there, our definition is the most general one.

The transition from monadic to dyadic quantification is very challenging. Indeed, while monadic second-order logics are often decidable [7, 17], this is not the case for dyadic second-order logics. For example, while monadic second-order theory of one successor is decidable [7], this is not the case for the dyadic theory [6]. The main result of this work is that regular vacuity is decidable. We show that the automata-theoretic approach to LTL [20] can be extended to handle dyadic universal quantification. Unlike monadic universal quantification, which can be handled with no increase in computational complexity [3], the extension to dyadic quantification involves an exponential blow-up (in addition to the standard exponential blow-up of handling LTL [19]), resulting in an EXPSPACE upper bound, which should be contrasted with a PSPACE upper bound for RELTL model checking. Our NEXPTIME-hardness lower bound, while leaving a small gap with respect to the upper bound, shows that an exponential overhead on top of the complexity of RELTL model checking seems inevitable. The above results suggest that, in practice, one may need to restrict attention to specifications in which regular expressions are of pure polarity. We show that under this assumption, the techniques of [13] can be extended to regular vacuity, which can then be reduced to standard model checking. In addition, for specifications of mixed polarity, the two weaker definitions we suggest for regular vacuity can also be checked in PSPACE – like standard RELTL model checking.

2 RELTL

The linear temporal logic RELTL extends LTL with a regular layer. We consider LTL in a positive normal form, where formulas are constructed from atomic propositions and their negations by means of Boolean ($\wedge$ and $\vee$) and temporal (next , until , and its dual release) connectives. For details, see [15]. Let AP be a finite set of atomic propositions, and let $\mathcal{B}$ denote the set of all Boolean functions $b : 2^{AP} \rightarrow \{\mathbf{false}, \mathbf{true}\}$ (in practice, members of $\mathcal{B}$ are expressed by Boolean expressions over AP). Consider an infinite word $\pi = \pi_0, \pi_1, \ldots \in (2^{AP})^\omega$. For integers $j \geq i \geq 0$, and a language $L \subseteq \mathcal{B}^*$, we say that $\pi_i, \ldots, \pi_{j-1}$ *tightly satisfies* L, denoted $\pi, i, j \models L$, if there is a word $b_0 \cdot b_1 \cdots b_{j-1-i} \in L$ such that for all $0 \leq k < j - i$, we have that $b_k(\pi_{i+k}) = \mathbf{true}$. Note that when $i = j$, the interval $\pi_i, \ldots, \pi_{j-1}$ is empty, in which case $\pi, i, j \models L$ iff $\epsilon \in L$.

The logic RELTL contains two regular modalities: $(e \text{ seq } \varphi)$ and $(e \text{ triggers } \varphi)$, where e is a regular expression over the alphabet $\mathcal{B}$, and φ is an RELTL formula. Intuitively, $(e \text{ seq } \varphi)$ asserts that some interval satisfying e is followed by a suffix satisfying φ, whereas $(e \text{ triggers } \varphi)$ asserts that all intervals satisfying e are followed by a suffix satisfying φ. Note that the seq and triggers connectives are essentially

the "diamond" and "box" modalities of PDL [10]. Formally, let π be an infinite word over 2^{AP} then,[2]

- $\pi, i \models (e \ \mathsf{seq} \ \varphi)$ if for some $j \geq i$, we have $\pi, i, j \models\!\!\models L(e)$ and $\pi, j \models \varphi$.
- $\pi, i \models (e \ \mathsf{triggers} \ \varphi)$ if for all $j \geq i$ such that $\pi, i, j \models\!\!\models L(e)$, we have $\pi, j \models \varphi$.

In the automata-theoretic approach to model checking, we translate temporal logic formulas to automata [20]. A *nondeterministic generalized Büchi word automaton* (NBGW, for short) is a tuple $A = \langle \Sigma, S, \delta, S_0, \mathcal{F} \rangle$, where Σ is a finite alphabet, S is a finite set of states, $\delta : S \times \Sigma \to 2^S$ is a transition function, $S_0 \subseteq S$ is a set of initial states, and $\mathcal{F} \subseteq 2^S$ is a set of sets of accepting states. A run ρ of A is an infinite sequence of states in S that starts in a state in S_0 and obeys δ. Let $inf(\rho) \subseteq S$ denote the set of states that are visited infinitely often in ρ. Since the run is infinite and S is finite, it must be that $inf(\rho)$ is not empty. An NGBW A accepts an infinite word π if it has an infinite run ρ over π such that for every $F \in \mathcal{F}$, we have $inf(\rho) \cap F \neq \emptyset$. The full definition of NGBW is given in Appendix A. We now describe a translation of RELTL formulas to NGBW. The translation can be viewed as a special case of the translation of ETL to NGBW [20] (see also [11]), but we need it as a preparation for our handling of regular vacuity.

Theorem 1. *Given an RELTL formula φ over AP, we can construct an NGBW A_φ over the alphabet 2^{AP} such that $L(A_\varphi) = \{\pi | \pi, 0 \models \varphi\}$ and the size of A_φ is exponential in φ.*

Proof: The translation of φ goes via an intermediate formula ψ in the temporal logic ALTL. The syntax of ALTL is identical to the one of RELTL, only that regular expressions over $\mathcal{B}$ are replaced by nondeterministic finite word automata (NFW, for short) over 2^{AP}. The adjustment of the semantics is as expected: let $\pi = \pi_0, \pi_1, \ldots$ be an infinite path over 2^{AP}. For integers i and j with $0 \leq i \leq j$, and an NFW Z with alphabet 2^{AP}, we say that $\pi_i, \ldots, \pi_{j-1}$ *tightly satisfies* $L(Z)$, denoted $\pi, i, j, \models\!\!\models L(Z)$, if $\pi_i, \ldots, \pi_{j-1} \in L(Z)$. Then, the semantics of the seq and $\mathsf{triggers}$ modalities are as in RELTL, with $L(Z)$ replacing $L(e)$.

A regular expression e over the alphabet $\mathcal{B}$ can be polynomially translated to an equivalent NFW Z_e with a single initial state [12]. To complete the translation to ALTL, we need to adjust the constructed NFW to the alphabet 2^{AP}. Given the NFW $Z_e = \langle \mathcal{B}, Q, \Delta, q_0, W \rangle$, let $Z'_e = \langle 2^{AP}, Q, \Delta', q_0, W \rangle$, where for every $q, q' \in Q$, and $a \in 2^{AP}$, we have that $q' \in \Delta'(q, a)$ iff there exists $b \in \mathcal{B}$ such that $q' \in \Delta(q, b)$ and $b(a) = \mathbf{true}$. It is easy to see that for all π, i, and j, we have that $\pi, i, j \models\!\!\models L(e)$ iff $\pi, i, j \models\!\!\models L(Z'_e)$. Let ψ be the ALTL formula obtained from φ by replacing every regular expression e in φ by the NFW Z'_e. It follows that for every word π and $i \geq 0$, we have that $\pi, i \models \varphi$ iff $\pi, i \models \psi$.

It is left to show that ALTL formulas can be translated to NGBW. Let ψ be an ALTL formula. For a state $q \in Q$ of an NFW Z, we use Z^q to denote Z with initial state q. Using this notation, ALTL formulas of the form $(Z'_e \ \mathsf{seq} \ \varphi)$ and $(Z'_e \ \mathsf{triggers} \ \varphi)$ now become $(Z'^{q_0}_e \ \mathsf{seq} \ \varphi)$ and $(Z'^{q_0}_e \ \mathsf{triggers} \ \varphi)$. The closure of ψ, denoted $cl(\psi) =$

[2] In industrial specification languages such as ForSpec and PSL the semantics is slightly different. There, it is required that the last letter of the interval satisfying $L(e)$ overlaps the first letter of the suffix satisfying ψ. In Appendix E we describe a linear translation between these two semantics.

$\{\xi | \xi$ is a subformula of $\psi\} \cup \{(Z^{q'}$ seq $\xi) | (Z^q$ seq $\xi) \in cl(\psi)$ and q' is a state of $Z^q\} \cup \{(Z^{q'}$ triggers $\xi) | (Z^q$ triggers $\xi) \in cl(\psi)$ and q' is a state of $Z^q\}$. Let $seq(\psi)$ denote the set of seq formulas in $cl(\psi)$. A subset $C \subseteq cl(\psi)$ is *consistent* if the following hold: (1) if $p \in C$, then $\neg p \notin C$, (2) if $\varphi_1 \wedge \varphi_2 \in C$, then $\varphi_1 \in C$ and $\varphi_2 \in C$, and (3) if $\varphi_1 \vee \varphi_2 \in C$, then $\varphi_1 \in C$ or $\varphi_2 \in C$.

Given ψ, we define the NGBW $A_\psi = \langle 2^{AP}, S, \delta, S_0, \mathcal{F} \rangle$, where $S \subseteq 2^{cl(\psi)} \times 2^{seq(\psi)}$ is the set of all pairs (L_s, P_s) such that L_s is consistent, and $P_s \subseteq L_s \cap seq(\psi)$. Intuitively, when A_ψ reads the point i of π and is in state (L_s, P_s), it guesses that the suffix $\pi_i, \pi_{i+1}, \ldots$ of π satisfies all the formulas in L_s. In addition, as explained below, the set P_s keeps track of the seq formulas in L_s whose eventuality needs to be fulfilled. Accordingly, $S_0 = \{(L_s, \emptyset) \in S : \psi \in L_s\}$.

Before we describe the transition function δ, let us explain how subformulas of the form $(Z^q$ seq $\psi)$ and $(Z^q$ triggers $\psi)$ are handled. In both subformulas, something should happen after an interval that tightly satisfies Z^q is read. In order to "know" when an interval $\pi_i, \pi_{i+1}, \ldots \pi_{j-1}$ tightly satisfies Z^q, the NGBW A_ψ simulates a run of Z^q on it. The seq operator requires a single interval that tightly satisfies Z^q and is followed by a suffix satisfying ψ. Accordingly, A_ψ simulates a single run, which it chooses nondeterministically. For the triggers operator, the requirement is for every interval that tightly satisfies Z^q. Accordingly, here A_ψ simulates all possible runs of Z^q. Formally, $\delta : (S \times 2^{AP}) \rightarrow 2^S$ is defined as follows: $(L_t, P_t) \in \delta((L_s, P_s), a)$ iff the following conditions are satisfied:

- For all $p \in AP$, if $p \in L_s$ then $p \in a$, and if $\neg p \in L_s$ then $p \notin a$.
- If (next $\varphi_1) \in L_s$, then $\varphi_1 \in L_t$.
- If $(\varphi_1$ until $\varphi_2) \in L_s$, then either $\varphi_2 \in L_s$, or $\varphi_1 \in L_s$ and $(\varphi_1$ until $\varphi_2) \in L_t$.
- If $(\varphi_1$ release $\varphi_2) \in L_s$, then $\varphi_2 \in L_s$ and either $\varphi_1 \in L_s$, or $(\varphi_1$ release $\varphi_2) \in L_t$.

  Let $Z = \langle 2^{AP}, Q, \Delta, q_0, W \rangle$ be an NFW.
- If $(Z^q$ seq $\psi) \in L_s$, then (a) $q \in W$ and $\psi \in L_s$, or (b) $(Z^{q'}$ seq $\psi) \in L_t$ for some $q' \in \Delta(q, a)$.
- If $(Z^q$ triggers $\psi) \in L_s$, then (a) if $q \in W$, then $\psi \in L_s$, and (b) $(Z^{q'}$ triggers $\psi) \in L_t$ for all $q' \in \Delta(q, a)$.
- If $P_s = \emptyset$, then $P_t = L_t \cap seq(\varphi)$. Otherwise, for every $(Z^q$ seq $\psi) \in P_s$, we have that (a) $q \in W$ and $\psi \in L_s$, or (b) $(Z^{(q')}$ seq $\psi) \in P_t \cap L_t$ for some $q' \in \Delta(q, a)$.

Finally, the generalized Büchi acceptance condition is used to impose the fulfillment of until and seq eventualities. Thus, $\mathcal{F} = \{\Phi_1, \ldots, \Phi_m, \Phi_{seq}\}$, where for every $(\varphi_i$ until $\psi_i) \in cl(\varphi)$, we have a set $\Phi_i = \{(L_s, P_s) \in S | \psi_i \in L_s$ or $(\varphi_i$ until $\psi_i) \notin L_s\}$, and in addition we have the set $\Phi_{seq} = \{(L_s, P_s) \in S | P_s = \emptyset\}$. As in [20], we count on the fact that as long as a seq formula has not reached its eventuality, then some of its derivations appear in the successor state. In addition, whenever P_s is empty, we fill it with new seq formulas that need to be fulfilled. Therefore, the membership of Φ_{seq} in $\mathcal{F}$ guarantees that the eventualities of all seq formulas are fulfilled. The correctness of the construction is proved in Appendix B. □

The exponential translation of RELTL formulas to NGBW implies a PSPACE model-checking procedure for it [20]. A matching lower bound is immediate from LTL being a fragment of RELTL [19]. Hence the following theorem.

Theorem 2. *The model-checking problem for RELTL is PSPACE-complete.*

In Section 4, we elaborate on the construction described here in order to solve the regular vacuity problem.

3 Regular Vacuity

In this section we define when an RELTL formula is satisfied vacuously in a model.

3.1 A general definition

As discussed in Section 1, we follow the semantic approach to vacuity [3]. According to this approach, a subformula ψ of an RELTL formula φ *does not affect* φ in a model M if M satisfies $(\forall x)\varphi[\psi \leftarrow x]$. Thus, ψ is replaced by a universally quantified *propositional variable*. Unlike a subformula ψ, which defines a set of points in a path π (these that satisfy ψ), a regular expression e defines a set of intervals (that is, pairs of points) in π (these that tightly satisfy e). Accordingly, we are going to define "does not affect" for regular expressions by means of universally quantified *interval variables*. For that, we first define the temporal logic QRELTL, which extends RELTL by universal quantification over a single interval variable.

Recall that the regular expressions of RELTL formulas are defined with respect to the alphabet $\mathcal{B}$ of Boolean expressions over AP. Let y be the interval variable, and let φ be an RELTL formula whose regular expressions are defined with respect to the alphabet $\mathcal{B} \cup \{y\}$. Then $(\forall y)\varphi$ and $(\exists y)\varphi$ are QRELTL formulas. For example, $(\forall y)$ $\mathsf{globally}$ $[(y \ \mathsf{seq}\ \psi) \wedge (ab^* \ \mathsf{triggers}\ \neg\psi)]$ is a well-formed RELTL formula, while $\psi \vee [(\exists y)(y \ \mathsf{seq}\ \psi)]$ is not.

We now define QRELTL semantics. Let $I = \{(i,j) \mid i,j \in \mathbb{N}, j \geq i\}$ be a set of all (natural) intervals. An *interval set* is a set $\beta \subseteq I$. The interval variable y ranges over interval sets and is associated with β. Thus, $(i,j) \in \beta$ means that y is satisfied over an interval of length $j - i$ that starts at i. For a universally (existentially) quantified formula, satisfaction is checked with respect to every (some) interval set β. We first define when a word $\hat{\pi} = \pi_i \dots \pi_{j-1}$ over 2^{AP} tightly satisfies, with respect to β, a language L over $\mathcal{B} \cup \{y\}$. Intuitively, it means we can partition $\hat{\pi}$ to sub-intervals that together correspond to a word w in L. Note that since some of the letters in w may be y, the sub-intervals may be of arbitrary (possibly 0) length, corresponding to intervals in β. Formally, we have the following.

Definition 1. *Consider a language $L \subseteq (\mathcal{B} \cup \{y\})^*$, an infinite path π over 2^{AP}, indices i and j with $i \leq j$, and an interval set $\beta \subseteq I$. We say that $\pi_i, \dots, \pi_{j-1}$ and β* tightly satisfies *L, denoted $\pi, i, j, \beta \models L$ iff there is $w \in L$ such that either $w = \epsilon$ and $i = j$, or $w = w_0, w_1, \dots, w_n$ and there is a sequence of integers $i = l_0 \leq l_1 \leq \dots \leq l_{n+1} = j$ such that for every $0 \leq k \leq n$, the following conditions hold:*

- *If $w_k \in \mathcal{B}$, then $w_k(\pi_{l_k}) = \mathbf{true}$ and $l_{k+1} = l_k + 1$.*
- *If $w_k = y$, then $(l_k, l_{k+1}) \in \beta$.*

For example, if $AP = \{p\}$, $\beta = \{(3,3),(3,4)\}$, and $\pi = \{\{p\}, \emptyset\}^\omega$, then $\pi, 2, 4, \beta \models \{p \cdot y\}$ since $p(\{p\}) = \mathbf{true}$ and $(3,4) \in \beta$. Also, $\pi, 2, 4, \beta \models \{p \cdot y \cdot \neg p\}$, since $p(\{p\}) = \mathbf{true}$, $(3,3) \in \beta$, and $\neg p(\emptyset) = \mathbf{true}$. Note that when the required w does

not contain y, the definition is independent of β and coincides with tight satisfaction for languages over $\mathcal{B}$.

The semantics of the RELTL subformulas of a QRELTL formula is defined inductively as in RELTL, only with respect to an interval set β. In particular, for the seq and triggers modalities, we have

- $\pi, i, \beta \models (e \text{ seq } \varphi)$ iff for some $j \geq i$, we have $\pi, i, j, \beta \models L(e)$ and $\pi, j, \beta \models \varphi$.
- $\pi, i, \beta \models (e \text{ triggers } \varphi)$ iff for all $j \geq i$ s.t. $\pi, i, j, \beta \models L(e)$ we have $\pi, j, \beta \models \varphi$.

In addition, for QRELTL formulas, we have

- $\pi, i \models (\forall y)\varphi$ iff for every interval set $\beta \subseteq I$, we have $\pi, i, \beta \models \varphi$.
- $\pi, i \models (\exists y)\varphi$ iff there exists an interval set $\beta \subseteq I$, such that $\pi, i, \beta \models \varphi$.

An infinite word π over 2^{AP} satisfies a QRELTL formula φ, denoted $\pi \models \varphi$, if $\pi, 0 \models \varphi$. A model M satisfies φ, denoted $M \models \varphi$, if all traces of M satisfy φ.

Definition 2. *Consider a model M. Let φ be an RELTL formula that is satisfied in M and let e be a regular expression appearing in φ. We say that e* does not affect *φ in M iff $M \models (\forall y)\varphi[e \leftarrow y]$. Otherwise, e affects φ in M. Finally, φ is regularly vacuous in M if there exists a regular expression e that does not affect φ.*

As an example for regular vacuity, consider the property $\varphi =$ globally $((req \cdot \mathbf{true} \cdot \mathbf{true})$ triggers $ack)$, which states that an ack is asserted exactly three cycles after a req. When φ is satisfied in a M, one might conclude that all requests are acknowledged, and with accurate timing. However, the property is also satisfied in a model M that keeps ack high at all times. Regular vacuity of φ with respect to $(req \cdot \mathbf{true} \cdot \mathbf{true})$ will be detected by showing that the QRELTL formula $(\forall y)\varphi[(req \cdot \mathbf{true} \cdot \mathbf{true}) \leftarrow y]$ is also satisfied in M. This can direct us to the erroneous behavior.

In the previous example we considered regular vacuity with respect to the entire regular expression. Sometimes, a vacuous pass can only be detected by checking regular vacuity with respect to sub-regular expression. Consider the property $\varphi =$ globally $((req \cdot (\neg ack)^* \cdot ack)$ triggers $grant)$, which states that when an ack is asserted sometime after req, then $grant$ is asserted one cycle later. Regular vacuity on the sub-regular expression $((\neg ack)^* \cdot ack)$ can detect that ack is actually ignored, and that $grant$ is asserted immediately after req and remains high. On the other hand, regular vacuity would not be detected on the regular expression $e = (req \cdot (\neg ack)^* \cdot ack)$, as it does affect φ. This is because φ does not hold if e is replaced by an interval $(0, j)$, in which req does not hold in model M.

3.2 Alternative definitions

In this section we describe two alternative definitions for "does not affect" and hence also for regular vacuity. We argue that the definitions are weaker, in the sense that a formula that is satisfied vacuously with respect to Definition 2, is satisfied vacuously also with respect to the alternative definitions, but not vice versa. On the other hand, as we discuss in Section 5, vacuous satisfaction with respect to the alternative definitions is computationally easier to detect.

Regular vacuity modulo duration Consider a regular expression e over $\mathcal{B}$. We say that e is of *duration* d, for $d \geq 0$, if all the words in $L(e)$ are of length d. For example, $a \cdot b \cdot c$ is of duration 3. We say that e is of a *fixed duration* if it is of duration d for some $d \geq 0$. Let $e = a \cdot b \cdot c$ and let $\varphi = e$ triggers ψ. The property φ states that if the computation starts with the Boolean events a, b, and c, then ψ should hold at time 3. Suppose now that in a model M, the formula ψ does not hold at times 0,1, and 2, and holds at later times. In this case, φ holds due to the duration of e, regardless of the Boolean events in e. According to Definition 2, e affects φ (e.g., if $\beta = \{(0,1)\}$). On the other hand, e does not affect φ if we restrict the interval variable y to intervals of length 3. Thus, e does not affect the truth of φ in M modulo its duration iff φ is still true when e is replaced by an arbitrary interval of the *same* duration (provided e is of a fixed duration). Formally, for a duration d, let $I_d = \{(i, i+d) : i \in \mathbb{N}\}$ be the set of all natural intervals of duration d. The logic *duration-QRELTL* is a variant of QRELTL in which the quantification of y is parametrized by a duration d, and y ranges over intervals of duration d. Thus, $\pi, i \models (\forall_d y)\varphi$ iff for every interval set $\beta \subseteq I_d$, we have $\pi, i, \beta \models \varphi$, and dually for $(\exists_d y)\varphi$.

Definition 3. *Consider a model M. Let φ be an RELTL formula that is satisfied in M and let e be a regular expression of duration d appearing in φ. We say that e* does not affect *φ in M* modulo duration *iff $M \models (\forall_d y)\varphi[e \leftarrow y]$. Finally, φ is regularly vacuous in M modulo duration if there exists a regular expression e of a fixed duration that does not affect φ modulo duration.*

We note that instead of requiring e to have a fixed duration, one can restrict attention to regular expressions of a finite set of durations (in which case e is replaced by intervals of the possible durations); in particular, regular expressions of a bounded duration (in which case e is replaced by intervals shorter than the bound). As we show in Section 5, vacuity detection for all these alternative definitions is similar.

Regular vacuity modulo expression structure Consider again the formula $\varphi = e$ triggers ψ, for $e = a \cdot b \cdot c$. The formula φ is equivalent to the LTL formula $\varphi' = a \rightarrow X(b \rightarrow X(c \rightarrow X\psi))$. If we check the vacuity of the satisfaction of φ' in a system M, we check, for each of the subformulas a, b, and c whether they affect the satisfaction of φ'. For that, [3] uses universal monadic quantification. In regular vacuity modulo expression structure we do something similar – instead of replacing the whole regular expression with a universally quantified dyadic variable, we replace each of the Boolean functions in $\mathcal{B}$ that appear in the expression by a universally quantified monadic variable (or, equivalently, by a dyadic variable ranging over intervals of duration 1). Thus, in our example, φ passes vacuously in the system M described above, as neither a, b, nor c affect its satisfaction. Formally, we have the following[3].

Definition 4. *Consider a model M. Let φ be an RELTL formula that is satisfied in M and let e be a regular expression appearing in φ. We say that e* does not affect *φ in M* modulo expression structure *iff for all $b \in \mathcal{B}$ that appear in e, we have that $M \models$*

[3] Note that Definition 4 follows the semantic approach of [3]. A syntactic approach, as the one taken in [5, 13], would result in a different definition, where Boolean functions are replaced by different Boolean functions.

$(\forall_1 y)\varphi[b \leftarrow y]$. *Finally, φ is regularly vacuous in M modulo expression structure if there exists a regular expression e that does not affect φ modulo expression structure.*

Note that since vacuity modulo duration/structure of expression replaces the universal quantification on all intervals by a universal quantification over a subset of them, Definitions 3 and 4 are weaker than Definition 2, in the sense that more regular expressions do not affect φ in M according to Definitions 3 and 4. Therefore, as we demonstrate with the formula $a \cdot b \cdot c$ triggers ψ above, more formulas are going to pass vacuously with respect to the alternative definitions. Thus, when a formula passes vacuously according to Definition 2, this points to a more serious problem in the system and/or its specification.

4 Algorithmic Aspects of Detecting Vacuity

In this section we study the complexity of the regular-vacuity problem. As discussed in Section 3, vacuity detection can be reduced to model checking of a QRELTL formula of the form $(\forall y)\varphi$. We describe an automata-based EXPSPACE solution to the latter problem, and conclude that regular vacuity is in EXPSPACE. Recall that we saw in Section 2 that RELTL model checking is in PSPACE. As shown in [3], vacuity detection for LTL is not harder than LTL model checking, and can be solved in PSPACE. We show that regular vacuity is NEXPTIME-hard. Thus, while the precise complexity of regular vacuity is open, the lower bound indicates that an exponential overhead on top of the complexity of RELTL model checking seems inevitable.

4.1 Upper bound

We describe a model-checking algorithm for QRELTL formulas of the form $(\forall y)\varphi$. Recall that in the automata-theoretic approach to LTL model checking, one constructs, given an LTL formula φ, an automaton $A_{\neg\varphi}$ that accepts exactly all paths that do not satisfy φ. Model checking is then reduced to the emptiness of the product of $A_{\neg\varphi}$ with the model M [20]. For a QRELTL formula $(\forall y)\varphi$, we need to construct an automaton $A_{(\exists y)\neg\varphi}$, which accepts all paths that do not satisfy $(\forall y)\varphi$. Since we considered RELTL formulas in a positive normal form, the construction of $\neg\varphi$ has to propagate the negation inward to φ's atomic propositions, using De-Morgan laws and dualities. In particular, $\neg(e \text{ seq } \varphi) = (e \text{ triggers } \neg\varphi)$ and $\neg(e \text{ triggers } \varphi) = (e \text{ seq } \neg\varphi)$. It is easy to see that the length of $\neg\varphi$ in positive normal form is linear in the length of φ.

Theorem 3. *Given an existential QRELTL formula $(\exists y)\varphi$ over AP, we can construct an NGBW A_φ over the alphabet 2^{AP} such that $L(A_\varphi) = \{\pi | \pi, 0 \models (\exists y)\varphi\}$, and the size of A_φ is doubly exponential in φ.*

Proof: The translation of $(\exists y)\varphi$ goes via an intermediate formula $(\exists y)\psi$ in the temporal logic QALTL. The syntax of QALTL is identical to the one of QRELTL, only that regular expressions over $\mathcal{B} \cup \{y\}$ are replaced by NFW over $2^{AP} \cup \{y\}$. The closure of QALTL formulas is defined similarly to the closure of ALTL formulas. The adjustment of the semantics is similar to the adjustment of RELTL to ALTL described in Section 2. In particular, the adjustment of Definition 1 to languages over the alphabet $2^{AP} \cup \{y\}$ replaces the condition "if $w_k \in \mathcal{B}$ then $w_k(\pi_{l_k}) = \mathbf{true}$ and $l_{k+1} = l_k + 1$" there by the condition "if $w_k \in 2^{AP}$, then $w_k = \pi_{l_k}$ and $l_{k+1} = l_k + 1$" here.

Given a QRELTL formula $(\exists y)\varphi$, its equivalent QALTL formula $(\exists y)\psi$ is obtained by replacing every regular expression e in $(\exists y)\varphi$ by Z'_e, where Z'_e is as defined in Section 2. Note that the alphabet of Z'_e is $2^{AP} \cup \{y\}$. It is easy to see that for all π, i, j, and β, we have that $\pi, i, j, \beta \models L(e)$ iff $\pi, i, j, \beta \models L(Z'_e)$. Thus, for every word π and $i \geq 0$, we have that $\pi, i \models (\exists y)\varphi$ iff $\pi, i \models (\exists y)\psi$.

The construction of the NGBW A_φ from $(\exists y)\psi$ is based on the construction presented in Section 2. As there, when A_φ reads π_i and is in state (L_s, P_s), it guesses that the suffix $\pi_i, \pi_{i+1} \ldots$ satisfies all the subformulas in L_s. Since, however, here A_φ needs to simulate NFWs with transitions labelled by the interval variable y, the construction here is more complicated. While a transition labelled by a letter in 2^{AP} corresponds to reading the current letter π_i, a transitions labelled by y corresponds to reading an interval $\pi_i, \ldots, \pi_{j-1}$ in β. Recall that the semantics of QALTL is such that $(\exists y)\psi$ is satisfied in π if there is an interval set $\beta \subseteq I$ for which π, β satisfies ψ. Note that triggers formulas are trivially satisfied for an empty β, whereas seq formulas require β to contain some intervals. Assume that A_φ is in point i of π, it simulates a transition labelled y in an NFW that corresponds to a seq formula in L_s, and it guesses that β contains some interval (i, j). Then, A_φ has to make sure that all the NFWs that correspond to triggers formulas in L_s and that have a transition labelled y, would complete this transition when point j is reached. For that, L_s has to be associated with a set of triggers formulas.

Formally, for a set $L_s \subseteq cl(\psi)$, we define $wait(L_s) = \{(Z^{q'} \text{ triggers } \xi) | (Z^q \text{ triggers } \xi) \in L_s \text{ and } q' \in \Delta(q, y)\}$. Intuitively, $wait(L_s)$ is the set of triggers formulas that are waiting for an interval in β to end. Once the interval ends, as would be enforced by a seq formula, the members of $wait(L_s)$ should hold. Let $seq(\psi)$ and $trig(\psi)$ be the sets of seq and triggers formulas in $cl(\psi)$, respectively. An *obligation* for ψ is a pair $o \in seq(\psi) \times 2^{trig(\psi)}$. Let $obl(\psi)$ be the set of all the obligations for ψ. Now, to formalize the intuition above, assume that A_φ is in point i and it simulates a transition labelled y in the NFW Z for some $(Z^q \text{ seq } \xi) \in L_s$. Then, A_φ creates the obligation $o = ((Z^q \text{ seq } \xi), wait(L_s))$ and propagates it until the end of the interval.

The NGBW $A_\varphi = \langle 2^{AP}, S, \delta, S_0, \mathcal{F} \rangle$, where the set of states S is the set of all pairs (L_s, P_s) such that L_s is a consistent set of formulas and obligations, and $P_s \subseteq L_s \cap (seq(\varphi) \cup obl(\varphi))$. Note that the size of A_φ is doubly exponential in φ. The set of initial states is $S_0 = \{(L_s, P_s) | \psi \in L_s, P_s = \emptyset\}$. The acceptance condition is used to impose the fulfillment of until and seq eventualities, and are similar to the construction is Section 2; thus $\mathcal{F} = \{\Phi_1, \ldots, \Phi_m, \Phi_{seq}\}$ where $\Phi_i = \{s \in S | (\varphi_1 \text{ until } \varphi_2), \varphi_2 \in L_s \text{ or } (\varphi_1 \text{ until } \varphi_2) \notin L_s\}$, and $\Phi_{seq} = \{s \in S | P_s = \emptyset\}$. We define the transition relation δ as the set of all triples $((L_s, P_s), a, (L_t, P_t))$ that satisfy the following conditions:

1. For all $p \in AP$, if $p \in L_s$ then $p \in a$.
2. For all $p \in AP$, if $\neg p \in L_s$ then $p \notin a$.
3. If $(\text{ next } \varphi_1) \in L_s$, then $\varphi_1 \in L_t$.
4. If $(\varphi_1 \text{ until } \varphi_2) \in L_s$, then either $\varphi_2 \in L_s$, or $\varphi_1 \in L_s$ and $(\varphi_1 \text{ until } \varphi_2) \in L_t$.
5. If $(\varphi_1 \text{ release } \varphi_2) \in L_s$, then $\varphi_2 \in L_s$ and either $\varphi_1 \in L_s$, or $(\varphi_1 \text{ release } \varphi_2) \in L_t$.
6. If $(Z^q \text{ seq } \xi) \in L_s$, then at least one of the following holds:
   (a) $q \in W$ and $\xi \in L_s$.
   (b) $(Z^{q'} \text{ seq } \xi) \in L_t$ for some $q' \in \Delta(q, a)$.

(c) $\Delta(q, y) \neq \emptyset$ and $o = ((Z^q \ \mathsf{seq}\ \xi), wait(L_s)) \in L_s$. In this case we say that there is a y-transition from $(Z^q \ \mathsf{seq}\ \xi)$ to o in L_s.

If conditions a or b hold, we say that $(Z^q \ \mathsf{seq}\ \xi)$ is *strong* in L_s w.r.t. $((L_s, P_s), a, (L_t, P_t))$.

7. If $(Z^q \ \mathsf{triggers}\ \xi) \in L_s$, then the following holds:
   (a) If $q \in W$, then $\xi \in L_s$.
   (b) $(Z^{q'} \ \mathsf{triggers}\ \xi) \in L_t$ for all $q' \in \Delta(q, a)$.
8. For every $(Z^q \ \mathsf{seq}\ \xi) \in P_s$, at least one of the following holds:
   (a) $q \in W$ and $\xi \in L_s$.
   (b) $(Z^{q'} \ \mathsf{seq}\ \xi) \in P_t \cap L_t$ for some $q' \in \Delta(q, a)$.
   (c) $\Delta(q, y) \neq \emptyset$ and $o = ((Z^q \ \mathsf{seq}\ \xi), wait(L_s)) \in P_s$. In this case we say that there is a y-transition from $(Z^q \ \mathsf{seq}\ \xi)$ to o in P_s.

   If conditions a or b hold, we say that $(Z^q \ \mathsf{seq}\ \xi)$ is strong in P_s w.r.t. $((L_s, P_s), a, (L_t, P_t))$.
9. If $o = ((Z^q \ \mathsf{seq}\ \xi), \Upsilon) \in L_s$ then at least one of the following holds:
   (a) For some $q' \in \Delta(q, y)$, we have that $(Z^{q'} \ \mathsf{seq}\ \xi) \in L_s$ and $\Upsilon \subseteq L_s$. In this case we say that there is a y-transition from o to $(Z^{q'} \ \mathsf{seq}\ \xi)$ in L_s.
   (b) $o \in L_t$.

   If condition b holds, we say that o is strong in L_s w.r.t. $((L_s, P_s), a, (L_t, P_t))$.
10. If $o = ((Z^q \ \mathsf{seq}\ \xi), \Upsilon) \in P_s$ then at least one of the following holds:
    (a) For some $q' \in \Delta(q, y)$, we have that $(Z^{q'} \ \mathsf{seq}\ \xi) \in P_s$ and $\Upsilon \subseteq L_s$. In this case we say that there is a y-transition from o to $(Z^{q'} \ \mathsf{seq}\ \xi)$ in P_s.
    (b) $o \in P_t$.

    If condition b holds, we say that o is strong in P_s w.r.t. $((L_s, P_s), a, (L_t, P_t))$.
11. If $P_s = \emptyset$, then $P_t = L_t \cap (seq(\varphi) \cup obl)$.
12. If $wait(L_s) \subseteq L_s$, then for every element in $L_s \cap (seq(\varphi) \cup obl(\varphi))$ there exists a path (possibly of length 0) of y transitions to a strong element w.r.t. $((L_s, P_s), a, (L_t, P_t))$.
13. If $wait(L_s) \subseteq L_s$, then for every element in $P_s \cap (seq(\varphi) \cup obl(\varphi))$ there exists a path (possibly of length 0) of y transitions to a strong element w.r.t. $((L_s, P_s), a, (L_t, P_t))$.

We now explain the role of conditions 12 and 13 of δ. As explained above, for every formula $(Z^q \ \mathsf{seq}\ \xi)$ that should hold at point i, the NGBW A_φ simulates a run of Z^q that should eventually accept an interval of π. Since Z^q has transitions labelled by y, it is possible for Z^q to loop forever in (L_i, P_i) (when $(i, i) \in \beta$). Conditions 12 and 13 force the run of Z^q to eventually reach an accepting state, and prevent such an infinite loop. The correctness of the construction is proved in Appendix C. □

Given a model M and the NGBW A_φ for $(\exists y)\varphi$, the emptiness of their intersection can be tested in time polynomial or in space polylogarithmic in the sizes of M and A_φ (note that M and A_φ can be generated on the fly) [20]. A path in the intersection of M and A_φ is a witness that e affects φ. It follows that the problem of deciding whether a regular expression e affects φ in M can be solved in EXPSPACE. Since the number of regular expressions appearing in φ is linear in the length of φ, we can conclude with the following upper bound to the regular-vacuity problem.

Theorem 4. *The regular-vacuity problem for RELTL can be solved in EXPSPACE.*

In Section 5, we analyze the complexity of regular vacuity more carefully and show that the computational bottle-neck is the length of regular expressions appearing in triggers formulas in φ. We also describe a fragment of RELTL for which regular vacuity can be solved in PSPACE.

4.2 Lower bound

In the *exponential bounded-tiling* problem we are given a fixed set T of tiles, two relations $H, V \subseteq T \times T$, two tiles $t_{init}, t_{fin} \in T$, and an integer n. The goal is to tile a $(2^n \times 2^n)$-square so that horizontal neighbors belong to H, vertical neighbors belong to V, the first tile in the first row is t_{init}, and the first tile in the last row is t_{fin}. Thus, formally, a legal tiling is a function $t : \{0, \ldots, 2^n - 1\}^2 \to T$ such that: (1) for all $0 \leq i \leq 2^n - 2$ and $0 \leq j \leq 2^n - 1$, we have that $H(t(i, j), t(i + 1, j))$, (2) for all $0 \leq i \leq 2^n - 1$ and $0 \leq j \leq 2^n - 2$, we have that $V(t(i, j), t(i, j + 1))$, (3) $t(0, 2^n - 1) = t_{init}$, and (4) $t(0, 2^n - 1) = t_{fin}$. The exponential bounded tiling problem is known to be NEXPTIME-hard [18].

Theorem 5. *The regular vacuity problem for RELTL is NEXPTIME-hard.*

Proof: We do a reduction from the exponential bounded tiling problem. Given a tiling system $\mathcal{T} = \langle T, H, V, n, t_{init}, t_{fin} \rangle$, we construct a model $M_{\mathcal{T}}$ of a fixed size and an RELTL formula φ of length $O(n)$ such that $\neg\varphi$ is not regularly vacuous in $M_{\mathcal{T}}$ iff there is a legal tiling t for $\mathcal{T}$.

The details of the reduction are described in Appendix D. We describe only the idea. The formula φ is defined over the fixed set AP of atomic propositions, some of which encode tiles in T. We refer to a word $\pi \in (2^{AP})^*$ as an attempt to encode a legal tiling t. We divide the word π to blocks of length $2n$. Every block corresponds to a single location $[i, j]$ in the $(2^n \times 2^n)$-square, with i being the column, and j being the row. The tile $t(i, j)$ is encoded in the first point in the block. Using an atomic proposition that acts as a counter, the block also encodes i and j. Proper increase of the counter is forced by a conjunct in φ.

Let $t_0 \ldots t_{2^n-1}, t'_0 \ldots t'_{2^n-1}$ be two successive rows of the tiling t. For each $0 \leq i \leq 2^n - 1$ we know, given t_i, the possible values for t_{i+1} (these for which $H(t_i, t_{i+1})$, in case $i < 2^n - 1$) and the possible values for t'_i (these for which $V(t_i, t'_i)$). Consistency with H and V gives us a necessary condition for a word to encode a legal tiling. In addition, the tiling should satisfy the edge conditions; it should start with t_{init} and has t_{fin} in position $[0, 2^n - 1]$. It is easy to define conjuncts in φ that guarantee the satisfaction of the conditions on horizontal neighbors and the edge conditions. The difficult part in the reduction is in guaranteeing that the condition for vertical neighbors, which are exponentially far, holds. This is where regular vacuity comes into the picture. They enable us to relate $t[i, j]$ with $t[i, j + 1]$, for all i and j.

Let e be a regular expression. The formula φ contains a conjunct ψ_1 that, using the triggers modality, says that whenever we are in a beginning of a block u corresponding to position $[i, j]$, for some i and j, then e is tightly satisfied only in intervals that end when the block u' that corresponds to position $[i, j + 1]$ starts. The formula φ also contains a conjunct ψ_2 that, using the seq modality, says that whenever a block u, not in the last row, starts, there is a block u' in π that starts when an interval satisfying e ends, and the blocks u and u' encode tiles that are related by V.

The formula φ contains a conjunct $\psi_1 \wedge \psi_2$, with $e = b$. With such a conjunct, the formula φ is unsatisfiable, and a system $M_{\mathcal{T}}$ that generates all the computations over AP satisfies $\neg\varphi$. Also, a legal tiling for $\mathcal{T}$ exists iff the substitution of e by the interval set containing all intervals of length $2n2^n$ makes φ satisfiable. Accordingly, $\neg\varphi$ is not regularly vacuous in $M_{\mathcal{T}}$ iff there is a legal tiling t for $\mathcal{T}$. □

5 Regular Vacuity in Practice

The results in Section 4 suggest that, in practice, one may need to work with weaker definitions of vacuity or restrict attention to specifications in which the usage of regular expressions is constrained. In this section we show that under certain polarity constraints, regular vacuity can be reduced to standard model checking. In addition we show that even without polarity constraints, detection of the weaker definitions of vacuity, presented in Section 3.2, is also not harder than standard model checking.

5.1 Specifications of pure polarity

Examining industrial examples shows that in practice the number of trigger formulas that share a regular expression with a seq formula is quite small. One of the few examples that use both describes a clock tick pattern and is expressed by the formula $tick_pattern = (e \text{ seq } \mathbf{true}) \wedge \text{ globally } (e \text{ triggers } (e \text{ seq } \mathbf{true}))$, where e defines the clock ratio, e.g. $e = clock_low \cdot clock_low \cdot clock_high \cdot clock_high$.

As shown in the previous section, the general case of regular vacuity adds an exponential blow-up on top of the complexity of RELTL model checking. A careful analysis of the state space of A_φ shows that with every set L_s of formulas, we associate obligations that are relevant to L_s. Thus, if L_s contains no seq formula with an NFW that reads a transition labelled y, then its obligation is empty. Otherwise, $wait(L_s)$ contains only trigger formulas that appear in L_s and whose NFWs read a transition labelled y. In particular, in the special case where seq and trigger subformulas do not share regular expressions, we have$|obl(\varphi)| = 0$. For this type of specifications, where all regular expressions have a *pure polarity*, regular vacuity is much easier. Rather than analyzing the structure of A_φ in this special case, we describe here a direct algorithm for its regular-vacuity problem.

We first define *pure polarity* for regular expression. As formulas in RELTL are in positive normal form, polarity of a regular expression e is not defined by number of negations, but rather by the operator applied to e. Formally, an occurrence of a regular expression e is of *positive polarity* in φ if it is on the left hand side of a seq modality, and of *negative polarity* if it is on the left hand side of a triggers modality. The polarity of a regular expression is defined by the polarity of its occurrences as follows. A regular expression e is of *positive polarity* if all occurrences of e in φ are of positive polarity, of *negative polarity* if all occurrences of e in φ are of negative polarity, of *pure polarity* if it is either of positive or negative polarity, and of *mixed polarity* if some occurrences of e in φ are of positive polarity and some are of negative polarity.

Definition 5. *Given a formula φ and a regular expression of pure polarity e, we denote by $\varphi[e \leftarrow \bot]$ the formula obtained from φ by replacing e by $\mathbf{true}^*$, if e is of negative polarity, and by $\mathbf{false}$ if e is of positive polarity.*

We now show that for e with pure polarity in φ, checking whether e effects φ, can be reduced to RELTL model checking:

Theorem 6. *Consider a model M, RELTL formula φ, and regular expression e of pure polarity. Then, $M \models (\forall y)\varphi[e \leftarrow y]$ iff $M \models \varphi[e \leftarrow \bot]$.*

Since the model-checking problem for RELTL can be solved in PSPACE, it follows that the regular-vacuity problem for the fragment of RELTL in which all regular expressions are of pure polarity is PSPACE-complete.

5.2 Weaker definitions of regular vacuity

In Section 3.2, we suggested two alternative definitions for regular vacuity. We now show that vacuity detection according to these definitions is in PSPACE – not harder than RELTL model checking.

We first show that the dyadic quantification in duration-QRELTL can be reduced to a monadic one. Intuitively, since the quantification in duration-QRELTL ranges over intervals of a fixed and known duration, it can be replaced by a quantification over the points where intervals start. Formally, we have the following:

Lemma 1. *Consider a system M, an RELTL formula φ, a regular expression e appearing in φ, and $d > 0$. Then, $M \models (\forall_d y)\varphi[e \leftarrow y]$ iff $M \models (\forall x)\varphi[e \leftarrow (x \cdot \mathbf{true}^{d-1})]$, where x is a monadic variable.*

Universal quantification of monadic variables does not make model checking harder: checking whether $M \models (\forall x)\varphi$ can be reduced to checking whether there is a computation of M that satisfies $(\exists x)\neg\varphi$. Thus, as detailed in [3], when we construct the intersection of M with the NGBW for $\neg\varphi$, the values for x can be guessed, and the algorithm coincides with the one for RELTL model checking. Since detection of vacuity modulo duration and modulo expression structure are both reduced to duration-QRELTL model checking, Theorem 2 implies the following.

Theorem 7. *The problem of detecting regular vacuity modulo duration or modulo expression structure is PSPACE-complete.*

We note that when the formula is of a pure polarity, no quantification is needed, and e may be replaced, in the case of vacuity modulo duration, by $\mathbf{false}$ or $\mathbf{true}^d$ according its polarity. Likewise, in the case of vacuity modulo expression structure, the Boolean formulas in e may be replaced by $\mathbf{false}$ or $\mathbf{true}$.

6 Concluding Remarks

We extended in this work vacuity detection to a regular layer of linear-temporal logics. We focused here on RELTL, which is the extension of LTL with a regular layer. We defined the notion of "does not affect," for regular expressions in terms of universal dyadic quantification. We showed that regular vacuity is decidable, but involves an exponential blow-up (in addition to the standard exponential blow-up for LTL model checking). We showed that under certain polarity constraints on regular expressions, regular vacuity can be reduced to standard model checking.

We suggested two alternative definitions for regular vacuity and showed that with respect to these definitions, even for formulas that do not satisfy the polarity constraints, vacuity detection can be reduced to standard model checking, which makes them of practical interest. The two definitions are weaker than our general definition, in the sense that a vacuous pass according to them may not be considered vacuous according to the general definition. It may seem that working with a more sensitive

definition would be an advantage, but experience with vacuity detection in industrial settings shows that flooding users with too many reports of vacuous passes may be counterproductive. Thus, it is difficult to make at this point definitive statements about the overall usability of the weaker definitions, as more industrial experience with them is needed.

References

1. Accellera - www.accellera.org.
2. R. Armoni, L. Fix, A. Flaisher, R. Gerth, B. Ginsburg, T. Kanza, A. Landver, S. Mador-Haim, E. Singerman, A. Tiemeyer, M. Vardi, and Y. Zbar. The ForSpec temporal logic: A new temporal property-specification logic. In *Proc. 8th Int'l Conf/ on Tools and Algorithms for the Construction and Analysis of Systems*, volume 2280 of *LNCS*, pages 296–211, 2002.
3. R. Armoni, L. Fix, A. Flaisher, O. Grumberg, N. Piterman, A. Tiemeyer, and M. Vardi. Enhanced vacuity detection in linear temporal logic. In *Computer Aided Verification, 15th Inte.l Conf., (CAV 2003)*, volume 2725 of *LNCS*, pages 368–380, 2003.
4. I. Beer, S. Ben-David, C. Eisner, D. Fisman, A. Gringauze, and Y. Rodeh. The temporal logic Sugar. In *Proc. 13th Inte'l Conf. on Computer Aided Verification*, volume 2102 of *LNCS*, pages 363–367, 2001.
5. I. Beer, S. Ben-David, C. Eisner, and Y. Rodeh. Efficient detection of vacuity in ACTL formulas. *Formal Methods in System Design*, 18(2):141–162, 2001.
6. E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer-Verlag, 1996.
7. J. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Internat. Congr. Logic, Method. and Philos. Sci. 1960*, pages 1–12, 1962.
8. E. Clarke, O. Grumberg, K. McMillan, and X. Zhao. Efficient generation of counterexamples and witnesses in symbolic model checking. In *Proc. 32nd Design Automation Conf.*, pages 427–432, 1995.
9. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
10. M. Fischer and R. Ladner. Propositional dynamic logic of regular programs. *J. of Computer and Systems Sciences*, 18:194–211, 1979.
11. J. Henriksen and P. Thiagarajan. Dynamic linear time temporal logic. *Annals of Pure and Applied Logic*, 96(1–3):187–207, 1999.
12. J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
13. O. Kupferman and M. Vardi. Vacuity detection in temporal model checking. *Software Tools for Technology Transfer*, 4(2):224–233, 2003.
14. R. Kurshan. *FormalCheck User's Manual*. Cadence Design, Inc., 1998.
15. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, 1992.
16. M. Purandare and F. Somenzi. Vacuum cleaning CTL formulae. In *Proc. 14th Conf. on Computer Aided Verification*, volume 2404 of *LNCS*, pages 485–499, 2002.
17. M. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.
18. M. Savelsberg and P. van emde Boas. Bounded tiling, an alternative to satisfiability. In *2nd Frege conference*, pages 354–363. Akademya Verlag, 1984.
19. A. Sistla and E. Clarke. The complexity of propositional linear temporal logic. *J. ACM*, 32:733–749, 1985.
20. M. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.

A Preliminaries

Definition 6. *A nondeterministic finite word automaton (NFW) is a tuple* $Z = \langle \Sigma, Q, \Delta, q_0, W \rangle$ *s.t.* Σ *is an alphabet,* Q *is a set of states,* $\Delta : (Q \times \Sigma) \rightarrow 2^Q$ *is a transition relation,* q_0 *is a single initial state and* $W \subseteq Q$ *is the set of accepting states.*

Let $\pi = \pi_0, \pi_1, \ldots$ be a finite/infinite word over Σ. For $i \in \mathbb{N}$, let $\pi^i = \pi_i, \pi_{i+1}, \ldots$ denote the suffix of π from its ith letter. A sequence $\xi = q_0, q_1, \ldots q_n$ in Q^ω is a *run* of Z over a finite word $\pi \in \Sigma^*$, if q_0 is the initial state, and for every $0 \leq i < n$, we have $q_{i+1} \in \Delta(q_i, \pi_i)$. A run ξ of Z is *accepting* if $q_n \in W$. An NFW Z accepts a word π if Z has an accepting run over π. We use $L(Z)$ to denote the set of words that are accepted by Z. For $q \in Q$, we denote by Z^q the automaton Z with a single initial state q.

Definition 7. *A* nondeterministic generalized Büchi word automaton *(NBGW) is* $\mathcal{A} = \langle \Sigma, S, S_0, \delta, \mathcal{F} \rangle$*, where* Σ *is a finite set of alphabet letters,* S *is a set of states,* $\delta : S \times \Sigma \to 2^S$ *is a transition function,* $S_0 \subseteq S$ *is a set of initial states, and* $\mathcal{F} \subseteq 2^S$ *is a set of sets of accepting states.*

A sequence $\rho = s_0, s_1, \ldots$ in S^ω is a *run* of $\mathcal{A}$ over an infinite word $\pi \in \Sigma^\omega$, if $s_0 \in S_0$ and for every $i > 0$, we have $s_{i+1} \in \delta(s_i, \pi_i)$. We use $inf(\rho)$ to denote the set of states that appear infinitely often in ρ. A run ρ of $\mathcal{A}$ is *accepting* if for every $F \in \mathcal{F}$, we have $inf(\rho) \cap F \neq \emptyset$. An NGBW $\mathcal{A}$ accepts a word π if $\mathcal{A}$ has an accepting run over π. We use $L(\mathcal{A})$ to denote the set of words that are accepted by $\mathcal{A}$. For $s \in S$, we denote by $\mathcal{A}^s$ the automaton $\mathcal{A}$ with a single initial state s.

Next, we define the temporal logic LTL over a set of atomic propositions AP in positive normal form. The syntax of LTL is as follows. An atom $p \in AP$ is a formula and so is $\neg p$. If φ_1 and φ_2 are LTL formulas, then so are $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\mathsf{next}\ \varphi_1$, $\varphi_1\ \mathsf{until}\ \varphi_2$, and $\varphi_1\ \mathsf{release}\ \varphi_2$. Each LTL formula φ induces a language $L(\varphi) \subseteq (2^{AP})^\omega$ of exactly all the infinite words that satisfy φ.

The semantics of LTL is defined with respect to an infinite word $\pi \in (2^{AP})^\omega$. We use $\pi, i \models \varphi$ to indicate that the word π^i satisfies the formula φ. The relation $\models$ is defined by induction on the structure of φ as follows.

- For $p \in AP$, we have $\pi, i \models p$ iff $p \in \pi_i$, and $\pi, i \models \neg p$ iff $p \notin \pi_i$.

Let $\varphi, \varphi_1, \varphi_2$ be formulas.

- $\pi, i \models \varphi_1 \wedge \varphi_2$ iff $\pi, i \models \varphi_1$ and $\pi, i \models \varphi_2$.
- $\pi, i \models \varphi_1 \vee \varphi_2$ iff $\pi, i \models \varphi_1$ or $\pi, i \models \varphi_2$.
- $\pi, i \models \mathsf{next}\ \varphi$ iff $\pi, i+1 \models \varphi$.
- $\pi, i \models \varphi_1\ \mathsf{until}\ \varphi_2$ iff there exists $k \geq i$ such that $\pi, k \models \varphi_2$ and for all $i \leq j < k$ we have $\pi, j \models \varphi_1$.
- $\pi, i \models \varphi_1\ \mathsf{release}\ \varphi_2$ iff either for some $j \geq i$ $\pi, j \models \varphi_1$ and for every $i \leq k \leq j$ we have $\pi, k \models \varphi_2$, or for every $j \geq i$, $\pi, j \models \varphi_2$.

We use the operator ($\mathsf{eventually}\ \varphi$) as a shorthand for ($\mathbf{true}\ \mathsf{until}\ \varphi$), and we use the operator ($\mathsf{globally}\ \varphi$) as a shorthand for ($\mathbf{false}\ \mathsf{release}\ \varphi$).

Finally we define *Regular expression* over an alphabet Σ. The syntax of regular expressions is as follows. A letter $a \in \Sigma$ is a regular expression. If e_1 and e_2 are regular expressions, then so are $e \cdot e$, $e + e$, and e^*. Each regular expression e induces a language $L(e) \subseteq \Sigma^*$ of exactly all the finite words that satisfy e. The semantics of regular expressions is defined as follows:

- For $a \in \Sigma$, $L(a)$ is the single word a.

For regular expressions e_1 and e_2.

- $L(e_1 \cdot e_2)$ consists of words formed by concatenating a word in $L(e_1)$ with a word in $L(e_2)$.
- $L(e_1 + e_2)$ is the union of $L(e_1)$ and $L(e_2)$.
- $L(e^*)$ consists of words formed by concatenating zero or more words from $L(e)$, and includes the empty word ϵ.

B The correctness of the construction for ALTL

Theorem 8. *Let φ be an ALTL formula and let A_φ be its automaton. Then, $L(A_\varphi) = L(\varphi)$.*

B.1 $L(\varphi) \subseteq L(A_\varphi)$

Definition 8. *Let π be an infinite word, i an index, and $(Z^q \ \mathsf{seq}\ \xi)$ an ALTL formula, s.t. $\pi, i \models (Z^q \ \mathsf{seq}\ \xi)$. We define the* minimum satisfying index *of $\pi, i, (Z^q \ \mathsf{seq}\ \xi)$ denoted $msi(\pi, i, (Z^q \ \mathsf{seq}\ \xi))$ as the minimal index $j \geq i$ such that $\pi, i, j \models Z^q$ and $\pi, j \models \xi$.*

The minimal satisfying index determines the first index where the seq formula could be released from its obligation.

Lemma 2. *Let π be in $L(\varphi)$, then $\pi \in L(A_\varphi)$.*

Proof: We construct a fair run $\rho = (L_0, P_0), (L_1, P_1), \ldots$ of A_φ on π. For every $i \geq 0$ we define L_i to be a subset of $cl(\varphi)$ s.t. a subformula φ' of $cl(\varphi)$ is in L_i iff $\pi, i, \models \varphi'$. We define P_i to be a subset of $seq(\varphi) \cap L_i$. The subsets P_i are inductively defined. For $i = 0$, $P_0 = \emptyset$. For $i > 0$ we distinguish between two cases:

1. If $P_{i-1} = \emptyset$, then $P_i = L_i \cap seq(\varphi)$.
2. Otherwise, P_i contains a formula $(Z^{q'} \ \mathsf{seq}\ \xi)$ iff it is in L_i and there exists a formula $(Z^q \ \mathsf{seq}\ \xi)$ in P_{i-1} s.t. $msi(\pi, i-1, (Z^q \ \mathsf{seq}\ \xi)) = msi(\pi, i, (Z^{q'} \ \mathsf{seq}\ \xi)) > i$ and q' is in $\Delta(q, \pi_{i-1})$.

We need to prove that ρ is a fair run of A_φ on π. Since $\pi, 0 \models \varphi$, we have that L_0 contains φ. The definition of ρ implies that $P_0 = \emptyset$, thus, (L_0, P_0) is an initial state. The following two propositions complete the proof of Lemma 2.

Proposition 1. *For every i, we have that (L_i, P_i) is in $\delta((L_{i-1}, P_{i-1}), \pi_i)$.*

Proof: We show that all the conditions of δ are satisfied:

1. For all $p \in AP$, if $p \in L_{i-1}$, then $p \in \pi_{i-1}$.
2. For all $p \in AP$, if $\neg p \in L_{i-1}$ then $p \notin \pi_{i-1}$
3. If $(\ \mathsf{next}\ \varphi_1) \in L_{i-1}$, then since $\pi, i-1 \models (\ \mathsf{next}\ \varphi_1)$ we have that $\pi, i \models \varphi_1$ and thus $\varphi_1 \in L_i$.
4. If $(\varphi_1 \ \mathsf{until}\ \varphi_2) \in L_{i-1}$ then either $\pi, i-1 \models \varphi_2$, in which case $\varphi_2 \in L_{i-1}$, or $\pi, i-1 \models \varphi_1$ and $\pi, i \models (\varphi_1 \ \mathsf{until}\ \varphi_2)$ in which case $\varphi_1 \in L_{i-1}$ and $(\varphi_1 \ \mathsf{until}\ \varphi_2) \in L_i$.
5. If $(\varphi_1 \ \mathsf{release}\ \varphi_2) \in L_{i-1}$ then $\pi, i-1 \models \varphi_2$ and thus $\varphi_2 \in L_{i-1}$ and either $\pi, i-1 \models \varphi_1$ in which case $\varphi_1 \in L_s$, or $\pi, i \models (\varphi_1 \ \mathsf{release}\ \varphi_2)$ in which case $(\varphi_1 \ \mathsf{release}\ \varphi_2) \in L_t$.

6. If $(Z^q \ \mathsf{seq}\ \xi) \in L_{i-1}$, then we distinguish between two cases:
   (a) If $\epsilon \in L(Z^q)$ and $\pi, i-1 \models \xi$, then $q \in W$ and ξ is in L_{i-1}.
   (b) Otherwise, $j = msi(\pi, i-1, Z^q \ \mathsf{seq}\ \xi) \geq i$. Since there exists an accepting run of Z^q on $\pi_{i-1}, \pi_i, \ldots, \pi_{j-1}$ and $\pi, j \models \xi$, for the second state q' of the run, we have that $q' \in \Delta(q, \pi_{i-1})$ and $\pi_i, \pi_{i+1}, \ldots, \pi_{j-1}$ is in $L(Z^{q'})$. This implies that $\pi, i \models (Z^{q'} \ \mathsf{seq}\ \xi)$ and that $msi(\pi, i, Z^{q'} \ \mathsf{seq}\ \xi) = j$. Thus, $(Z^{q'} \ \mathsf{seq}\ \xi)$ in L_i.
7. If $(Z^q \ \mathsf{triggers}\ \xi) \in L_{i-1}$ then the following hold:
   (a) If $q \in W$, then $\epsilon \in L(Z^q)$ thus $\pi, i-1 \models \xi$. This implies that ξ is in L_{i-1}.
   (b) For every $q' \in \Delta(q, \pi_{i-1})$ we have that for every $j \geq i$, if $\pi_i, \pi_{i+1}, \ldots \pi_{j-1}$ is in $L(Z^{q'})$, then $\pi_{i-1}, \pi_i, \ldots \pi_{j-1}$ is in $L(Z^q)$, and $\pi, j \models \xi$. Thus $\pi, i \models (Z^{q'} \ \mathsf{triggers}\ \xi)$. This implies that $(Z^{q'} \ \mathsf{triggers}\ \xi)$ is in L_i for all $q' \in \Delta(q, a)$.
8. If $P_{i-1} = \emptyset$, then $P_i = L_i \cap seq(\varphi)$. Otherwise, for every $(Z^q \ \mathsf{seq}\ \xi) \in P_{i-1}$, we distinguish between two cases
   (a) If $\epsilon \in L(Z^q)$ and $\pi, i-1 \models \xi$, then $q \in W$ and ξ is in L_{i-1}.
   (b) Otherwise, $msi(\pi, i-1, Z^q \ \mathsf{seq}\ \xi) \geq i$. Since $P_{i-1} \subseteq L_{i-1}$, the formula $(Z^q \ \mathsf{seq}\ \xi)$ is in L_{i-1}. By item 6b there exists a formula $Z^{q'} \ \mathsf{seq}\ \xi$ in L_i s.t. $q' \in \Delta(q, \pi_{i-1})$ and $msi(\pi, i, Z^{q'} \ \mathsf{seq}\ \xi) = msi(\pi, i-1, Z^q)$. This implies that $(Z^{q'} \ \mathsf{seq}\ \xi)$ is in P_i.

□

Proposition 2. *ρ is a fair run of A.*

Proof: First, we prove that for every $i \leq n$ we have that $inf(\rho) \cap \Phi_i \neq \emptyset$. We prove that for every $j \geq 0$ there exists $k \geq j$ s.t. $\rho_k \in \Phi_i$. Let $\varphi_1 \ \mathsf{until}\ \varphi_2$ be the until formula which corresponds to Φ_i, we distinguish between two cases:

1. If $\pi, j \not\models (\varphi_1 \ \mathsf{until}\ \varphi_2)$ then $(\varphi_1 \ \mathsf{until}\ \varphi_2)$ is not in L_j, thus $\rho_j \in \Phi_i$.
2. Otherwise, $\pi, j \models (\varphi_1 \ \mathsf{until}\ \varphi_2)$, thus there exists $k \geq j$ s.t. $\pi, k \models \varphi_2$. This implies that φ_2 is in L_k, thus, $\rho_k \in \Phi_i$.

Next, we prove that $inf(\rho) \cap \Phi_{seq} \neq \emptyset$. We prove that for every $j \geq 0$ there exists $k \geq j$ such that $\rho_k \in \Phi_{seq}$. We distinguish between two cases:

1. If $P_j = \emptyset$, then $\rho_j \in \Phi_{seq}$.
2. Otherwise, P_j contains some seq formulas. Let $i = max_{\varphi' \in P_j} msi(\pi, j, \varphi')$. First we prove by induction that for every $j \leq k < i$, we have $max_{\varphi' \in P_k} msi(\pi, k, \varphi') = i$.
   - The base case $k = j$ is trivial.
   - Assume that $max_{\varphi' \in P_{k-1}} msi(\pi, k-1, \varphi') = i$. Let $(Z^q \ \mathsf{seq}\ \xi)$ be a formula in P_{k-1} s.t. $msi(\pi, k-1, (Z^q \ \mathsf{seq}\ \xi)) = i$. Since $i > k$, there exists a run $q, q_1, q_2, \ldots q_{i-k-1}$ of length > 1 of Z^q on $\pi_{k-1}, \pi_k, \ldots \pi_{i-1}$, and $\pi, i \models \xi$. This, implies that the formula $(Z^{q_1} \ \mathsf{seq}\ \xi)$ is in L_k. Since $q, q_1, q_2, \ldots q_{i-k-1}$ is the shortest accepting run of Z^q on a prefix of π^{k-1}, we have $msi(\pi, k, (Z^{q_1} \ \mathsf{seq}\ \xi)) = msi(\pi, k-1, Z^q \ \mathsf{seq}\ \xi) = i$, and that $q_1 \in \Delta(q, \pi_{k-1})$, thus $(Z^{q_1} \ \mathsf{seq}\ \xi) \in P_k$. On the other hand, for every other formula $\varphi' \in P_k$ there exists a formula φ'' in P_{k-1} s.t. $msi(\pi, k, \varphi') = msi(\pi, k-1, \varphi'') \leq i$. This implies that $max_{\varphi' \in P_k} msi(\pi, k, \varphi') = i$.

   Thus, for every formula $(Z^q \ \mathsf{seq}\ \xi)$ in P_{i-1}, we have that $msi(\pi, i-1, (Z^q \ \mathsf{seq}\ \xi)) = i$. This implies that for every formula $(Z^q \ \mathsf{seq}\ \xi)$ in P_{i-1}, we have that $q \in W$ and $\pi, i-1 \models \xi$. This implies that $P_i = \emptyset$, thus $\rho_i \in \Phi_{seq}$. □

B.2 $L(A_\varphi) \subseteq L(\varphi)$

Lemma 3. *Let π be in $L(A_\varphi)$, then $\pi \in L(\varphi)$.*

Before we prove the lemma we present a few propositions.

Proposition 3. *Let $\rho = (L_0, P_0,), (L_1, P_1), \ldots$ be a run of A_φ on π. Let $i \geq 0$ be an index s.t. the formula $(\varphi_1 \,\mathsf{until}\, \varphi_2)$ is in L_i. Then either for every $j \geq i$, we have $\{(\varphi_1 \,\mathsf{until}\, \varphi_2), \varphi_1\} \subseteq L_j$, or there exists $k \geq i$ s.t. $\varphi_2 \in L_k$ and for every $i \leq j < k$, we have $\varphi_1 \in L_j$.*

Proof: We prove with induction on $k \geq i$ that either there exists an index $k' \leq k$ s.t. $\varphi_2 \in L'_k$ and for every $i \leq j < k'$, $\varphi_1 \in L_j$, or for every $i \leq j \leq k$, $\{(\varphi_1 \,\mathsf{until}\, \varphi_2), \varphi_1\} \subseteq L_j$.

- Base case: $k = i$ follows trivially from the definition of δ.
- Induction step: assume that the proposition holds for $k-1$ we distinguish between two cases:
  1. If there exists $k' \leq k-1$ s.t. $\varphi_2 \in L'_k$ and for every $i \leq j < k'$, $\varphi_1 \in L_j$, then the lemma holds trivially.
  2. Otherwise, for every $i \leq j \leq k-1$, $\{(\varphi_1 \,\mathsf{until}\, \varphi_2), \varphi_1\} \subseteq L_j$. Since $\varphi_2 \notin L_{k-1}$, the definition of δ implies that either $\varphi_2 \in L_k$ or $\{(\varphi_1 \,\mathsf{until}\, \varphi_2), \varphi_1\} \subseteq L_k$, in both cases the induction holds for k.

□

Proposition 4. *Let $\rho = (L_0, P_0,), (L_1, P_1), \ldots$ be a run of A_φ on π. Let $i \geq 0$ be an index s.t. the formula $(\varphi_1 \,\mathsf{release}\, \varphi_2)$ is in L_i. Then either for every $j \geq i$, we have $\{(\varphi_1 \,\mathsf{release}\, \varphi_2), \varphi_2\} \subseteq L_j$, or there exists $k \geq i$ s.t. $\varphi_1 \in L_k$ and for every $i \leq j \leq k$, we have $\varphi_2 \in L_j$.*

Proof: We prove with induction on $k \geq i$ that either there exists an index $k' \leq k$ s.t. $\varphi_1 \in L'_k$ and for every $i \leq j \leq k'$, $\varphi_1 \in L_j$, or for every $i \leq j \leq k$, $\{(\varphi_1 \,\mathsf{release}\, \varphi_2), \varphi_1\} \subseteq L_j$.

- Base case: $k = i$ follows trivially from the definition of δ.
- Induction step: assume that the proposition holds for $k-1$ we distinguish between two cases:
  1. If there exists $k' \leq k-1$ s.t. $\varphi_1 \in L'_k$ and for every $i \leq j \leq k'$, $\varphi_2 \in L_j$, then the lemma holds trivially.
  2. Otherwise, for every $i \leq j \leq k-1$, $\{(\varphi_1 \,\mathsf{release}\, \varphi_2), \varphi_2\} \subseteq L_j$. Since $\varphi_1 \notin L_{k-1}$, the definition of δ implies that $\{(\varphi_1 \,\mathsf{until}\, \varphi_2), \varphi_2\} \subseteq L_k$ thus, the induction holds for k.

□

Proposition 5. *Let $\rho = (L_0, P_0,), (L_1, P_1), \ldots$ be a run of A_φ on π. Let $i \geq 0$ be an index s.t. the formula $(Z^q \,\mathsf{seq}\, \xi)$ is in L_i. Then one of the followings holds:*

1. *There exists $k \geq i$ s.t. there exists an accepting run $q, q_1, q_2, \ldots q_{k-i}$ of Z^q over $\pi_i, \pi_{i+1}, \ldots, \pi_{k-1}$ s.t. for every $i \leq j \leq k$, we have $(Z^{q_{j-i}} \,\mathsf{seq}\, \xi) \in L_j$, and $\xi \in L_k$.*

2. *There exists an infinite run $q, q_1, q_2 \ldots$ of Z^q over $\pi_i, \pi_{i+1}, \ldots$ s.t. for every $j > i$, we have $(Z^{q_{j-i}}\ \mathsf{seq}\ \xi) \in L_j$, and either $q_{j-i} \notin W$ or $\xi \notin L_j$.*

Proof: We prove with induction on $k \leq i$ that one of the following conditions holds:

1. There exists an index $i \leq k' \leq k$ s.t. there exists an accepting run $q, q_1, q_2, \ldots q_{k'-i}$ of Z^q over $\pi_i, \pi_{i+1}, \ldots, \pi_{k'-1}$ s.t. for every $i \leq j \leq k'$, we have $(Z^{q_{j-i}}\ \mathsf{seq}\ \xi) \in L_j$, and $\xi \in L_{k'}$.
2. There exists a run $q, q_1, q_2 \ldots q_{k-i}$ of Z^q over $\pi_i, \pi_{i+1}, \ldots \pi_{k-1}$ s.t. for every $i \leq j \leq k$, we have $(Z^{q_{j-i}}\ \mathsf{seq}\ \xi) \in L_j$, and either $q_{j-i} \notin W$ or $\xi \notin L_j$.

– Base case: $k = i$ follows trivially from the definition of δ.
– Induction step: assume that the proposition holds for $k-1$ we distinguish between two cases:
  1. There exists an index $i \leq k' \leq k-1$ s.t. there exists an accepting run $q, q_1, q_2, \ldots q_{k'-i}$ of Z^q over $\pi_i, \pi_{i+1}, \ldots, \pi_{k'-1}$ s.t. for every $i \leq j \leq k'$, we have $(Z^{q_{j-i}}\ \mathsf{seq}\ \xi) \in L_j$, and $\xi \in L_{k'}$. Then the lemma holds trivially.
  2. Otherwise, there exists a run $q, q_1, q_2 \ldots q_{k-i-1}$ of Z^q over $\pi_i, \pi_{i+1}, \ldots \pi_{k-2}$ s.t. for every $i \leq j \leq k-1$, we have $(Z^{q_{j-i}}\ \mathsf{seq}\ \xi) \in L_j$, and either $q_{j-i} \notin W$ or $\xi \notin L_j$. If $q_{k-i} \in W$ and $\xi \in L_k$, then there exists an accepting run $q, q_1, q_2, \ldots q_{k-i}$ of Z^q over $\pi_i, \pi_{i+1}, \ldots, \pi_{k-1}$ and the lemma holds
  Otherwise, the definition of δ implies that L_k contains a formula $(Z^{q_{k-i}}\ \mathsf{seq}\ \xi)$ s.t. $q_{k-i} \in \Delta(q_{k-i-1}, \pi_{k-1})$, thus the lemma holds.

□

Proposition 6. *Let $\rho = (L_0, P_0,), (L_1, P_1), \ldots$ be a run of A_φ on π. Let $i \geq 0$ be an index s.t. the formula $(Z^q\ \mathsf{seq}\ \xi)$ is in P_i. Then one of the following holds:*

1. *There exists $k \geq i$ s.t. there exists an accepting run $q, q_1, q_2, \ldots q_{k-i}$ of Z^q over $\pi_i, \pi_{i+1}, \ldots, \pi_{k-1}$ s.t. for every $i \leq j \leq k$, we have $(Z^{q_{j-i}}\ \mathsf{seq}\ \xi) \in P_j$, and $\xi \in L_k$.*
2. *There exists an infinite run $q, q_1, q_2 \ldots$ of Z^q over $\pi_i, \pi_{i+1}, \ldots$ s.t. for every $j \geq i$, we have $(Z^{q_{j-i}}\ \mathsf{seq}\ \xi) \in P_j$, and either $(q_{j-i} \notin W$ or $\xi \notin L_j$.*

The proof of this proposition is similar to the proof of Proposition 5, and thus omitted.

Proposition 7. *Let $\rho = (L_0, P_0,), (L_1, P_1), \ldots$ be a run of A_φ on π. Let $i \geq 0$ be an index s.t. the formula $(Z^q\ \mathsf{triggers}\ \xi)$ is in L_i. Then for every $j \geq i$ and every run $q, q_1, q_2, \ldots q_{j-i}$ of $Z(q)$ on $\pi_i, \pi_{i+1}, \ldots, \pi_{j-1}$, we have that the formula $(Z^{q_{j-i}}\ \mathsf{triggers}\ \xi)$ is in L_j. Furthermore, if $q_{j-i} \in W$, then $\xi \in L_j$.*

Proof: We prove the proposition by induction on $j \geq i$.

– Base case: $j = i$ follows directly from the definition of δ.
– Assume that the proposition holds for j we prove it for $j+1$. Let $q, q_1, q_2, \ldots q_{j-i+1}$ of $Z(q)$ on $\pi_i, \pi_{i+1}, \ldots, \pi_j$. The induction hypothesis implies that the formula $(Z^{q_{j-i}}\ \mathsf{triggers}\ \xi)$ is in L_j. Since $q_{j-i+1} \in \Delta(q_{j-i}, \pi_j)$, δ implies that the formula $(Z^{q_{j-i+1}}\ \mathsf{triggers}\ \xi)$ is in L_{j+1}. If $q_{j-i+1} \in W$, then δ implies that $\xi \in L_{j+1}$.

□

We now prove Lemma 3. Let $\rho = (L_0, P_0,), (L_1, P_1), \ldots$ be a fair run of A_φ on π. We prove with induction over the structure of φ that for every $i \geq 0$ and every subformula φ' we have that φ' in L_i iff $\pi, i \models \varphi'$. Since for every initial state (L, P) of A_φ, we have $\varphi \in L$, we have that $\pi, 0 \models \varphi$.

- Base case: For p and $\neg p$ the definition of the automaton implies that the lemma holds.
- Induction step: Assume that the lemma holds for φ_1, φ_2, and ξ.
  - The consistency of the states of the automaton implies that the lemma holds for the formulas $(\varphi_1 \wedge \varphi_2)$ and $(\varphi_1 \vee \varphi_2)$.
  - Let ($\mathsf{next}\ \varphi_1$) be a formula in L_i. The definition of δ implies that $\varphi_1 \in L_{i+1}$. The induction hypothesis implies that $\pi, i+1 \models \varphi_1$, thus $\pi, i \models (\ \mathsf{next}\ \varphi_1)$.
  - Let $(\varphi_1\ \mathsf{until}\ \varphi_2)$ be a formula in L_i. Proposition 3 implies that either there exists $k \geq i$ s.t. $\varphi_2 \in L_k$ and for every $i \leq j < k$, we have $\varphi_1 \in L_j$ or for every $j \geq i$, we have $\{(\varphi_1\ \mathsf{until}\ \varphi_2), \varphi_1\} \subseteq L_j$.
    In the first case, the induction hypothesis implies that for all $i \leq j < k$, we have $\pi, j \models \varphi_1$ and that $\pi, k \models \varphi_2$. Thus $\pi, i \models (\varphi_1\ \mathsf{until}\ \varphi_2)$.
    As for the second case let Φ_l be the fairness set which corresponds to $(\varphi_1\ \mathsf{until}\ \varphi_2)$. Since ρ is fair, there exists $k \geq i$ such that $\rho_k \in \Phi_l$. Given that $(\varphi_1\ \mathsf{until}\ \varphi_2) \in L_k$, $\rho_k \in \Phi_l$ implies that $\varphi_2 \in L_k$. Thus, the induction hypothesis implies that for all $i \leq j < k$, we have $\pi, j \models \varphi_1$ and that $\pi, k \models \varphi_2$. Thus $\pi, i \models (\varphi_1\ \mathsf{until}\ \varphi_2)$.
  - Let $(\varphi_1\ \mathsf{release}\ \varphi_2)$ be a formula in L_i. Proposition 4 implies that either for every $j \geq i$, we have $\{(\varphi_1\ \mathsf{release}\ \varphi_2), \varphi_2\} \subseteq L_j$, or there exists $k \geq i$ s.t. $\varphi_1 \in L_k$ and for every $i \leq j \leq k$, we have $\varphi_2 \in L_j$.
    The induction assumption implies that either for all $j \geq i$, we have $\pi, j \models \varphi_2$, or there or there exists $k \geq i$ s.t. for all $i \leq j \leq k$, we have $\pi, j \models \varphi_2$ and $\pi, k \models \varphi_1$. Thus, $\pi, i \models (\varphi_1\ \mathsf{release}\ \varphi_2)$.
  - Let $(Z^q\ \mathsf{seq}\ \xi)$ be a formula in L_i. Proposition 5 implies that one of the following holds
    1. There exists $k \geq i$ s.t. there exists an accepting run $q, q_1, q_2, \ldots q_{k-i}$ of Z^q over $\pi_i, \pi_{i+1}, \ldots, \pi_{k-1}$ s.t. for every $i \leq j \leq k$, we have $(Z^{q_{j-i}}\ \mathsf{seq}\ \xi) \in L_j$, and $\xi \in L_k$. In this case the induction hypothesis implies that $\pi, k \models \xi$ thus $\pi, i \models (Z^q\ \mathsf{seq}\ \xi)$.
    2. There exists an infinite run $q, q_1, q_2 \ldots$ of Z^q over $\pi_i, \pi_{i+1}, \ldots$ s.t. for every $j \geq i$, we have $(Z^{q_{j-i}}\ \mathsf{seq}\ \xi) \in L_j$, and either $q_{j-i} \notin W$ or $\xi \notin L_j$.
       Since ρ is fair there exists $k \geq i$ s.t. $P_k = \emptyset$. This implies that $(Z^{q_{k+1-i}}\ \mathsf{seq}\ \xi) \in P_{k+1}$. By proposition 6, one of the following should hold:
       (a) There exists $k' \geq k+1$ s.t. there exists an accepting run $q_{k+1-i}, q_{k+2-i}, \ldots q_{k'-i}$ of $Z^{q_{k+1-i}}$ over $\pi_{k+1}, \pi_{k+2}, \ldots, \pi_{k'-1}$ s.t. for every $k+1 \leq j \leq k'$, we have $(Z^{q_{j-k-1}}\ \mathsf{seq}\ \xi) \in P_j$, and $\xi \in L_{k'}$. In this case the induction hypothesis implies that $\pi, k \models \xi$. Since the run $q, q_1, q_2, \ldots, q_{k-i}, q_{k+1-i}, \ldots q_{k'-i}$ is accepting, we have $\pi, i \models (Z^q\ \mathsf{seq}\ \xi)$.
       (b) There exists an infinite run $q_{k+1-i}, q_{k+2-i}, \ldots$ of Z^q over $\pi_{k+1}, \pi_{k+2}, \ldots$ s.t. for every $j > k+1$, we have $(Z^{q_{j-k-1}}\ \mathsf{seq}\ \xi) \in P_j$, and either $q_{j-k} \notin W$ or $\xi \notin L_j$. In this case P_j is empty only finitely many time in ρ, thus ρ is not fair, contradiction.

- Let $(Z^q \text{ triggers } \xi)$ be a formula in L_i. Proposition 7 implies that for every $j \geq i$, if Z^q has an accepting run over $\pi_i, \pi_{i+1}, \ldots \pi_{j-1}$, then $\xi \in L_j$. In this case the induction hypothesis implies that $\pi, j \models \xi$. This implies that $\pi, i \models (Z^q \text{ triggers } \xi)$.

□

C The Correctness of the Construction for QALTL

Theorem 9. $L(A_\varphi) = L((\exists y)\varphi)$.

C.1 First direction $L((\exists y\varphi)) \subseteq L(A_\varphi)$

We start by extending the definition of msi to obligations. We say that an obligation $o = ((Z^q \text{ seq } \xi), \Upsilon)$ *is possible* for π, i, β if there exists an index $j \geq i$ s.t. $\pi, j, \beta \models \Upsilon$ and for some $q' \in \Delta(q, y)$, we have $\pi, j, \beta \models (Z^{q'} \text{ seq } \xi)$.

Definition 9. *Let π be a word, i an index, β an interval set, and $(Z^q \text{ seq } \xi)$ a formula in $seq(\varphi)$ s.t. $\pi, i, \beta \models (Z^q \text{ seq } \xi)$. Then $msi(\pi, i, \beta, (Z^q \text{ seq } \xi)) = min(\{j | \pi, i, j, \beta \models\!\!\models L(Z^q) \wedge \pi, j, \beta \models \xi\})$. Let $o = ((Z^q \text{ seq } \xi), \Upsilon)$ be an obligation that is possible for π, i, β. We define $msi(\pi, i, \beta, o) = \{j | \exists q' \in \Delta(q, y) \exists k \geq i \text{ s.t. } \pi, k, \beta \models \Upsilon \wedge j = msi(\pi, k, (Z^{q'} \text{ seq } \xi))\})$.*

We now present a lemma, which defines the conditions for tight satisfaction of $L(Z^q)$ in terms of states of Z^q.

Lemma 4. *Let Z^q be NFW, let π be an infinite word, let $j \geq i \geq 0$, and let $\beta \subseteq I$ be an interval assignment. Then $\pi, i, j, \beta \models\!\!\models L(Z^q)$ iff at least one of the following holds:*

1. *$j = i$ and $q \in W$.*
2. *$j > i$ and there exists $q' \in \Delta(q, \pi_i)$ s.t. $\pi, i+1, j, \beta \models\!\!\models L(Z^{q'})$.*
3. *There exists some k, $i \leq k \leq j$ and a state $q' \in \Delta(q, y)$ s.t. $(i, k) \in \beta$ and $\pi, k, j, \beta \models\!\!\models L(Z^{q'})$.*

Proof: $\pi, i, j, \beta \models\!\!\models L(Z^q)$ iff there is $w \in L(Z^q)$, $w = x_0, x_1, \ldots, x_n$ and there is a sequence of integers $i_0, i_1, \ldots, i_n, i_{n+1}$, such that $i_0 = i$ and $i_{n+1} = j$. Moreover, for every $0 \leq k \leq n$ the following conditions hold:

- If $x_k \in 2^{AP}$ then $x_k = \pi_{i_k}$ and $i_{k+1} = i_k + 1$.
- If $x_k = y$ then $(i_k, i_{k+1}) \in \beta$.

We partition this condition into four cases:

1. The case where $w = \epsilon$. In this case $j = i$. Since $w = \epsilon \in L(Z^q)$, we have $q \in W$. Thus the first condition of the lemma holds.
2. The case where $|w| > 0$ and $x_0 \in 2^{AP}$. In this case $x_0 = \pi_i$ thus, for some $q' \in \Delta(q, \pi_i)$ we have that $x_1, x_2, \ldots x_n \in L(Z^{q'})$. Furthermore, for every $1 \leq k \leq n$ we have that the following conditions hold:
   - If $x_k \in 2^{AP}$ then $x_k = \pi_{i_k}$ and $i_{k+1} = i_k + 1$.
   - If $x_k = y$ then $(i_k, i_{k+1}) \in \beta$.

   Thus, $\pi, i+1, j, \beta \models\!\!\models L(Z^{q'})$ and the second condition holds.

3. The case where $|w| > 0$ and $x_0 = y$. For $k = i_1$, we have that $(i,k) \in \beta$, and for some $q' \in \Delta(q,y)$, we have $x_1, x_2, \ldots x_n \in L(Z^{q'})$. Furthermore, for the sequence $i_1, i_2, \ldots, i_{n+1}$ and for every $1 \leq l \leq n$ we have that the following conditions hold:
   - If $x_l \in 2^{AP}$ then $x_l = \pi_{i_l}$ and $i_{l+1} = i_l + 1$.
   - If $x_l = y$ then $(i_l, i_{l+1}) \in \beta$.

   Thus $\pi, k, j, \beta \models L(Z^{q'})$ and the third condition holds. □

Lemma 5. *Let π be in $L((\exists y)\varphi)$, then $\pi \in L(A_\varphi)$.*

Proof: We construct a fair run ρ of A_φ on π. Let β be an interval set such that $\pi, 0, \beta \models \varphi$. For every $i \geq 0$ we define $L_i = \{\varphi' | \pi, i, \beta \models \varphi'\} \cup \{o | o \text{ is possible for } \pi, i, \beta\}$. We define P_i to be a subset of L_i. The subsets P_i are inductively defined. For $i = 0$, $P_0 = \emptyset$. For $i > 0$ we distinguish between two cases:

1. If $P_{i-1} = \emptyset$, then $P_i = L_i \cap (seq(\varphi) \cup O(\varphi))$.
2. Otherwise, P_i contains a formula $(Z^{q'} \text{ seq } \xi)$ iff it is in L_i and $msi(\pi, i, \beta, (Z^{q'} \text{ seq } \xi)) \leq max\{msi(\pi, i-1, \beta, x) | x \in P_{i-1}\}$. P_i contains an obligation formula $o = ((Z^q \text{ seq } \xi), \Upsilon)$ iff it is in L_i and $msi(\pi, i, \beta, o) \leq max\{msi(\pi, i-1, \beta, x) | x \in P_{i-1}\}$.

The following two propositions completes the proof of Lemma 5.

Proposition 8. *For every i, we have that (L_i, P_i) is in $\delta((L_{i-1}, P_{i-1}), \pi_{i-1})$.*

Proof: We need to show that all the conditions for the transition relation are fulfilled. The conditions for p, $\neg p$, $\wedge$, $\vee$, until , release , and triggers are identical to the condition for the automaton defined in the Section 2, and thus, can be proved similarly to Proposition 1. Next, we prove that the other conditions hold as well.

1. If $(Z^q \text{ seq } \xi) \in L_{i-1}$, then for some $j \geq i-1$, we have that $\pi, i-1, j, \beta \models L(Z^q)$ and $\pi, j, \beta \models \xi$. Lemma 4 implies at least one of the following holds:
   (a) $j = i-1$ and $q \in W$. In this case condition $6a$ of δ is satisfied.
   (b) $j \geq i$ and there exists $q' \in \Delta(q, \pi_{i-1})$ s.t. $\pi, i, j, \beta \models L(Z^{q'})$. This implies that $\pi, i, \beta \models (Z^{q'} \text{ seq } \xi)$, thus, $(Z^{q'} \text{ seq } \xi) \in L_i$, and condition $6b$ of δ is satisfied.
   (c) There exists an index $i-1 \leq k \leq j$ and a state $q' \in \Delta(q,y)$ s.t. $(i,k) \in \beta$ and $\pi, k, j, \beta \models L(Z^{q'})$. Since $i-1 \leq k$ and $\pi, k, \beta \models \xi$, we have that $o = ((Z^q \text{ seq } \xi), wait(L_{i-1}))$ is possible in $\pi, i-1, \beta$. Thus $o \in L_{i-1}$ and condition $6c$ of δ is satisfied.
2. If $(Z^q \text{ seq } \xi) \in P_{i-1}$, then it is also in L_i. Let $j \geq i-1$ be the minimal index s.t. $\pi, i-1, j, \beta \models L(Z^q)$ and $\pi, j, \beta \models \xi$. Lemma 4 implies at least one of the following holds:
   (a) $j = i-1$ and $q \in W$. In this case condition $8a$ of δ is satisfied.
   (b) $j \geq i$ and there exists $q' \in \Delta(q, \pi_{i-1})$ s.t. $\pi, i, j, \beta \models L(Z^{q'})$. This implies that $\pi, i, \beta \models (Z^{q'} \text{ seq } \xi)$, and that $msi(\pi, i-1, \beta, (Z^q \text{ seq } \xi)) = msi(\pi, i, \beta, (Z^{q'} \text{ seq } \xi)) = j$. Thus, $(Z^{q'} \text{ seq } \xi) \in P_i$, and condition $8b$ of δ is satisfied.

(c) There exists an index $i-1 \leq k \leq j$ and a state $q' \in \Delta(q,y)$ s.t. $(i,k) \in \beta$ and $\pi, k, j, \beta \models L(Z^{q'})$. Since $\pi, k, j, \beta \models L(Z^{q'})$ and $\pi, j, \beta \models \xi$, we have that the obligation $o = ((Z^q \text{ seq } \xi), wait(L_{i-1}))$ is possible in $\pi, i-1, \beta$. Furthermore, $msi(\pi, i-1, \beta, o) \leq msi(\pi, i-1, \beta, (Z^q \text{ seq } \xi)) = j$. Thus $o \in P_{i-1}$ and condition $8c$ of δ is satisfied.

3. If $o = ((Z^q \text{ seq } \xi), \Upsilon) \in L_{i-1}$, then o is possible for $\pi, i-1, \beta$. This implies that there exists an index $j \geq i-1$ s.t. $\pi, j, \beta \models \Upsilon$ and for some $q' \in \Delta(q,y)$, we have $\pi, j, \beta \models (Z^{q'} \text{ seq } \xi)$. If $j = i-1$, then condition $9a$ of δ is satisfied. Otherwise, $j \geq i$. This implies that o is possible for π, i, β. Thus, $o \in L_i$ and condition $9b$ of δ is satisfied.
4. If $o = ((y \text{ seq } \xi), \Upsilon) \in P_{i-1}$, then $o \in L_{i-1}$, thus, o is possible for $\pi, i-1, \beta$. This implies that $j' = msi(\pi, i-1, \beta, o)$ is defined. Thus, there exists an index $j \geq i$ s.t. $\pi, j, \beta \models \Upsilon$, for some $q' \in \Delta(q,y)$, we have $\pi, j, \beta \models (Z^{q'} \text{ seq } \xi)$, and $msi(\pi, i, \beta, (Z^{q'} \text{ seq } \xi)) = j'$. In this case if $j = i-1$, then $msi(\pi, i-1, \beta, (Z^{q'} \text{ seq } \xi)) = msi(\pi, i-1, \beta, o)$, thus, condition $10a$ of δ is satisfied. Otherwise, $j \geq i$. This implies that o is possible for π, i, β. Furthermore, $msi(\pi, i-1, \beta, o) = msi(\pi, i, \beta, o)$. Thus, $o \in L_i$ and condition $10b$ of δ is satisfied.
5. The definitions of P_i implies that if $P_s = \emptyset$, then $P_i = L_i \cap (seq(\varphi) \cup O)$.
6. (Condition 12) Suppose that $wait(L_{i-1}) \subseteq L_{i-1}$. Let $(Z^q \text{ seq } \xi)$ be in L_{i-1}. Then, $\pi, i-1, \beta \models (Z^q \text{ seq } \xi)$. This implies that for some j we have that $\pi, i, j, \beta \models L(Z^q)$ and $\pi, j, \beta \models \xi$. Then, the definition of tight satisfaction implies that there is $w \in L$, $w = x_0, x_1, \ldots, x_n$ and there is a sequence of integers $i_0, i_1, \ldots, i_n, i_{n+1}$, such that $i_0 = i$ and $i_{n+1} = j$. Moreover, for every $0 \leq k \leq n$ the following conditions hold:
   - If $x_k \in 2^{AP}$ then $x_k = \pi_{i_k}$ and $i_{k+1} = i_k + 1$.
   - If $x_k = y$ then $(i_k, i_{k+1}) \in \beta$.

   Let $q, q_1, q_n, \ldots q_{n+1}$ be an accepting run of Z^q on w. We distinguish between three cases:
   (a) If $w = \epsilon$, then $q \in W$, and $j = i-1$ thus condition $6a$ of delta is satisfied and $(Z^q \text{ seq } \xi)$ is strong in L_{i-1}.
   (b) If $j = i_{n+1} = i-1$, then $w = y^n$ and for every $k \leq n$ we have that $\pi, i, \beta \models (Z^{q_k} \text{ seq } \xi)$, and since $wait(L_{i-1}) \subseteq L_{i-1}$, $o_k = ((Z^{q_k} \text{ seq } \xi), wait(L_i))$ is possible in $\pi, i-1, \beta$. This implies that for every $k \leq n+1$ we have that $(Z^{q_k} \text{ seq } \xi) \in L_{i-1}$, and $o_k \in L_{i-1}$. This implies for every $k < n$ there exists a y transition from $(Z^{q_k} \text{ seq } \xi)$ to o_k and from o_k to $(Z^{q_{k+1}} \text{ seq } \xi)$. Furthermore, $q_{n+1} \in W$, thus $(Z^{q_{n+1}} \text{ seq } \xi)$ is strong in L_{i-1} and the condition holds.
   (c) If $j > i-1$, then let l be the maximal index s.t. $i_l = i_0 = i-1$. Then, for every $k \leq l$ we have that $\pi, i-1, \beta \models (Z^{q_k} \text{ seq } \xi)$, and since $wait(L_{i-1}) \subseteq L_{i-1}$, $o_k = ((Z^{q_k} \text{ seq } \xi), wait(L_{i-1}))$ is possible in $\pi, i-1, \beta$. This implies that for every $k \leq l$ we have that $(Z^{q_k} \text{ seq } \xi) \in L_{i-1}$, and $o_k \in L_{i-1}$. This implies for every $k < l$ there exists a y transition from $(Z^{q_k} \text{ seq } \xi)$ to o_k and from o_k to $(Z^{q_{k+1}} \text{ seq } \xi)$. It is left to show that the path ends at a strong element of L_{i-1}. We distinguish between two cases:
      i. If $x_l = y$, then $o = ((Z^{q_l} \text{ seq } \xi), wait(L_{i-1})) \in L_{i-1}$. Since $i_{l+1} > i-1$, o is strong in L_i.

ii. If $x_l \in 2^{AP}$, then $(Z^{q_l}\ \mathsf{seq}\ \xi)$ is strong in L_{i-1}.

7. The proof that condition 13 of δ holds is similar to the proof for condition 12, and thus omitted. □

Proposition 9. *ρ is a fair run of A_φ.*

Proof: The proof that $\Phi_1, \Phi_2, \ldots, \Phi_n$ are satisfied is similar to the proof In Section 2. We prove that for every i there exists $j \geq i$ s.t. $P_j = \emptyset$. we distinguish between two cases:

1. If $P_i = \emptyset$, then we are done.
2. Otherwise, let $k = max\{j | j = msi(\pi, i, \beta, x)$ where $x \in P_i\}$. We prove by induction on j that for every $j \geq k$ one of the following holds:
   (a) There exists $i \leq j' \leq j$ s.t. $P_{j'} = \emptyset$.
   (b) $max\{l | l = msi(\pi, j, \beta, x)$ for some element in $P_j\} \leq k$.
   - Base case: $j = i$, hold trivially.
   - Assume that the induction proposition holds for j. We distinguish between two cases:
     (a) There exists $i \leq j' \leq j$ s.t. $P_{j'} = \emptyset$, then the induction proposition hold for $j + 1$ as well.
     (b) $k' = max\{l | l = msi(\pi, j, \beta, x)$ for some element in $P_j\} \leq k$. If $P_{j+1} = \emptyset$, then the induction hold. Otherwise, for every element x in P_{j+1}, we have $msi(\pi, j + 1, \beta, x) \leq k' \leq k$.

   Since msi of index j is greater or equal to j, for some $i < j \leq k + 1$, we have that $P_j = \emptyset$. □

C.2 $A_\varphi \subseteq L((\exists y)\varphi)$

Lemma 6. *Let π be in $L(A_\varphi)$, then $\pi \in L((\exists y)\varphi)$.*

In the rest of this section, we prove Lemma 6. Let $\rho = (L_0, P_0), (L_1, P_1), \ldots$ be a fair run of A_φ on π. First we construct an interval set β according to ρ. Then, we prove with induction over the structure of φ that for every $i \geq 0$ and every subformula φ' in L_i we have that $\pi, i, \beta \models \varphi'$. For the rest of this section, we fix π and ρ.

We define β as follows: An interval (i, j) is in β iff the following conditions hold:

1. There exists a formula $(Z^q\ \mathsf{seq}\ \xi) \in L_i$, for which condition 6$c$ holds.
2. For some $q' \in \Delta(q, y)$ we have that $(Z^{q'}\ \mathsf{seq}\ \xi) \in L_j$, and $wait(L_i) \subseteq L_j$.

Lemma 7. *For every formula $(Z^q\ \mathsf{seq}\ \xi) \in L_i$ for which conditions 6c of δ holds, there exists an index $j \geq i$ s.t. $wait(L_i) \subseteq L_j$ and for some $q' \in \Delta(q, y)$ we have that $(Z^{q'}\ \mathsf{seq}\ \xi) \in L_j$.*

Proof: Since condition 6c of δ holds, we have $o = ((Z^q\ \mathsf{seq}\ \xi), wait(L_i)) \in L_i$. First we prove that for every $j \geq i$ one of the following conditions holds:

1. There exists $i \leq j' \leq j$ s.t. j' satisfies the conditions of the lemma.
2. $o \in L_j$.

- Base case: $j = i$ holds trivially.

– Assume that the induction proposition holds for j. Then, if here exists $i \leq j' \leq j$ s.t. j' satisfies the conditions of the lemma, then the induction holds for $j+1$ as well. Otherwise, condition $9a$ of δ does not hold for $o \in L_j$. This implies that condition $9b$ does, thus, $o \in L_{j+1}$.

This implies that either there exists an index j that satisfies the conditions of the lemma, in which case the lemma holds, or for every $j \geq i$, we have $o \in L_j$. Since ρ is fair, there exists $k \geq i$ s.t. $P_k = \emptyset$. Then, since $o \in L_{k+1}$, we have that $o \in P_{k+1}$. By the same induction we can prove that either there exists an index $j \geq k+1$ that satisfies the conditions of the lemma, in which case the lemma holds, or for every $j \geq k+1$, we have $o \in P_j$, this case however, contradicts the fairness of ρ, thus the lemma holds. □

Lemma 7 implies that β is well defined.

Proposition 10. *Let $(Z^q \ \mathsf{seq}\ \xi)$ be a formula in L_i. Then, for every $j \geq i$, one of the following conditions holds:*

1. *There exists $j' \leq j$ s.t. $\pi, i, j', \beta \models L(Z^q)$, $\xi \in L_{j'}$.*
2. *There exists a word $w = x_0, x_1, \ldots x_n$ over $2^{AP} \cup \{y\}$, a run $q, q_1, \ldots, q_{n+1}$ of Z^q on w, and a sequence $i_0, i_1, \ldots i_{n+1}$ s.t. $i_0 = i$, $i_{n+1} = j$, and for every $0 \leq k \leq n$, we have the following:*
   (a) If $x_k \in 2^{AP}$, then $x_k = \pi_{i_k}$, $i_{k+1} = i_k + 1$, $(Z^{q_k} \ \mathsf{seq}\ \xi) \in L_{i_k}$, and $(Z^{q_{k+1}} \ \mathsf{seq}\ \xi) \in L_{i_{k+1}}$.
   (b) If $x_k = y$, then $q_{k+1} \in \Delta(q_k, y)$, and for every $i_k \leq l \leq i_{k+1}$, we have $o = ((Z^{q_k} \ \mathsf{seq}\ \xi), wait(L_{i_k})) \in L_l$.

Proof: we prove the proposition by induction on j.

– Base case: $j = i$. Condition 6 of δ implies that one of the following should hold:
  1. $q \in W$ and $\xi \in L_i$. In this case the first condition of the proposition holds for $j' = i$.
  2. $(Z^{q'} \ \mathsf{seq}\ \xi) \in L_{i+1}$ for some $q' \in \Delta(q, \pi_i)$. In this case second condition holds for $w = \pi_i$, the run q, q' and the sequence $i, i+1$.
  3. If $\Delta(q, y) \neq \emptyset$, and $((Z^q \ \mathsf{seq}\ \xi), wait(L_i)) \in L_i$, then the second condition holds for $w = y$, the run q, q' (for some $q' \in \Delta(q, y)$), and the sequence i, i.
– Induction step: Assume that the proposition holds for j. If condition 1 of proposition holds for j, then it holds for $j+1$ as well. Otherwise, there exists a word $w = x_0, x_1, \ldots x_n$ over $2^{AP} \cup \{y\}$ a run $q, q_1, \ldots, q_{n+1}$ of Z^q on w, and a sequence $i_0, i_1, \ldots i_{n+1}$ s.t. $i_0 = i$, $i_{n+1} = j$, and for every $0 \leq k \leq n$, we have the following:
  1. If $x_k \in 2^{AP}$, then $q_{k+1} \in \Delta(q_k, \pi_{i_k})$, $i_{k+1} = i_k + 1$, $(Z^{q_k} \ \mathsf{seq}\ \xi) \in L_{i_k}$, and $(Z^{q_{k+1}} \ \mathsf{seq}\ \xi) \in L_{i_{k+1}}$.
  2. If $x_k = y$, then $q_{k+1} \in \Delta(q_k, y)$, and for every $i_k \leq l \leq i_{k+1}$, we have $o = ((Z^{q_k} \ \mathsf{seq}\ \xi), wait(L_{i_k})) \in L_l$.

  We distinguish between four cases:
  1. If $x_n \in 2^{AP}$, and $(Z^{q_{n+1}} \ \mathsf{seq}\ \xi) \in L_j$ is strong in L_j. Condition 6 of δ implies that one of the following should hold:
     (a) $q_{n+1} \in W$ and $\xi \in L_j$. In this case condition 1 of the proposition is satisfied with $j' = j$.

(b) $(Z^{q'}\ \mathsf{seq}\ \xi) \in L_{j+1}$ for some $q' \in \Delta(q_{n+1}, \pi_j)$. In this case condition 2 of the proposition holds for $w' = w \cdot \pi_j$, the run $q, q_1, \ldots, q_{n+1}, q'$, and the sequence $i_0, i_1, \ldots, i_{n+1}, j+1$.

2. If $x_n = y$, and $o = ((Z^{q_n}\ \mathsf{seq}\ \xi), wait(L_{i_n})) \in L_j$ is strong in L_j. Then, Condition 9 of δ implies that $o \in L_{j+1}$. In this case the second condition holds for $w' = w$, the run $q, q_1, \ldots, q_{n+1}$, and the sequence $i_0, i_1, \ldots, i_n, j+1$. (note that we remove the old i_{n+1}).
3. If $x_n \in 2^{AP}$, and $(Z^{q_{n+1}}\ \mathsf{seq}\ \xi)) \in L_j$ is not strong in L_j. Condition 6 of δ implies the following should hold: There exists $q' \in \Delta(q_{n+1}, y)$, and $o = ((Z^{q_n}\ \mathsf{seq}\ \xi), wait(L_j)) \in L_j$. We distinguish between two cases:
   (a) If $wait(L_j) \not\subseteq L_j$, condition $9a$ of δ do not hold for o. This implies that condition $9b$ of δ does hold for o. In this case the second condition holds for $w' = w \cdot y$, the run $q, q_1, \ldots q_{n+1}, q'$ (for some $q' \in \Delta(q, y)$) and the sequence $i_0, i_1, \ldots, i_{n+1}, j+1$.
   (b) If $wait(L_j) \subseteq L_j$, then condition 12 of δ implies that there exists a path of y transitions from $(Z^{q_{n+1}}\ \mathsf{seq}\ \xi)$ to a strong element in L_j. Note that if there is a y transition from $(Z^{q}\ \mathsf{seq}\ \xi)$ to $o = ((Z^{q}\ \mathsf{seq}\ \xi), \Upsilon)$, and a y transition from o to $(Z^{q'}\ \mathsf{seq}\ \xi)$, then $q' \in \Delta(q, y)$. This implies that there exists a sequence $q_{n+1}, q_{n+2}, \ldots q_{n+m}$ s.t. for every $n+1 \leq l < n+m$, $q_{l+1} \in \Delta(q_l, y)$, and either $(Z^{q_{n+m}}\ \mathsf{seq}\ \xi)$ is strong, or $o = ((Z^{q_{n+m}}\ \mathsf{seq}\ \xi), wait(L_j))$ is strong. If $(Z^{q_{n+m}}\ \mathsf{seq}\ \xi)$ is strong, then we have the same proof as in item 1 with $w \cdot y^m$, the run $q, q_1 \ldots q_{n+m}$ and the sequence $i_0, i_1, \ldots, i_{n+1}, j, j, \ldots, j, j+1$. If $o = ((Z^{q_{n+m}}\ \mathsf{seq}\ \xi), wait(L_j))$ is strong, we have the same proof as in item 2 with $w \cdot y^m$, the run $q, q_1 \ldots q_{n+m}$, and the sequence $i_0, i_1, \ldots, i_{n+1}, j, j, \ldots, j, j+1$.
4. If $x_n = y$, and $o = ((Z^{q_n}\ \mathsf{seq}\ \xi), wait(L_{i_n})) \in L_j$ is not strong in L_j. Then, Condition 9 of δ implies following should hold: For some $q' \in \Delta(q_n, y)$, we have that $(Z^{q'}\ \mathsf{seq}\ \xi) \in L_j$ and $\Upsilon \subseteq L_j$. If $(Z^{q'}\ \mathsf{seq}\ \xi)$ is strong, then the proof is as in item 1, otherwise it is as in item 3.

□

Proposition 11. *Let $(Z^q\ \mathsf{seq}\ \xi)$ be a formula in L_i, and let $l \geq i$ be an index s.t. $P_l = \emptyset$. Then, for every $j \geq l+1$, one of the following conditions holds:*

1. *There exists $i \leq j' \leq j$ s.t. $\pi, i, j', \beta \not\models L(Z^q)$, $\xi \in L'_j$.*
2. *There exists a word $w = x_0, x_1, \ldots x_n$ over $2^{AP} \cup \{y\}$, a run $q, q_1, \ldots, q_{n+1}$ of Z^q on w, and a sequence $i_0, i_1, \ldots i_{n+1}$ s.t. $i_0 = i$, $i_{n+1} = j$, and for every $0 \leq k \leq n$, we have the following:*
   (a) If $x_k \in 2^{AP}$, then $q_{k+1} \in \Delta(q_k, \pi_{i_k})$, $i_{k+1} = i_k + 1$, $(Z^{q_k}\ \mathsf{seq}\ \xi) \in L_{i_k}$, $(Z^{q_{k+1}}\ \mathsf{seq}\ \xi) \in L_{i_{k+1}}$, and if $i_k \geq l+1$, then $(Z^{q_k}\ \mathsf{seq}\ \xi) \in P_{i_k}$, $(Z^{q_{k+1}}\ \mathsf{seq}\ \xi) \in P_{i_{k+1}}$.
   (b) If $x_k = y$, then $q_{k+1} \in \Delta(q_k, y)$, and for every $i_k \leq l \leq i_{k+1}$, we have $o = ((Z^{q_k}\ \mathsf{seq}\ \xi), wait(L_{i_k})) \in L_l$, and if $i_k \geq l+1$, then $o \in P_{i_k}$.

The proof of Proposition 11 is similar to the proof of Proposition 10, and thus omitted. Proposition 11 implies the following Corollary.

Corollary 1. *Let $(Z^q\ \mathsf{seq}\ \xi)$ be a formula in L_i, and let $l \geq i$ be an index s.t. $P_l = \emptyset$. Then, one of the following conditions holds:*

1. *There exists $j \geq i$ s.t. $\pi, i, j, \beta \models L(Z^q)$, $\xi \in L_j$.*
2. *There exists an infinite sequence $i_0, i_1, \ldots i_{n+1}$ s.t. $i_0 = i$, and for every $k \geq 0$, we have the following:*
   (a) If $x_k \in 2^{AP}$ and $i_k \geq l+1$, then $(Z^{q_k}$ seq $\xi) \in P_{i_k}$, $(Z^{q_{k+1}}$ seq $\xi) \in P_{i_{k+1}}$.
   (b) If $x_k = y$ and $i_k \geq l+1$, then $o \in P_{i_k}$.
   (c) For every $j \geq l$ there exists j' s.t. $i_{j'} \geq j$.

We now complete the proof of Lemma 6. We prove by induction over the structure of φ that for every $i \geq 0$ and every subformula φ' in L_i we have that $\pi, i, \beta \models \varphi'$.

- Base case: For p and $\neg p$ the definition of the automaton implies that the lemma holds.
- Induction step: The induction step for the operators $\wedge, \vee,$ until , release , next is identical to the proof of Lemma 3. We prove the induction step for the seq , and triggers operators.
  - Let $(Z^q$ seq $\xi)$ be a formula in L_i. Since ρ is fair, there exists $l \geq i$ s.t. $P_l = \emptyset$. Then, Corollary 1 implies that one of the following conditions holds:
    1. There exists $j \geq i$ s.t. $\pi, i, j, \beta \models L(Z^q)$, $\xi \in L_j$. In this case $\pi, i, \beta \models (Z^q$ seq $\xi)$.
    2. There exists an infinite sequence $i_0, i_1, \ldots i_{n+1}$ s.t. $i_0 = i$, and for every $k \geq 0$, we have the following:
       (a) If $x_k \in 2^{AP}$ and $k \geq l+1$, then $(Z^{q_k}$ seq $\xi) \in P_{i_k}$, $(Z^{q_{k+1}}$ seq $\xi) \in P_{i_{k+1}}$.
       (b) If $x_k = y$ and $k \geq l+1$, then $o \in P_{i_k}$.
       (c) For every $j \geq l$ there exists j' s.t. $i_{j'} \geq j$.

    In this case for every $j \geq l+1$, we have $P_j \neq \emptyset$, thus ρ is not fair, contradiction.
  - Let $(Z^q$ triggers $\xi) \in L_i$. We need to prove that for every $j \geq i$ s.t. $\pi, i, j, \beta \models L(Z^q)$, we have $\xi \in L_j$. Suppose that for $j \geq i$ we have that $\pi, i, j, \beta \models L(Z^q)$, then there is $w \in L(Z^q)$, $w = x_0, x_1, \ldots, x_n$ and there is a sequence of integers $i_0, i_1, \ldots, i_n, i_{n+1}$, such that $i_0 = i$ and $i_{n+1} = j$. Moreover, for every $0 \leq k \leq n$ the following conditions hold:
    * If $x_k \in 2^{AP}$ then $x_k = \pi_{i_k}$ and $i_{k+1} = i_k + 1$.
    * If $x_k = y$ then $(i_k, i_{k+1}) \in \beta$.

    Let $q, q_1, q_2, \ldots, q_{n+1}$ be an accepting run of Z^q on w. We prove with induction on $0 \leq k \leq n$ that $(Z^{q_k}$ seq $\xi) \in L_{i_k}$. Furthermore, if $q_k \in W$, then $\xi \in L_{i_k}$.
    * Base case $k = 0$, then $i_0 = i$. By definition $(Z^q$ seq $\xi) \in L_i$. If $q \in W$, condition $7a$ of δ implies that $\xi \in L_{i_0}$.
    * Assume that the lemma holds for k, then $(Z^{q_k}$ seq $\xi) \in L_{i_k}$. we distinguish between two cases:
      1. If $x_k = y$, then the definition of β implies that for every $j \geq i_k$ s.t. $(i_k, j) \in \beta$, in particular i_{k+1}, we have $(Z^{q_{k+1}}$ seq $\xi) \in L_{i_{k+1}}$. Condition $7a$ of δ implies that if $q_{k+1} \in W$, then $\xi \in L_{i_{k+1}}$.
      2. If $x_k \in 2^{AP}$, then condition $6b$ of δ implies that $(Z^{q_{k+1}}$ seq $\xi) \in L_{i_{k+1}}$ and condition $7a$ of δ implies that if $q + k + 1 \in W$, then $\xi \in L_{i_{k+1}}$.

□

D lower bound

In the *exponential bounded-tiling* problem we are given a fixed set T of tiles, two relations $H, V \subseteq T \times T$, two tiles $t_{init}, t_{fin} \in T$, and an integer n. The goal is to tile a $(2^n \times 2^n)$-square so that horizontal neighbors belong to H, vertical neighbors belong to V, the first tile in the first row is t_{init}, and the first tile in the last row is t_{fin}. Thus, formally, a legal tiling is a function $t : \{0, \ldots, 2^n - 1\}^2 \to T$ such that the following hold:

- for all $0 \leq i \leq 2^n - 2$ and $0 \leq j \leq 2^n - 1$, we have that $H(t(i, j), t(i + 1, j))$,
- for all $0 \leq i \leq 2^n - 1$ and $0 \leq j \leq 2^n - 2$, we have that $V(t(i, j), t(i, j + 1))$,
- $t(0, 2^n - 1) = t_{init}$, and $t(0, 2^n - 1) = t_{fin}$.

The exponential bounded tiling problem is known to be NEXPTIME-hard [18].

Theorem 10. *The regular vacuity problem for RELTL is NEXPTIME-hard.*

Proof: We do a reduction from the exponential bounded tiling problem. Given a tiling system $\mathcal{T} = \langle T, H, V, n, t_{init}, t_{fin} \rangle$, we construct a model $M_{\mathcal{T}}$ of a fixed size and an RELTL formula φ of length $O(n)$ such that $\neg\varphi$ is not regularly vacuous in $M_{\mathcal{T}}$ iff there is a legal tiling t for $\mathcal{T}$.

We encode the tiles in T by a set $AP(T) = \{p_1, \ldots, p_m\}$ of atomic propositions. We define the formula φ over the set $AP = AP(T) \cup \{b, c, d, r\}$ of atomic propositions. The task of the last four atoms will be explained shortly. Since T is fixed, so is AP.

Consider an infinite word π over 2^{AP}. For an atomic proposition $p \in AP$ and a point u in π, we use $p(u)$ to denote the truth value of p at u. That is, $p(u)$ is 1 if p holds at u and is 0 if p does not hold at u. We divide the word π to blocks of length $2n$. Every block corresponds to a single location in the $(2^n \times 2^n)$-square. Consider a block $u_1, \ldots, u_{2n}$ that corresponds to location $[i, j]$ of the square. We use the point u_1 to encode the tile in location $[i, j]$. Thus, the bit vector $p_1(u_1), \ldots, p_m(u_1)$ encodes the tile $t(i, j)$. We use the atomic proposition b to mark the beginning of the block; that is, b holds on u_1 and fails on $u_2, \ldots, u_{2n}$. This is enforced by the formula φ. Thus, φ contains a conjunct[4]

$$b \wedge (\bigwedge_{1 \leq k \leq 2n-1} \mathsf{next}^{\,k} \neg b) \wedge \mathsf{globally}\,(b \leftrightarrow \mathsf{next}^{\,2n} b).$$

The block $u_1, \ldots, u_{2n}$ also encodes the location of the tile in the square. Since the square is of dimensions $2^n \times 2^n$, this location is a pair $\langle i, j \rangle$, for $0 \leq i, j \leq 2^n - 1$, where i is the column of the tile and j is its row. Encoding the location eliminates the need for exponentially many next operators when we attempt to relate tiles that are vertical neighbors. Encoding is done by the atomic proposition c, called *counter*. Let $c(u_n), \ldots, c(u_1)$ encode i, and $c(u_{2n}), \ldots, c(u_{n+1})$ encode j. Note that, for technical convenience, the least significant bits of the counters are in u_1 and u_{n+1}, and i is encoded before j. A sequence of 2^n blocks corresponds to 2^n tiles and, when starts

[4] Note that the formula is of quadratic length. An equivalent formula of a linear length replaces conjuncts like $\bigwedge_{1 \leq k \leq 2n-1} X^k p$ by $X(p \wedge X(p \wedge \cdots \wedge Xp) \cdots)$. In order to keep the reference to indices clear, we describe here and in the sequel the quadratic version.

with $i = 0$, encodes some row j in the square. The values of the counters along this sequence go from $\langle 0, j\rangle$ to $\langle 2^n - 1, j\rangle$, and then start again with $i = 0$, but with an increased j. Thus, the next sequence goes from $\langle 0, j + 1\rangle$ to $\langle 2^n - 1, j + 1\rangle$. The way we encode the counters guarantees that an increase of the counter by one corresponds to either a transition from $\langle i, j\rangle$ to $\langle i + 1, j\rangle$, in case $i \neq 2^n - 1$, or to a transition from $\langle 2^n - 1, j\rangle$ to $\langle 0, j + 1\rangle$, otherwise. A proper behavior of the counters is enforced by φ. Since we want the length of φ to be $O(n)$, we need also an atomic proposition d that acts as a "carry" bit. Note that $b \vee d$ holds in a point u_i iff $c(u_i) \neq c(u'_i)$, where u' is the successor block of u. Formally, φ contains the following conjuncts.

1. The counter starts at 0: $\bigwedge_{0\leq k\leq 2n-1}$ next $^k \neg c$.
2. The counter is increased properly. Note that as we always want to increase the counter by 1 we take b as a carry to the least significant bit:
   - globally $(((b \vee d) \wedge \neg c) \rightarrow (\text{ next } (\neg d) \wedge \text{ next } ^{2n} c))$.
   - globally $((\neg(b \vee d) \wedge \neg c) \rightarrow (\text{ next } (\neg d) \wedge \text{ next } ^{2n} \neg c))$.
   - globally $(((b \vee d) \wedge c) \rightarrow (\text{ next } d \wedge \text{ next } ^{2n} \neg c))$.
   - globally $((\neg(b \vee d) \wedge c) \rightarrow ((\text{ next } \neg d) \wedge \text{ next } ^{2n} c))$.

Since each location is encoded by a block of length $2n$, the whole tiling is encoded in a finite prefix of π, namely $\pi_0, \ldots \pi_{2n(2^n)^2-1}$. We use the atomic proposition r in order to label this "relevant prefix." More precisely, r holds exactly in this prefix. Thus, φ contains a conjunct

$$(r \text{ until } (b \wedge (\bigwedge_{0\leq k\leq 2n-1} \text{next}^{\,k}(r \wedge c)))) \wedge \text{globally } (b \wedge (\bigwedge_{0\leq k\leq 2n-1} \text{next}^{\,k} c)) \rightarrow \text{next}^{\,2n} \text{ globally } \neg r).$$

Let $t_0 \ldots t_{2^n-1}, t'_0 \ldots t'_{2^n-1}$ be two successive rows of the tiling t. For each i, $0 \leq i \leq 2^n - 1$, we know, given t_i, the possible values for t_{i+1} (these for which $H(t_i, t_{i+1})$, in case $i < 2^n - 1$) and the possible values for t'_i (these for which $V(t_i, t'_i)$). Consistency with H and V gives us a necessary condition for a word to encode a legal tiling. In addition, the tiling should satisfy the edge conditions; it should start with t_{init} and has t_{fin} in position $[0, 2^n - 1]$. For a tile $t \in T$, let $\rho(t)$ be the propositional formula over AP that encodes t. That is, $\rho(t)$ holds in point u_1 of exactly all blocks that encode the tile t. In order to make sure that the edge conditions hold, φ contains the conjunct

$$\rho(t_{init}) \wedge \text{ globally } ((b \wedge (\bigwedge_{0\leq k\leq n-1} \text{next}^{\,k} \neg c) \wedge (\bigwedge_{n+1\leq k\leq 2n} \text{next}^{\,k} c)) \rightarrow \rho(t_{fin})).$$

Since the distance between the point where t_i is encoded to the one where t_{i+1} is encoded is exactly $2n$, it is also easy to specify the conditions for horizontal neighbors. Note that the conditions are imposed only when $i \neq 2^n - 1$:

$$\text{globally } ((b \wedge (\bigvee_{0\leq k\leq n-1} \text{next}^{\,k} \neg c) \wedge \rho(t)) \rightarrow \text{ next}^{\,2n} \bigvee_{t':H(t,t')} \rho(t')).$$

The difficult part in the reduction is in guaranteeing that the condition for vertical neighbors hold. This is where regular vacuity comes into the picture. They enable us to relate $t[i, j]$ with $t[i, j + 1]$, for all i and j. Let e be a regular expression. Consider the formula ξ_1 below. The formula says that whenever we are in a beginning of a block

u corresponding to position $[i, j]$, for some i and j, then the regular expressions e is tightly satisfied only in intervals that end when the block u' that corresponds to position $[i, j + 1]$ starts. To see this, recall that $b \vee d$ holds in a point u_i iff $c(u_i) \neq c(u'_i)$, and note that the formula requires the value of the counter in $u'_{2n}, \ldots, u'_{n+1}$ (that is, the j-coordinate of u') to be greater by 1 than the value in $u_{2n}, \ldots, u_{n+1}$ (the j-coordinate of u), and requires the value of the counter in $u'_n, \ldots, u'_1$ (the i-coordinate of u') to be equal to the value of the counter in $u_n, \ldots, u_1$ (the i-coordinate of u). Note that the requirement is imposed only when u is not in the last row, thus $j \neq 2^n - 1$:

$$\xi_1 = \textsf{ globally } ((b \wedge \textsf{ next }^n \bigvee_{0 \leq k \leq n-1} \textsf{ next }^k \neg c) \rightarrow \bigwedge_{1 \leq k \leq n} \theta_1^k \wedge \theta_2^k \wedge \textsf{ next }^n (\theta_3^k \wedge \theta_4^k)), \text{ where}$$

- $\theta_1^k = (\textsf{ next }^k c) \rightarrow e$ TRIGGERS $\textsf{ next }^k c$,
- $\theta_2^k = (\textsf{ next }^k \neg c) \rightarrow e$ TRIGGERS $\textsf{ next }^k \neg c$,
- $\theta_3^k = (\textsf{ next }^k ((c \wedge \neg(b \vee d)) \vee (\neg c \wedge (b \vee d)))) \rightarrow e$ TRIGGERS $\textsf{ next }^k c$, and
- $\theta_4^k = (\textsf{ next }^k ((\neg c \wedge \neg(b \vee d)) \vee (c \wedge (b \vee d)))) \rightarrow e$ TRIGGERS $\textsf{ next }^k \neg c$.

Consider now the formula ξ_2 below. The formula says that whenever a block u, not in the last row, starts, there is a block u' in the relevant prefix of π that starts when an interval satisfying e ends, and the blocks u and u' encode tiles that are related by V.

$$\xi_2 = \bigwedge_{t \in T} \textsf{ globally } ((r \wedge b \wedge (\bigvee_{1 \leq k \leq n} \textsf{ next }^{n+k} \neg c) \wedge \rho(t)) \rightarrow e \text{ SEQ}(r \wedge \bigvee_{t' : V(t,t')} \rho(t'))).$$

The formula φ contains a conjunct $\xi_1 \wedge \xi_2$, with $e = b$ (in fact any $e \neq \mathbf{true}^{2n2^n}$ will do). Note that for $e = b$, the formula ξ_1 does not hold in a path in which the counters are increased properly.

Let $M_{\mathcal{T}}$ be a Kripke structure that generates all the computations over AP. Thus, $M_{\mathcal{T}} = \langle AP, 2^{AP}, 2^{AP}, 2^{AP} \times 2^{AP}, L \rangle$ with $L(\sigma) = \sigma$. We prove that $\neg\varphi$ is not regularly vacuous in $M_{\mathcal{T}}$ iff there is a legal tiling t for $\mathcal{T}$.

Assume first that there is a legal tiling t for $\mathcal{T}$. Recall that ξ_1 does not hold in a path in which the counters are increased properly. Therefore, φ is not satisfiable, and all the paths of $M_{\mathcal{T}}$ satisfy $\neg\varphi$. We show that all the regular expressions in $\neg\varphi$ affect it, thus $\neg\varphi$ is not regularly vacuous in $M_{\mathcal{T}}$. The single regular expression is $\neg\varphi$ is $e = b$. By the definition of φ, the path π that describes t satisfies $(\exists y).\varphi[e \leftarrow y]$. Indeed, since π describes a legal tiling and since the distance between points where successive points start is $2n2^n$, the interval set β that contains all intervals of length $2n2^n$ is such that $\pi, 0, \beta \models \varphi$. It follows that e affects $\neg\varphi$ in $M_{\mathcal{T}}$, thus $\neg\varphi$ is not regularly vacuous in $M_{\mathcal{T}}$.

For the other direction, assume that $\neg\varphi$ is not regularly vacuous in $M_{\mathcal{T}}$. Since e is the only regular expression in φ, it follows that $M_{\mathcal{T}} \models (\exists y)\varphi[e \leftarrow y]$. Let β be an interval set such that $M_{\mathcal{T}}$ has a path π for which $\pi, 0, \beta \models \varphi$. The conjuncts of φ that are independent of e (that is, all conjuncts except for $\xi_1 \wedge \xi_2$) guarantee that the path π describes a tiling that satisfies the edge conditions and the conditions for horizontal neighbors. By the definition of ξ_1, the path π is such that whenever we are in a beginning of a block u corresponding to position $[i, j]$, for some i and j, then the regular expressions e is tightly satisfied only in intervals that end when the block u' that corresponds to position $[i, j + 1]$ starts. Therefore, β contains only intervals

of length $2n2^n$. Hence, ξ_2 guarantees that π describes a tiling that also satisfies the conditions for vertical neighbors. Thus, π describes a legal tiling for $\mathcal{T}$, and we are done. □

E Alternative Definition for RELTL

In industrial specification languages the seq and triggers operators have a slightly different semantics. For pragmatic reasons, we present the semantics for ALTL.

- $\pi, i \models Z$ seq $'\varphi$ if for some $j > i$ we have $\pi, i, j \models\!\!\!\equiv L(Z)$ and $\pi, j-1 \models \varphi$.
- $\pi, i \models Z$ triggers $'\varphi$ if for all $j > i$ such that $\pi, i, j \models\!\!\!\equiv L(Z)$ we have $\pi, j-1 \models \varphi$.

We now present a translation between the different semantics. The first direction is simple, for regular expression e s.t. $\epsilon \notin L(Z)$, we have that given a formula $\varphi = (Z \text{ seq } \xi)$, the formula $\varphi' = (Z \text{ seq}' \text{ next } \xi)$ is equivalent. Similarly, given a formula $(Z \text{ triggers } \xi)$, the formula $(Z \text{ triggers next } \xi)$ is equivalent. In case that $\epsilon \in L(Z)$, the equivalent seq formula is $\varphi' = \xi \vee (Z \text{ seq}' \text{ next } \xi)$, and the triggers equivalent formula is $\xi \wedge (Z \text{ triggers next } \xi)$.

Next we show a translation from the industrial semantics to ours.

Definition 10. *Let L be a language of finite words over alphabet Σ, and let $a \in \Sigma$. We define $L^{-a} = \{w | w \cdot a \in L\}$.*

Definition 11. *Let $Z = \langle \Sigma, Q, \Delta, q_0, W \rangle$ be an NFW, and $a \in \Sigma$, we define $Z^{-a} = \langle \Sigma, Q, \Delta, q_0, W' \rangle$ where $w' = \{q | \Delta(q, a) \cap W \neq \emptyset\}$.*

Lemma 8. *Let $Z = \langle \Sigma, Q, \Delta, q_0, W \rangle$ be an NFW, and $a \in \Sigma$. Then, $L(Z)^{-a} = L(Z^{-a})$.*

Lemma 9. *For every infinite word π, $i \geq 0$, an NFW Z, and a formula ξ, we have that $\pi, i \models (Z \text{ seq}' \xi)$ iff $\pi, i, \models \vee_{a \in \Sigma}(Z^{-a} \text{ seq } (a \wedge \xi))$.*

Proof: Suppose that $\pi, i \models (Z \text{ seq}' \xi)$, then for some $j > i$ we have that $\pi, i, j \models\!\!\!\equiv L(Z)$ and $\pi, j-1 \models \xi$. This implies that $\pi_i, \pi_{i+1}, \ldots \pi_{j-1} \in L(Z)$. Let $a = \pi_{j-1}$, then by Lemma 8, $\pi_i, \pi_{i+1}, \ldots \pi_{j-2} \in L(Z^{-a})$, and $\pi, j-1 \models (a \wedge \xi)$. Thus, $\pi, j \models (Z^{-a} \text{ seq } (a \wedge \xi))$.

Next, suppose that $\pi, i, \models \vee_{a \in \Sigma}(Z^{-a} \text{ seq } (a \wedge \xi))$. Then for some $a \in \Sigma$, we have that $\pi, i, \models (Z^{-a} \text{ seq } (a \wedge \xi))$. This implies that for some $j \geq i$, we have that $\pi_i, \pi_{i+1}, \ldots \pi_{j-1} \in L(Z^{-a})$, and that $\pi, j \models (a \wedge \xi)$. Since $\pi, j \models a$ we have that $\pi_i, \pi_{i+1}, \ldots \pi_{j-1}, \pi_j \in L(Z)$. Thus $\pi, i, j+1 \models\!\!\!\equiv Z$ and $j+1-1 \models \xi$. This implies that $\pi, i \models (Z \text{ seq}' \xi)$. □

Lemma 10. *For every infinite word π, $i \geq 0$, an NFW Z, and a formula ξ, we have that $\pi, i \models (Z \text{ triggers}' \xi)$ iff $\pi, i, \models \wedge_{a \in \Sigma}(Z^{-a} \text{ seq } (a \rightarrow \xi))$.*

Proof: Suppose that $\pi, i \models (Z \text{ triggers}' \xi)$, then for every $a \in \Sigma$ and $j \geq i$ s.t. $\pi, i, j \models\!\!\!\equiv L(Z^{-a})$ we distinguish between two cases:

1. $\pi_j = a$ in this case we have that $\pi, i, j+1 \models\!\!\!\equiv L(Z)$, thus $\pi, j \models \xi$, and $\pi, i \models (Z^{-a} \text{ seq } (a \rightarrow \xi))$.
2. Otherwise, $\pi, j \models (a \rightarrow \xi)$. Thus, $\pi, i \models (Z^{-a} \text{ seq } (a \rightarrow \xi))$.

Next, suppose that $\pi, i, \models \wedge_{a\in\Sigma}(Z^{-a}\ \mathsf{seq}\ (a \rightarrow \xi))$. Let $j > i$ s.t. $\pi, i, j \models L(Z)$. Then, for $a = \pi_{j-1}$, we have $\pi, i, j-1 \models L(z^{-a})$. Since $\pi, i \models (Z^{-a}\ \mathsf{seq}\ (a \rightarrow \xi))$, and $\pi_{j-1} = a$, we have that $\pi, j-1 \models \xi$. Thus, $\pi, i \models (Z\ \mathsf{triggers}\ '\xi)$. □

F Proofs for section 5

Theorem ?? For every structure M, RELTL formula φ and regular expression e of pure polarity, $M \models \varphi[e \leftarrow \bot]$ iff $M \models (\forall y)\varphi[e \leftarrow y]$.

Proof: If $M \models \forall y\varphi[e \leftarrow y]$ then $M, \beta \models \varphi[e \leftarrow y]$ for every assignment β, including $\beta_\emptyset = \emptyset$ and $\beta_I = I$ (the set of all intervals). $M, \beta_\emptyset \models \varphi[e \leftarrow y]$ implies $M \models \varphi[e \leftarrow \mathbf{false}]$ since no interval satisfies $\mathbf{false}$. $M, \beta_I \models \varphi[e \leftarrow y]$ implies $M \models \varphi[e \leftarrow \mathbf{true}^*]$ since every interval satisfies $\mathbf{true}^*$. Thus $M \models \varphi[e \leftarrow \bot]$.

The other direction is proved by induction on the structure of φ (given in positive normal form). As regular expression are only used on the left hand side of seq and $\mathsf{triggers}$, the base case and all operators apart from seq and $\mathsf{triggers}$ are immediate.

Let $\varphi = E\ \mathsf{seq}\ \xi$ where E is a RELTL regular expression and e is a sub-regular expression of E. The polarity of e is positive in φ and therefore $\bot \equiv \mathbf{false}$. If $M, \pi, i \models \varphi[e \leftarrow \mathbf{false}]$ then there exists some $j \geq i$ s.t. $M, \pi, i, j \models E[e \leftarrow \mathbf{false}]$ and $M, \pi, j \models \xi$. Let $b_0, b_1, \ldots, b_{j-1-i}$ be a word in $L(E[e \leftarrow \mathbf{false}])$ s.t. $M, \pi, k \models b_{k-i}$ for all $i \leq k < j$. Clearly $b_{k-i} \neq \mathbf{false}$. This implies that $M, \pi, i, j \models E$ regardless of e. Thus $M, \pi, i \models \forall y\varphi[e \leftarrow y]$.

Let $\varphi = E\ \mathsf{triggers}\ \xi$ where E is a RELTL regular expression and e is a sub-regular expression of E. The polarity of e is negative in φ and therefore $\bot \equiv \mathbf{true}^*$. Assume that $M, \pi, i \not\models \forall y E[e \leftarrow y]\ \mathsf{triggers}\ \xi$. This implies that $M, \pi, i \models \exists y E[e \leftarrow y]\ \mathsf{seq}\ \neg\xi$. Thus $M, \pi, i, j, \beta \models E[e \leftarrow y]$ for some $j \geq i$ and interval set β, and $M, \pi, j \models \neg\xi$. By the definition of tight satisfaction there exists a word $w = b_0, b_1, \ldots, b_n$ over $AP \cup \{y\}$ s.t. $M, \pi, i, j, \beta \models w$. Furthermore, if $b_m \in AP$, $0 \leq m \leq n$, then there exists a k s.t. $i \leq k \leq j$ and $M, \pi, k, k+1 \models b_m$. Otherwise $b_m = y$ and $M, \pi, k, k' \models b_m$ for some k' s.t. $i \leq k \leq k' \leq j$. We now show that there exists a word $w' \in L(E[e \leftarrow \mathbf{true}^*])$ s.t. $M, \pi, i, j \models w'$. The word w' is equal to w except that every $b_m = y$ is replaced by $k'_m - k_m$ concatenated $\mathbf{true}$ (where $M, \pi, k_m, k'_m \models b_m$). This implies that $M, \pi, i, j \models E[e \leftarrow \mathbf{true}^*]$. Since $M, \pi, j \models \neg\xi$ we have $M, \pi, i \models E[e \leftarrow \mathbf{true}^*]\ \mathsf{seq}\ \neg\xi$, which implies $M, \pi, i \not\models E[e \leftarrow \mathbf{true}^*]\ \mathsf{triggers}\ \xi$. □