Linear Time Logics Around PSL: Complexity, Expressiveness, and a Little Bit of Succinctness

Martin Lange

Department of Computer Science, University of Aarhus, Denmark

Abstract. We consider linear time temporal logic enriched with semi-extended regular expressions through various operators that have been proposed in the literature, in particular in Accelera's Property Specification Language. We obtain results about the expressive power of fragments of this logic when restricted to certain operators only: basically, all operators alone suffice for expressive completeness w.r.t. ω-regular expressions, just the closure operator is too weak. We also obtain complexity results. Again, almost all operators alone suffice for EXPSPACE-completeness, just the closure operator needs some help.

1 Introduction

Pnueli has acquired the linear time temporal logic LTL from philosophy for the specification of runs of reactive systems [7]. It has since been established as a milestone in computer science, particularly in automatic program verification through model checking. Its success and popularity as such is not matched by its expressive power though which was shown to be rather limited: LTL is equi-expressive to First-Order Logic on infinite words and, thus, can express exactly the star-free ω-languages [5,11]. This is not enough for compositional verification for instance.

Some early attempts have been made at extending LTL with the aim of capturing all ω-regular languages. Wolper's ETL incorporates non-deterministic Büchi automata as connectives in the language [14]. QPTL extends LTL with quantifications over atomic propositions in the style of Monadic Second-Order Logic [10]. The linear time μ-calculus μTL allows arbitrary recursive definitions of properties via fixpoint quantifiers in the logic [2,13].

None of these logics has seen an impact that would make it replace LTL as the most popular specification language for linear time properties. This may be because of non-elementary complexity (QPTL), syntactical inconvenience (ETL), and difficulty in understanding specifications (μTL), etc.

Nevertheless, the need for a convenient temporal specification formalism for ω-regular properties is widely recognised. This is for example reflected in the definition of Accelera's *Property Specification Language* (PSL), designed to provide a general-purpose interface to hardware verification problems [1]. At its temporal layer it contains a logic that is capable of expressing all ω-regular properties. This is mainly achieved through the introduction of operators in LTL that take semi-extended regular expressions as arguments. Following common agreement in the

© Springer-Verlag Berlin Heidelberg 2007

temporal logic community we also use PSL to refer to this logic. To be more precise: PSL is often used to describe a temporal logic enhanced with semi-extended regular expressions through various operators. This is because the original definition of Accelera's PSL allows a multitude of operators of which some are just syntactical sugar. For example, the intersection operator on regular expressions that does not require the same length of its operands can easily be expressed using the usual intersection operator: $\alpha \& \beta \equiv (\alpha; \Sigma^*) \&\& (\beta; \Sigma^*)$, etc.

Much effort has already been taken to design verification algorithms for PSL properties, for example through automata-theoretic constructions [3]. Nevertheless, we are not aware of a rigorous analysis of its computational complexity, and the expressive power of fragments obtained by restricting the use of temporal operators in the logic. In order to make up for this, we consider linear time temporal logic enriched with various operators that have occurred in the literature so far, not only in PSL. For pragmatics and complexity-theoretic upper bounds it is clearly desirable to consider the richest possible logic – not just in terms of expressive power but mainly regarding its variety of temporal operators. We include the commonly known *until* operator from LTL and fixpoint quantifiers from μTL. We also allow a stengthened *until* operator $\mathtt{U}^\alpha$ from *Dynamic LTL* which – roughly speaking – asserts that the until must be satisfied at the end of a word in $L(\alpha)$ [6]. We include an *and-then* operator $\diamond\!\!\rightarrow$ which realises concatenation of a semi-extended regular expression with a temporal formula. It occurs in PSL in a slightly different way where the regular expression and the temporal formula are required to overlap in one state. This can, however, easily be defined using an $\mathtt{U}^\alpha$, and in order to cover as many cases as possible we let it denote the non-overlapping version here. Finally, inspired by the other PSL operator that links regular expressions with formulas we also include a *closure* operator $\mathtt{Cl}(\alpha)$. It asserts that at no position does a word look like it is not going to belong to $L(\alpha)$. In other words, all of its prefixes must be extendable to a finite word in $L(\alpha)$. This is slightly more general than the original PSL closure operator, see below for details.

Clearly, not all operators are needed in order to achieve expressive completeness w.r.t. ω-regular properties. For example, it is known that the specialised strengthened *until* operator $\mathtt{F}^\alpha$ is sufficient [6]. Sect. 3 shows what happens w.r.t. expressive power when certain operators are excluded from the logic. Because of lack of space here we only scrape upon the issue of succinctness. Clearly, since semi-extended regular expressions are only as expressive as ordinary ones there is no gain in expressive power from using them instead. However, some properties may be definable using shorter formulas. Note that Dynamic LTL for example, known to be PSPACE-complete [6], allows ordinary regular expressions only. Sect. 4 uses the expressiveness results to derive a general upper bound of det. exponential space for the satisfiability problem of this logic via a reduction to μTL. We also show that this is optimal obtaining the rule of thumb: regular expressions leave temporal logic PSPACE-complete, semi-extended ones make it EXPSPACE-complete.

2 Preliminaries

We start by recalling *semi-extended regular expressions*. Let $\mathcal{P} = \{q, p, \ldots\}$ be finite set of atomic propositions. *Propositional Logic* (PL) is the least set containing $\mathcal{P}$, the constants $\top, \bot$, and being closed under the operators $\vee, \wedge, \rightarrow$, etc., as well as the complementation operator $\overline{\cdot}$.

The satisfaction relation between symbols of the alphabet $2^{\mathcal{P}}$ and formulas of PL is the usual one: $a \models q$ iff $q \in a$; $a \models b_1 \wedge b_2$ iff $a \models b_1$ and $a \models b_2$; $a \models \overline{b}$ iff $a \not\models b$; etc.

We fix $\Sigma := 2^{\mathcal{P}}$ as an alphabet for the remainder of the paper. For a word $w \in \Sigma^*$ we use $|w|$ to denote its length, ϵ to denote the empty word, and w^i to denote the i-th symbol of w for $i \in \{0, \ldots, |w| - 1\}$. The set of all infinite words is Σ^ω. For a finite or infinite word w we write $w^{i..j}$ to denote the subword $w^i \ldots w^{j-1}$ of length $j - i$, and $w^{i..}$ to denote its suffix starting with w^i. We write $w \prec v$ to denote that w is a proper prefix of v.

For two languages L, L' with $L \subseteq \Sigma^*$ and L' either a language of finite or of infinite words, their composition LL' denotes the concatenation of all words in L and L' in the usual way. The n-fold iteration (in the finite case) is denoted L^n with $L^0 = \{\epsilon\}$. An important mapping from languages of finite words to those of infinite ones is the *closure*, defined for all $L \subseteq \Sigma^*$ as $Cl(L) := \{w \in \Sigma^\omega \mid \forall k \in \mathbb{N}\ \exists v \in \Sigma^*$ s.t. $w^{0..k}v \in L\}$.

Semi-extended regular expressions (SERE) are now built upon formulas of PL in the following way.

$$\alpha \ ::= \ b \mid \alpha \cup \alpha \mid \alpha \cap \alpha \mid \alpha ; \alpha \mid \alpha^*$$

where $b \in$ PL. Let $|\alpha| := |Sub(\alpha)|$ measure the size of an SERE with $Sub(\alpha)$ being the set of all subexpressions occurring in α where propositional formulas are atomic. Another important measure is the number of intersections occurring in α: $is(\alpha) := |\{\beta \in Sub(\alpha) \mid \beta$ is of the form $\beta_1 \cap \beta_2\}|$.

SERE define languages of finite words in the usual way. $L(b) := \{w \in \Sigma^* \mid |w| = 1, w_0 \models b\}$, $L(\alpha_1 \cup \alpha_2) := L(\alpha_1) \cup L(\alpha_2)$, $L(\alpha_1 \cap \alpha_2) := L(\alpha_1) \cap L(\alpha_2)$, $L(\alpha_1 ; \alpha_2) := L(\alpha_1)L(\alpha_2)$, and $L(\alpha^*) := \bigcup_{n \in \mathbb{N}} (L(\alpha))^n$.

The intersection operator $\cap$ allows us to define some usual ingredients of the syntax of regular expressions as abbreviations: $\emptyset := (q) \cap (\overline{q})$ for some $q \in \mathcal{P}$, and $\epsilon := (q)^* \cap (\overline{q})^*$. Equally, the symbolic encoding of symbols in PL enables a constant representation of the universal regular expression: $\Sigma^* = (\top)^*$.

An ordinary *regular expressions* (RE) is an SERE α with $is(\alpha) = 0$. We allow ϵ and $\emptyset$ as primitives in RE though. An ω*-regular expression* (ω-RE) γ is of the form $\bigcup_{i=1}^{n} \alpha_i ; \beta_i^\omega$ for some $n \in \mathbb{N}$ s.t. α_i, β_i are RE, and $\epsilon \notin L(\beta_i)$. Its language $L(\gamma)$ is defined in the usual way using union, concatenation and infinite iteration of finite languages.

Next we consider linear time temporal logic. Take $\mathcal{P}$ from above. Let $\mathcal{V} = \{X, Y, \ldots\}$ be a countably infinite set of monadic second-order variables. Formulas of full *Linear Time Temporal Logic with SERE* (TL^{SERE}) are given by the following grammar.

$$\varphi \ ::= \ q \mid \varphi \wedge \varphi \mid \neg \varphi \mid X \mid \bigcirc \varphi \mid \varphi \,\mathtt{U}\, \varphi \mid \varphi \,\mathtt{U}^\alpha\, \varphi \mid \mu X.\varphi \mid \alpha \diamond\!\!\rightarrow \varphi \mid \mathtt{Cl}(\alpha)$$

where $q \in \mathcal{P}$, $X \in \mathcal{V}$, and α is an SERE over $\mathcal{P}$. As usual, we require variables X to only occur under the scope of an even number of negation symbols within ψ if X is quantified by $\mu X.\psi$. If the α are restricted to ordinary regular expressions we obtain the logic TL^{RE}. We also use the usual abbreviated Boolean operators like $\vee$, and the LTL-operators $\mathtt{F}$ and $\mathtt{G}$ as well as the strengthened version $\mathtt{F}^{\alpha}\,\varphi := \mathtt{tt}\,\mathtt{U}^{\alpha}\,\varphi$.

The size of a formula is $|\varphi| := |Sub(\varphi)|$ where $Sub(\varphi)$ is the set of all subformulas of φ including $Sub(\alpha)$ for any SERE α occurring in φ. The reason for not counting them as atomic is the expressive power of the temporal operators, c.f. next section below. It is possible to encode a lot of temporal property in the SERE rather than the temporal operators themselves. Counting SERE as atomic would therefore lead to unnaturally small formulas. The measure $is(\alpha)$ can then be extended to formulas straight-forwardly: $is(\varphi) := |\{\beta \in Sub(\varphi) \mid \beta$ is an SERE of the form $\beta_1 \cap \beta_2\}$.

TL^{SERE} is interpreted over infinite words $w \in \Sigma^{\omega}$. The semantics assigns to each formula φ a language $L_{\rho}(\varphi) \subseteq \Sigma^{\omega}$ relative to some environment $\rho : \mathcal{V} \to 2^{\Sigma^{\omega}}$ which interprets free variables by languages of infinite words. We write $\rho[X \mapsto L]$ for the function that maps X to L and agrees with ρ on all other arguments.

$$
\begin{aligned}
L_{\rho}(q) &:= \{w \in \Sigma^{\omega} \mid q \in w^0\} \\
L_{\rho}(\varphi \wedge \psi) &:= L_{\rho}(\varphi) \cap L_{\rho}(\psi) \\
L_{\rho}(\neg\varphi) &:= \Sigma^{\omega} \setminus L_{\rho}(\varphi) \\
L_{\rho}(X) &:= \rho(X) \\
L_{\rho}(\bigcirc\varphi) &:= \{w \in \Sigma^{\omega} \mid w^{1..} \in L_{\rho}(\varphi)\} \\
L_{\rho}(\varphi\,\mathtt{U}\,\psi) &:= \{w \in \Sigma^{\omega} \mid \exists k \in \mathbb{N} \text{ s.t. } w^{k..} \in L_{\rho}(\psi) \text{ and } \forall j < k : \ w^{j..} \in L_{\rho}(\varphi)\} \\
L_{\rho}(\varphi\,\mathtt{U}^{\alpha}\,\psi) &:= \{w \in \Sigma^{\omega} \mid \exists k \in \mathbb{N} \text{ s.t. } w^{0..k+1} \in L(\alpha) \text{ and } w^{k..} \in L_{\rho}(\psi) \\
&\qquad\qquad \text{and } \forall j < k : \ w^{j..} \in L_{\rho}(\varphi)\} \\
L_{\rho}(\mu X.\varphi) &:= \bigcap\{L \subseteq \Sigma^{\omega} \mid L_{\rho[X \mapsto L]}(\varphi) \subseteq L\} \\
L_{\rho}(\alpha \diamond\!\!\rightarrow \varphi) &:= L(\alpha)L_{\rho}(\varphi) \\
L_{\rho}(\mathtt{Cl}(\alpha)) &:= Cl(L(\alpha))
\end{aligned}
$$

The formula $\mu X.\psi$ defines the least fixpoint of the monotone function which maps a language L of infinite words to the set of words that satisfy ψ under the assumption that X defines L. Due to negation closure and fixpoint duality we can also define greatest fixpoints of such maps via $\nu X.\psi := \neg\mu X.\neg\psi[\neg X/X]$, where the latter denotes the simultaneous substitution of $\neg X$ for every occurrence of X in ψ.

As usual, we write $\varphi \equiv \psi$ if for all environments ρ we have $L_{\rho}(\varphi) = L_{\rho}(\psi)$. We use the notation TL[..], listing certain operators, to denote syntactical fragments of TL^{SERE}, for instance $\text{TL}^{\text{SERE}} = \text{TL}[\bigcirc, \mathtt{U}, \mathtt{U}^{\alpha}, \mu, \diamond\!\!\rightarrow, \mathtt{Cl}]$. We do not list variables explicitly because variables without quantifiers or vice-versa are meaningless. Also, we always assume that these fragments contain atomic propositions and the

Boolean operators. For example, LTL is TL[$\bigcirc$, $\mathtt{U}$], and it has genuine fragments TL[$\bigcirc$, $\mathtt{F}$], TL[$\mathtt{F}$], TL[$\bigcirc$] [4]. The linear time μ-calculus μTL is TL[$\bigcirc, \mu$], and PSL is TL[$\mathtt{U}, \mathtt{F}^\alpha, \mathtt{Cl}$]: the PSL formula consisting of an SERE α is equivalent to the TL$^{\text{SERE}}$ formula $(\mathtt{F}^\alpha\, \mathtt{tt}) \vee \mathtt{Cl}(\alpha)$, where $\mathtt{tt} := q \vee \neg q$ for some $q \in \mathcal{P}$. If we restrict the use of semi-extended regular expressions in a fragment to ordinary regular expressions only then we denote this by TL$^{\text{RE}}$[..].

Some of the results regarding expressive power are obtained using automata on finite words, Büchi automata, and translations from SERE into those. We therefore quickly recall the automata-theoretic foundations. A *nondeterministic finite automaton* (NFA) is a tuple $\mathcal{A} = (Q, \mathcal{P}, q_0, \delta, F)$ with Q a finite set of states, $q_0 \in Q$ a starting state, and $F \subseteq Q$ a set of final states. Its alphabet is $\Sigma = 2^{\mathcal{P}}$, and its transition relation δ is a finite subset of $Q \times \text{PL} \times Q$. We define $|\mathcal{A}| := |\delta|$ as the size of the NFA. Note that these NFA use symbolic representations of symbols in their transitions in order to avoid unnecessary exponential blow-ups in their sizes. A run of $\mathcal{A}$ on a finite word $w \in \Sigma^*$ is a non-empty sequence $q_0, w^0, q_1, w^1, \ldots, q_n$ s.t. for all $i = 0, \ldots, n-1$ there is a b s.t. $w^i \models b$ and $(q_i, b, q_{i+1}) \in \delta$. It is accepting iff $q_n \in F$.

Translating RE into NFA is a standard exercise. It is not hard to see that the usual uniform translation applies to the case of symbolically represented transition relations as well. It is also relatively easy to see that it can be extended to SERE. The translation of the SERE $\alpha \cap \beta$ simply uses the well-known product construction on NFA. Clearly, the size of the product NFA is quadratic in the original sizes, and iterating this leads to an exponential translation.

Proposition 1. *For every SERE α there is an NFA $\mathcal{A}_\alpha$ s.t. $L(\mathcal{A}_\alpha) = L(\alpha)$ and $|\mathcal{A}_\alpha| = O(2^{|\alpha| \cdot (is(\alpha)+1)})$.*

To see that an exponential blow-up can occur simply consider the recursively defined RE α_n with $\alpha_0 := q$ and $\alpha_{n+1} := \alpha_n; \neg q; \alpha_n; q$.

A *nondeterministic Büchi automaton* (NBA) is syntactically defined like an NFA. It runs on infinite words $w \in \Sigma^\omega$ via $q_0, w^0, q_1, w^1, q_2, \ldots$ s.t. for all $i \in \mathbb{N}$ there is a $b \in \text{PL}$ s.t. $w^i \models b$ and $(q_i, b, q_{i+1}) \in \delta$. It is accepting iff there are infinitely many i s.t. $q_i \in F$. We use NBA to model the closure of languages L of finite words. This is particularly easy if L is given by an NFA.

Lemma 1. *For every NFA $\mathcal{A}$ there is an NBA $\mathcal{A}_{cl}$ s.t. $L(\mathcal{A}_{cl}) = Cl(L(\mathcal{A}))$, and $|\mathcal{A}_{cl}| \leq |\mathcal{A}|$.*

Proof. Let $\mathcal{A} = (Q, \mathcal{P}, q_0, \delta, F)$ and $Q' \subseteq Q$ consist of all states which are reachable from q_0 and productive, i.e. from each of them a final state is reachable. Then define $\mathcal{A}_{cl} := (Q', \mathcal{P}, q_0, \delta, Q')$. We have $L(\mathcal{A}_{cl}) \subseteq Cl(L(\mathcal{A}))$ because runs in $\mathcal{A}_{cl}$ only visit states from with final states in $\mathcal{A}$ are reachable.

For the converse direction suppose $w \in Cl(L(\mathcal{A}))$, i.e. for every $k \in \mathbb{N}$ there is a $v \in \Sigma^*$ and an accepting run of $\mathcal{A}$ on $w^{0..k}v$. Clearly, all these runs stay in Q'. Furthermore, they can be arranged to form an infinite but finitely branching tree, and König's Lemma yields an accepting run of $\mathcal{A}_{cl}$ on w. □

3 Expressive Power and Succinctness

For two fragments $\mathcal{L}_1$ and $\mathcal{L}_2$ of TL^{SERE} we write $\mathcal{L}_1 \leq_{f(n,k)} \mathcal{L}_2$ if for every $\varphi \in \mathcal{L}_1$ there is a $\psi \in \mathcal{L}_2$ s.t. $\varphi \equiv \psi$ and $|\psi| \leq f(|\varphi|, is(\varphi))$. This measures relative expressive power and possible succinctness. We write $\mathcal{L}_1 \equiv_{f(n,k)} \mathcal{L}_2$ if $\mathcal{L}_1 \leq_{f(n,k)} \mathcal{L}_2$ and $\mathcal{L}_2 \leq_{f(n,k)} \mathcal{L}_1$. In case the succinctness issue is unimportant we simply denote expressive subsumption and equivalence using $\leq$ and $\equiv$.

It is well-known that $\text{TL}[\bigcirc, \mu]$, the linear time μ-calculus, has the same expressive power as Monadic Second-Order Logic on infinite words, or equivalently, ω-RE. It is also known that $\text{TL}^{\text{RE}}[\mathtt{F}^{\alpha}]$ and, thus, $\text{TL}[\mathtt{F}^{\alpha}]$ is of the same expressiveness [6]. We start by showing that the non-overlapping *and-then* operator suffices for expressive completeness too.

Theorem 1. $\text{TL}[\mathtt{F}^{\alpha}] \leq_{O(n^{k+1})} \text{TL}[\diamond\!\!\to]$.

Proof. We use a function $cut : \text{SERE} \to 2^{\text{SERE}\times\text{PL}}$ that decomposes an SERE via $cut(\alpha) = \{(\beta_1, b_1), \ldots, (\beta_n, b_n)\}$ only if $L(\alpha) = \bigcup_{i=1}^{n} L(\beta_i)L(b_i)$. This can be recursively defined as

$$\begin{aligned}
cut(b) &:= \{(\epsilon, b)\}\\
cut(\alpha_1 \cup \alpha_2) &:= cut(\alpha_1) \cup cut(\alpha_2)\\
cut(\alpha_1 \cap \alpha_2) &:= \{(\beta_1 \cap \beta_2, b_1 \wedge b_2) \mid (\beta_i, b_i) \in cut(\alpha_i) \text{ for } i = 1, 2\}\\
cut(\alpha_1; \alpha_2) &:= \{(\alpha_1; \beta, b) \mid (\beta, b) \in cut(\alpha_2)\} \cup \begin{cases} cut(\alpha_1) & \text{, if } \epsilon \in L(\alpha_2)\\ \emptyset & \text{, o.w.}\end{cases}\\
cut(\alpha^*) &:= \{(\alpha^*; \beta, b) \mid (\beta, b) \in cut(\alpha)\}
\end{aligned}$$

Note that $|cut(\alpha)| \leq O(|\alpha|^{is(\alpha)+1})$. We can now use this decomposition to translate a formula of the form $\mathtt{F}^{\alpha}\,\varphi$ with an overlap between α and φ into a non-overlapping one: $\mathtt{F}^{\alpha}\,\varphi \equiv \bigvee_{(\beta,b)\in cut(\alpha)} \beta \diamond\!\!\to (b \wedge \varphi)$. □

The same reduction can be carried out from $\text{TL}^{\text{RE}}[\mathtt{F}^{\alpha}]$ to $\text{TL}^{\text{RE}}[\diamond\!\!\to]$, and it would be polynomial because we would have $k = 0$.

The expressive power of the closure operator is a natural concern. It turns out to be weaker than $\mathtt{U}^{\alpha}$ or $\diamond\!\!\to$ for instance. We use an invariant technique on automata structure to show this. An NBA $\mathcal{A} = (Q, \mathcal{P}, q_0, \delta, F)$ is called $\exists\forall$-accepting if there is no $q \in F$ from which a $q' \notin F$ is reachable. Runs of these automata accept iff they eventually only visit final states. Note that this is not the same as a co-Büchi automaton since it is a syntactical restriction, but it is a special case of a *weak* NBA. An important observation is that the NBA $\mathcal{A}_{cl}$ as constructed in the proof of Lemma 1 are actually $\exists\forall$-accepting.

Lemma 2. *Every language definable in* $\text{TL}[\mathtt{Cl}]$ *can be recognised by an* $\exists\forall$*-accepting NBA.*

Proof. Let $\psi \in \text{TL}[\mathtt{Cl}]$. Clearly, ψ is a Boolean combination of formulas of the form $\mathtt{Cl}(\alpha)$. Using deMorgan laws it is possible to transform ψ into a positive Boolean combination of formulas of the form $\mathtt{Cl}(\alpha)$ and $\neg\mathtt{Cl}(\beta)$.

Now note that $w \not\models \mathtt{Cl}(\beta)$ iff there is a $v \prec w$ s.t. for all $v' \in \Sigma^*$: $vv' \notin L(\beta)$. Let $\mathcal{A}_\beta$ be the NFA that recognises exactly the models of β, c.f. Prop. 1. It can be determinised and complemented into a DFA $\mathcal{A}_{\overline{\beta}} = (Q, \mathcal{P}, q_0, \delta, F)$. Let $Q' := \{q \in Q \mid \forall q' \in Q$: if q' is reachable from q then $q' \in F\}$. Note that $Q' \subseteq F$. Now consider the deterministic NBA $\mathcal{A}'_{\overline{\beta}} = (Q, \mathcal{P}, q_0, \delta, Q')$. It accepts a word w iff the unique run on it exhibits a prefix v which takes the NBA from q_0 to a state $q \in F$. Hence, $v \notin L(\beta)$. Furthermore, $vv' \notin L(\beta)$ for any v' because $\mathcal{A}'_{\overline{\beta}}$ is deterministic and only states in F are reachable from q.

This shows that ψ is equivalent to a positive Boolean combination of languages recognisable by $\exists\forall$-accepting NBA. Given two $\exists\forall$-accepting NBA recognising L_1 and L_2 it is easy to construct $\exists\forall$-accepting NBA for the languages $L_1 \cup L_2$ and $L_1 \cap L_2$ using the standard union and product constructions. Hence, $L(\psi)$ can be recognised by an $\exists\forall$-accepting NBA. ☐

Lemma 3. TL[$\bigcirc$, $\mathtt{Cl}$] $\equiv_{O(n)}$ TL[$\mathtt{Cl}$].

Proof. The $\geq$ part is of course trivial. For the $\leq$ part first note that because of the equivalences $\bigcirc\neg\psi \equiv \neg\bigcirc\psi$ and $\bigcirc(\psi_1 \wedge \psi_2) \equiv \bigcirc\psi_1 \wedge \bigcirc\psi_2$ every TL[$\bigcirc$, $\mathtt{Cl}$] formula φ can be transformed into a Boolean combination of atomic formulas of the form $\bigcirc^k q$ or $\bigcirc^k \mathtt{Cl}(\alpha)$ for some SERE α and some $k \in \mathbb{N}$. Using the equivalences $q \equiv \mathtt{Cl}(q; \top^*)$ and $\bigcirc\mathtt{Cl}(\alpha) \equiv \mathtt{Cl}(\top; \alpha)$ it is then possible to eliminate all occurrences of the $\bigcirc$ operators. The resulting formula is clearly of linear size. ☐

The following looks like an immediate consequence of this but it needs the observation that the $\bigcirc$ operator commutes with an $\mathtt{U}$. The same holds for the $\mathtt{F}$ operator.

Lemma 4. TL[$\bigcirc$, $\mathtt{U}$, $\mathtt{Cl}$] $\equiv_{O(n)}$ TL[$\mathtt{U}$, $\mathtt{Cl}$].

Proof. As the proof of Lemma 3 but also using the equivalence $\bigcirc(\varphi \, \mathtt{U} \, \psi) \equiv (\bigcirc\varphi) \, \mathtt{U} \, (\bigcirc\psi)$ in order to push the $\bigcirc$ operators down to atomic propositions and closure operators where they can be eliminated. ☐

Corollary 1. TL[$\bigcirc$, $\mathtt{F}$, $\mathtt{Cl}$] $\equiv_{O(n)}$ TL[$\mathtt{F}$, $\mathtt{Cl}$].

Theorem 2. TL[$\mathtt{Cl}$] *is $\leq$-incomparable to both* TL[$\bigcirc$, $\mathtt{U}$] *and* TL[$\mathtt{F}$].

Proof. Since TL[$\mathtt{F}$] $\leq$ TL[$\bigcirc$, $\mathtt{U}$] it suffices to show that there is a TL[$\mathtt{Cl}$] property which is not definable in TL[$\bigcirc$, $\mathtt{U}$], and a TL[$\mathtt{F}$] property which is not definable in TL[$\mathtt{Cl}$].

It is well-known that TL[$\bigcirc$, $\mathtt{U}$], aka LTL, cannot express "q holds in every even moment". This, however, can easily be expressed by $\mathtt{Cl}((q; \top)^*)$. For the converse direction take the TL[$\mathtt{F}$] formula $\mathtt{G}\,\mathtt{F}\, q$. According to Lemma 2 its language would be recognisable by an $\exists\forall$-accepting NBA $\mathcal{A} = (Q, \mathcal{P}, q_0, \delta, F)$ if it was definable in TL[$\mathtt{Cl}$]. Let $n := |Q|$. Consider the word $w := (\{q\}\emptyset^{n+1})^\omega$. Since $w \in L(\mathtt{G}\,\mathtt{F}\, q)$ there is an accepting run $q_0, w^0, q_1, w^1, \ldots$ of $\mathcal{A}$ on w. Since $\mathcal{A}$ is $\exists\forall$-accepting there is a k s.t. $q_i \in F$ for all $i \geq k$. Then there will be $j > i \geq k$ s.t. $q_i = q_j$ and $w^h = \emptyset$ for all $h = i, \ldots, j-1$. This gives us an accepting run on a word of the form $(\{q\}\emptyset)^*\emptyset^\omega$ which should not be accepted. ☐

Corollary 2. $\mathrm{TL}[\mathtt{F}, \mathtt{Cl}] \not\leq \mathrm{TL}[\bigcirc, \mathtt{U}]$.

The converse direction is still open. It remains to be seen whether or not every LTL-definable property can also be expressed in $\mathrm{TL}[\mathtt{F}, \mathtt{Cl}]$. Note for example that, for atomic propositions p and q, we have $p \,\mathtt{U}\, q \equiv \mathtt{F}\, q \wedge \mathtt{Cl}(p^*; q; \top^*)$. However, this does not extend easily to arbitrary formulas of the form $\varphi \,\mathtt{U}\, \psi$.

Clearly, adding the operator $\diamond\!\!\rightarrow$ or $\mathtt{F}^\alpha$ to the closure operator yields expressive completeness since these alone are sufficient for that. This poses the question of what happens when the closure operator is combined with LTL operators which are not enough to achieve ω-regular expressiveness. One answer is obtained easily from Thm. 2: $\mathrm{TL}[\bigcirc, \mathtt{U}]$ is strictly less expressive than $\mathrm{TL}[\mathtt{U}, \mathtt{Cl}]$. On the other hand, we suspect that $\mathrm{TL}[\mathtt{U}, \mathtt{Cl}]$ is also strictly contained in $\mathrm{TL}^{\mathrm{SERE}}$. In particular, we believe that the property "q holds infinitely often in even moments" is not expressible in $\mathrm{TL}[\mathtt{U}, \mathtt{Cl}]$.

It is known that each operator present in $\mathrm{TL}^{\mathrm{SERE}}$ does not exceed its expressive power beyond ω-regularity. Therefore it is fair to assume that $\mathrm{TL}^{\mathrm{SERE}}$ is only as expressive as μTL. We present a direct and conceptionally simple translation from $\mathrm{TL}^{\mathrm{SERE}}$ to $\mathrm{TL}[\bigcirc, \mu]$ which forms the basis for the complexity analysis in the next section. The following lemmas prepare for the not so straight-forward cases in that translation.

Lemma 5. *For every NBA $\mathcal{A}$ there is a closed* $\mathrm{TL}[\bigcirc, \mu]$ *formula $\varphi_{\mathcal{A}}$ s.t. $L(\varphi_{\mathcal{A}}) = L(\mathcal{A})$ and $|\varphi_{\mathcal{A}}| = O(|\mathcal{A}|)$.*

Proof. Let $\mathcal{A} = (Q, \mathcal{P}, q_0, \delta, F)$. W.l.o.g. we assume $Q = \{0, \dots, n\}$ for some $n \in \mathbb{N}$, $q_0 = 0$, and $F = \{0, \dots, m\}$ for some $m \leq n$. Note that the starting state can always be assumed to be final by adding a copy of the starting state which is not reachable from any other state.

The construction of $\varphi_{\mathcal{A}}$ uses n monadic second-order variables X_i, $i \in Q$, each X_i representing the moments of a run in which $\mathcal{A}$ is in state i. To understand the construction best we introduce an auxiliary device of an NBA $\mathcal{A}_\rho$ with a partial function $\rho : Q \to 2^{\Sigma^\omega}$ acting as an oracle.[1] $\mathcal{A}_\rho$ immediately accepts the language $\rho(i)$ upon entrance of state i when $\rho(i)$ is defined.

For every $i \in Q$ in the order $i = n, \dots, 0$ we construct a formula $\psi_i(X_0, \dots, X_{i-1})$ s.t. $L_\rho(\psi_i) = \{w \mid \mathcal{A}_{\rho'} \text{ accepts } w \text{ starting in state } i\}$ where $\rho'(j) := \rho(X_j)$ for all $j < i$.

$$\psi_i \;:=\; \sigma_i X_i. \bigvee_{(i,b,j) \in \delta} b \wedge \bigcirc \begin{cases} \psi_j & \text{, if } j > i \\ X_j & \text{, o.w.} \end{cases}$$

where $\sigma_i := \mu$ if $i > m$ and ν otherwise. The correctness claim above is straight-forwardly verified by induction on i. Also note that ψ_i is well-defined, and ψ_0 is a closed formula. Its correctness claim refers to the oracle NBA $\mathcal{A}_{\rho'}$ starting in the initial state 0 and not using the oracle at all. Hence, we have $L_\rho(\psi_0) = L(\mathcal{A})$ for any ρ, and therefore define $\varphi_{\mathcal{A}} := \psi_0$. Note that its size is linear in the size of $\mathcal{A}$. □

[1] Since oracles are for automata what environments are for the semantics function – an interpreter of open parts – we use the same symbol ρ here.

Lemma 6. *For every NFA $\mathcal{A}$ and every $\mathrm{TL}[\bigcirc, \mu]$ formula χ there is a $\mathrm{TL}[\bigcirc, \mu]$ formula $\varphi_{\mathcal{A},\chi}$ s.t. for any environment ρ: $L_\rho(\varphi_{\mathcal{A},\chi}) = L(\mathcal{A})L_\rho(\chi)$ and $|\varphi_{\mathcal{A},\chi}| = O(|\mathcal{A}| + |\chi|)$.*

Proof. Similar to the previous construction. Let $\mathcal{A} = (Q, \mathcal{P}, q_0, \delta, F)$ with $Q = \{0, \ldots, n\}$. We construct for every $i = n, \ldots, 0$ a $\mathrm{TL}[\bigcirc, \mu]$ formula $\psi_i(X_0, \ldots, X_{i-1})$ s.t. $L_\rho(\psi_i) = L_i L_\rho(\chi)$ where $L_i \subseteq \Sigma^*$ consists of all words that are accepted by $\mathcal{A}$ when starting in state i under the assumption that upon entering state $j < i$, it immediately accepts the language $\rho(X_j)$.

$$\psi_i \;:=\; \mu X_i. \bigvee_{(i,b,j)\in\delta} \chi_i \vee (b \wedge \bigcirc \begin{cases} \psi_j & \text{, if } j > i \\ X_j & \text{, o.w.} \end{cases})$$

where $\chi_i := \chi$ if $i \in F$, and $\mathtt{ff}$ otherwise. Note that the language defined by an NFA is given as the simultaneous least fixpoint of the languages recognised by each state. Hence, only μ quantifiers are needed here as opposed to the NBA case above where the language is given as a nested greatest/least fixpoint.

Again, define $\varphi_{\mathcal{A},\chi}$ as ψ_0. The correctness of this construction w.r.t. to the specification above can straight-forwardly be proved by induction on i. Also, the claim on its size is easily checked to be true. □

Lemma 7. *For every $\mathrm{TL}[\bigcirc, \mu]$ formulas φ_1, φ_2 and every NFA $\mathcal{A}$ there is a $\mathrm{TL}[\bigcirc, \mu]$ formula $\varphi_{\mathcal{A},\varphi_1,\varphi_2}$ s.t. $|\varphi_{\mathcal{A},\varphi_1,\varphi_2}| = O(|\mathcal{A} + |\varphi_1| + |\varphi_2|)$ and for all environments ρ: $L_\rho(\varphi_{\mathcal{A},\varphi_1,\varphi_2}) = \{w \in \Sigma^\omega \mid \exists k \in \mathbb{N}$ s.t. $w^{0..k+1} \in L(\mathcal{A})$ and $w^{k..} \in L_\rho(\varphi_2)$ and $\forall j < k : w^{j..} \in L_\rho(\varphi_1)\}$.*

Proof. This is done in very much the same way as in the proof of Lemma 6. There are only two minor differences. (1) Because of the overlap between the part accepted by $\mathcal{A}$ and the part satisfying φ_2 we need to assert φ_2 not after having been in a final state but before entering one. (2) In every step that $\mathcal{A}$ does without entering a final state we need to require φ_1 to hold.

Again, let $\mathcal{A} = (Q, \mathcal{P}, q_0, \delta, F)$ with $Q = \{0, \ldots, n\}$. Define for each $i \in Q$:

$$\psi_i \;:=\; \mu X_i. \bigvee_{(i,b,j)\in\delta} b \wedge \big(\chi_i \vee (\varphi_1 \wedge \bigcirc \left\{ \begin{array}{l} \psi_j \text{ , if } j > i \\ X_j \text{ , o.w.} \end{array} \right\})\big)$$

where $\chi_i := \varphi_2$ if $j \in F$, and $\chi_i := \mathtt{ff}$ otherwise. Note that the ability to prove φ_2 is linked to the fact whether or not j belongs to F because of the aforementioned overlap. Again, define $\varphi_{\mathcal{A},\varphi_1,\varphi_2} := \psi_0$ to finish the claim. □

Theorem 3. $\mathrm{TL}^{\text{SERE}} \leq_{O(2^{n\cdot(k+1)})} \mathrm{TL}[\bigcirc, \mu]$.

Proof. By induction on the structure of the formula. The claim is trivially true for atomic propositions and variables. Boolean operators, the temporal $\bigcirc$, and fixpoint quantifiers μ are translated uniformly. An U operator can be replaced by a least fixpoint formula. The remaining cases are the interesting ones.

Suppose $\varphi = \alpha \diamond\!\!\to \psi$. By hypothesis, there is a $\psi' \in \mathrm{TL}[\bigcirc, \mu]$ s.t. $\psi' \equiv \psi$ and $|\psi'| \leq O(2^{|\psi|\cdot(is(\psi)+1)})$. According to Prop. 1 there is an NFA $\mathcal{A}_\alpha$ s.t.

$L(\mathcal{A}_\alpha) = L(\alpha)$ and $|\mathcal{A}_\alpha| \leq O(2^{|\alpha|\cdot(is(\alpha)+1)})$. According to Lemma 6 there is a $\mathrm{TL}[\bigcirc, \mu]$ formula φ' s.t. $L_\rho(\varphi') = L(\mathcal{A}_\alpha)L_\rho(\psi')$ under any ρ. Thus, $L_\rho(\varphi') = L_\rho(\varphi)$. Furthermore, $|\varphi'| \leq O(2^{|\psi|\cdot(is(\psi)+1)} + 2^{|\alpha|\cdot(is(\alpha)+1)}) \leq O(2^{|\varphi|\cdot(is(\varphi)+1)})$.

The cases of $\varphi = \psi_1 \, \mathtt{U}^\alpha \, \psi_2$ and $\varphi = \mathtt{Cl}(\alpha)$ are done in the same way but using Lemmas 7, 1, and 5 instead. $\square$

4 The Complexity of $\mathrm{TL}^{\mathrm{SERE}}$ and Its Fragments

We can now easily obtain an upper bound on the complexity of the satisfiability problem for $\mathrm{TL}^{\mathrm{SERE}}$ from Thm. 3. It composes the exponential reduction with the known PSPACE upper bound for μTL, found many times in different ways.

Proposition 2. [9,13] *Satisfiability in* $\mathrm{TL}[\bigcirc, \mu]$ *is PSPACE-complete.*

Corollary 3. *Satisfiability in (a)* $\mathrm{TL}^{\mathrm{SERE}}$ *is in EXPSPACE, and (b)* $\mathrm{TL}^{\mathrm{RE}}$ *is in PSPACE.*

Part (b) is not necessarily surprising. The satisfiability problem of all the logics considered in the literature with some of the operators of $\mathrm{TL}^{\mathrm{RE}}$ can be decided in PSPACE, c.f. Prop. 2 and [9,6,3]. There is no need to assume that the combination of these operators should exceed the PSPACE bound.

Part (a) entails that PSL can be decided in deterministic exponential space. There is a translation of PSL into NBA of doubly exponential size – but measuring the size of regular expressions as their syntactical length – which goes via weak alternating Büchi automata [3]. It does not mention SERE, however, this can easily be incorporated, see above. Since the emptiness problem for NBA is known to be in NLOGSPACE, part (a) follows for PSL also from that translation and Savitch's Theorem [8].

A surprising fact is part (a) in conjunction with the emptiness problem for SERE. An immediate consequence of Prop. 1 is a PSPACE upper bound on the emptiness problem (i.e. satisfiability) of semi-extended regular expressions. However, combining that with the PSPACE decidable operators from temporal logic raises the space complexity by one exponential. This is also optimal. We will prove a lower complexity bound of deterministic exponential space by a reduction from the following 2^n-tiling problem, known to be EXPSPACE-hard [12]. Given an $n \in \mathbb{N}$ and a finite set $T = \{1, \ldots, m\}$ called tiles, and two binary relations $M_h, M_v \subseteq T \times T$ (for horizontal and vertical matching), decide whether or not there is a function $\tau : \{0, \ldots, 2^n - 1\} \times \mathbb{N} \to T$ s.t.

- $\forall i \in \{0, \ldots, 2^n - 2\}, \forall j \in \mathbb{N} : (\tau(i,j), \tau(i+1,j)) \in M_h$
- $\forall i \in \{0, \ldots, 2^n - 1\}, \forall j \in \mathbb{N} : (\tau(i,j), \tau(i,j+1)) \in M_v$

Note that such a function tiles the infinite corridor of width 2^n s.t. adjacent tiles match in the horizontal and vertical relations. In the following we will write iM_x for $\{j \mid (i,j) \in M_x\}$ where $x \in \{h, v\}$.

Theorem 4. *Satisfiability in* $\mathrm{TL}[\bigcirc, \mathtt{F}, \mathtt{F}^\alpha]$ *is EXPSPACE-hard.*

Proof. By a reduction from the 2^n-tiling problem. Suppose n and a set T of tiles are given. We can easily regard $\{0, \ldots, 2^n - 1\} \times \mathbb{N}$ as an infinite word in which the cell (i, j) is represented by the $(j \cdot 2^n + i)$-th symbol in that word. We will use atomic propositions $t_1, \ldots, t_m$ to model the tiles and $c_0, \ldots, c_{n-1}$ to enumerate the positions in the word modulo 2^n.

First of all we need a counter formula that axiomatises the enumeration of the symbols. It asserts that the cell at (i, j) is labeled with those propositions which represent set bits in the binary encoding of i. We use the fact that in binary increment a bit becomes set iff its current value correctly indicates whether the bit below does not decrease its value.

$$\varphi_{count} \;:=\; \chi_0 \;\wedge\; \mathtt{G}\Big(\; (c_0 \leftrightarrow \bigcirc\neg c_0) \;\wedge\; \bigwedge_{i=1}^{n-1} \bigcirc c_i \leftrightarrow \big(\; c_i \leftrightarrow (c_{i-1} \rightarrow \bigcirc c_{i-1})\;\big)\;\Big)$$

where $\chi_0 := \bigwedge_{i=0}^{n-1} \neg c_i$ marks the beginning of each row. We also need to say that each cell contains exactly one tile.

$$\varphi_{tile} \;:=\; \mathtt{G}\,(\bigvee_{i=1}^{m} t_i \wedge \bigwedge_{j \neq i} \neg t_j)$$

In order to compare vertically adjacent tiles we create an SERE α_n s.t. $L(\alpha_n) = \{w \mid |w| = 2^n + 1\}$. This is particularly easy on models which also satisfy φ_{count}. It suffices to require all counter bits to have the same value in the first and last symbol of each word, and to contain at most one symbol which satisfies χ_0 unless this is what it starts with.[2]

$$\alpha_n \;:=\; \top; (\neg\chi_0)^*; \chi_0; (\neg\chi_0)^* \;\cap\; \bigcap_{i=0}^{n-1} (c_i; \top^*; c_i) \cup (\neg c_i; \top^*; \neg c_i)$$

At last we need to axiomatise the two relations modelling the matching of tiles.

$$\begin{aligned}\varphi_h \;&:=\; \mathtt{G}\,(\;\bigwedge_{i=1}^{m} t_i \;\rightarrow\; \bigcirc(\chi_0 \vee \bigvee_{j \in iM_h} t_j)\;)\\ \varphi_v \;&:=\; \mathtt{G}\,(\;\bigwedge_{i=1}^{m} t_i \;\rightarrow\; \mathtt{F}^{\alpha_n}(\;\bigvee_{j \in iM_v} t_j)\;)\end{aligned}$$

Then the given tiling problem has a positive instance iff $\varphi := \varphi_{count} \wedge \varphi_{tile} \wedge \varphi_h \wedge \varphi_v$ is satisfiable. Note that φ can be constructed in logarithmic space from the tiling problem and n. $\square$

This is not optimal in terms of the operators that are being used. They are just chosen to make the presentation easiest. Now note that the only temporal

[2] Given our definition of size of an SERE we could also recursively define $\alpha_n := \alpha_{n-1}; \alpha_{n-1}$, etc. However, the definition we use here also shows EXPSPACE-hardness when the size is measured as the syntactical length.

operators occurring in that formula are $\bigcirc$, $\mathtt{G}$, and $\mathtt{F}^{\alpha}$. Hence, in order to reduce the number of operators used, and to strengthen the hardness result we simply need to express these operators in terms of others.

Corollary 4. *Satisfiability in* $\mathrm{TL}[\mathtt{F}^{\alpha}]$ *is EXPSPACE-hard.*

Proof. Because of $\bigcirc\varphi \equiv \mathtt{F}^{\top;\top}\,\varphi$ and $\mathtt{G}\,\varphi \equiv \neg(\mathtt{F}^{\top^*}\,\neg\varphi)$. □

Unfortunately, Thm. 1 does not immediately yield a lower bound for the fragment built upon the non-overlapping *and-then* operator as well. Remember that the translation from $\mathrm{TL}[\mathtt{F}^{\alpha}]$ to $\mathrm{TL}[\diamond\!\!\!\to]$ is exponential in the number of intersection operators. Nevertheless, the result can be restored.

Theorem 5. *Satisfiability in* $\mathrm{TL}[\diamond\!\!\!\to]$ *is EXPSPACE-hard.*

Proof. Consider the SERE

$$\begin{aligned}
\alpha'_n \;:=\; & \big(\; c^*_{n-1}; (\neg c_{n-1})^*; c^*_{n-1} \cup (\neg c_{n-1})^*; c^*_{n-1}; (\neg c_{n-1})^* \;\big) \\
& \cap \Big(\; \chi_0; \top^*; \big(\bigwedge_{i=0}^{n} c_i\big) \cup \\
& \qquad \bigcup_{i=0}^{n-1} c_i; \top^*; \neg c_i \cap \big(\bigcap_{j>i}(c_j; \top^*; c_j) \cup (\neg c_j; \top^*; \neg c_j)\big) \cap \big(\bigcap_{j<i} \neg c_j; \top^*; c_j\big) \;\Big)
\end{aligned}$$

It asserts that there is a bit which is now 1 and 0 in the end, all higher bits have the same value now and then, and all lower bits change from 0 to 1. There is the special case of the counter being at value 0 now and at value $2^n - 1$ at the end. Also, we make sure that the SERE is only fulfilled by minimal words. Here this simply means that the highest bit changes its value at most twice. Then α'_n is fulfilled exactly by subwords of length 2^n in the context of φ_{count} from the proof of Thm. 4. Hence, we can prove this claim in exactly the same way but using $\alpha'_n \diamond\!\!\!\to \psi$ instead of $\mathtt{F}^{\alpha_n}\,\psi$. Furthermore, $\bigcirc\varphi \equiv \top \diamond\!\!\!\to \varphi$ and $\mathtt{G}\,\varphi \equiv \neg(\top^* \diamond\!\!\!\to \neg\varphi)$. □

One question arises naturally: does the closure operator alone suffice to gain EXPSPACE-hardness? The proof of the following theorem reformulates the reduction in the proof of Thm. 4 using the closure operator. However, the vertical matching relation in the tiling requires a single occurrence of a $\mathtt{G}$. Without this $\mathtt{G}$ operator we can only establish PSPACE-hardness. This does not follow from PSPACE-hardness of LTL, c.f. Thm. 2 above.

Theorem 6. *Satisfiability in* $\mathrm{TL}[\mathtt{F}, \mathtt{Cl}]$ *is EXPSPACE-hard.*

Proof. We simply replace the definitions of the four conjuncts in the constructed formula φ from the proof of Thm. 4. The first two are relatively easy to transform.

$$\varphi_{count} \;:=\; \chi_0 \wedge \mathtt{Cl}((\alpha'_n)^*) \qquad\qquad \varphi_{tile} \;:=\; \mathtt{Cl}(\;(\bigvee_{i=1}^{m} t_i \wedge \bigwedge_{j\neq i} \neg t_j)^*\;)$$

where α'_n is taken from the proof of Thm. 5 above.

Now consider the RE $\beta_h := \bigcup_{i=1}^{m} t_i; (\chi_0 \cup \bigcup_{j \in iM_h} t_j)$. It describes all 2-symbol words that match horizontally including the case of the right one belonging to the next row already. Then β_h^* describes all words s.t. between each even position and its right neighbour there is a correct matching. Hence, we can specify the correct horizontal tiling as follows.

$$\varphi_h \;:=\; \mathtt{Cl}(\; \beta_h^*; \top \cap \top; \beta_h^* \;)$$

The same trick cannot be used to axiomatise the tiling in vertical direction because we would have to list conjuncts for each position in the first row, i.e. exponentially many. This can easily be overcome using the $\mathtt{G}$ operator.

$$\varphi_v \;:=\; \mathtt{G}\big(\mathtt{Cl}(\; \beta_v^* \;)\big) \qquad \text{where} \quad \beta_v \;:=\; \alpha_n \cap (\bigcup_{i=1}^{m} t_i; \top^*; (\bigcup_{j \in iM_v} t_j))$$

with α_n from the proof of Thm. 4. This includes some redundancy since the formula $\mathtt{Cl}(\beta_v^*)$ – when interpreted in the cell (i,j) – checks for correct matchings between (i,j) and $(i,j+1)$, between $(i+1 \mod 2^n, j+1)$ and $(i+1 \mod 2^n, j+2)$, etc. The latter matchings are also imposed by the $\mathtt{G}$ operator. □

Theorem 7. *Satisfiability in* TL$^{\text{RE}}$[$\mathtt{Cl}$] *is PSPACE-hard.*

Proof. By reduction from the tiling problem for $\{0,\ldots,n-1\} \times \mathbb{N}$, otherwise defined in the same way as the 2^n-tiling problem above. First note that for the corridor of width n no counter bits are needed because the vertical matching relation only requires statements of the form "in n steps" rather than 2^n. Here we assume a single proposition 0 marking the left edge of the corridor.

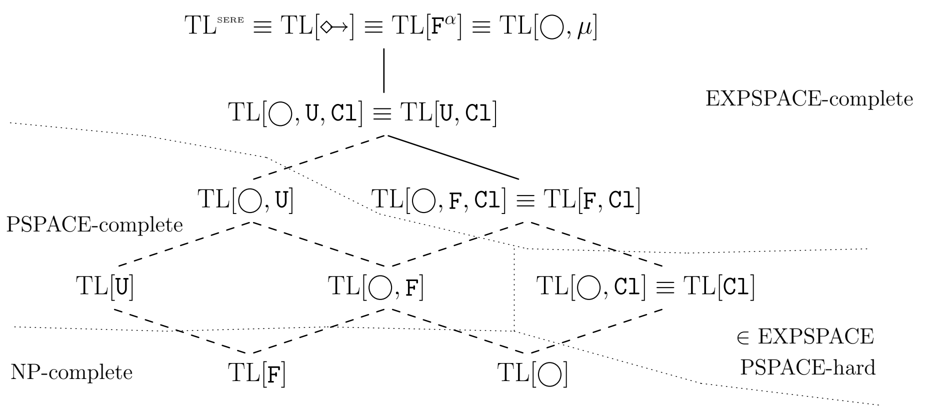
$$\varphi_{edge} \;:=\; \mathtt{Cl}(\; (0; \underbrace{\neg 0; \ldots; \neg 0}_{n-1 \text{ times}})^* \;)$$

Requiring every cell to carry a unique tile is done using φ_{tile} from the proof of Thm. 6. The horizontal matching relation can be axiomatised using formula φ_h from the proof of Thm. 6 with the proposition 0 instead of the formula χ_0. The vertical relation can be axiomatised as follows.

$$\varphi_v \;:=\; \bigwedge_{i=0}^{n} \bigcirc^i \mathtt{Cl}(\; (\bigcup_{i=1}^{m} t_i; \top^n; (\bigcup_{j \in iM_v} t_j))^* \;)$$

Note, again, that the first conjunct ensures matchings between $(0,0)$ and $(0,1)$, between $(1,1)$ and $(1,2)$, etc. The second conjunct ensures matchings between $(1,0)$ and $(1,1)$, between $(2,1)$ and $(2,2)$, etc. Therefore, we need $n+1$ conjuncts altogether to cover the entire corridor.

Let $\varphi := \varphi_{edge} \wedge \varphi_{tile} \wedge \varphi_h \wedge \varphi_v$. Finally, Lemma 3 shows that the $\bigcirc$-operators can be eliminated from φ at no blow-up which finishes the proof. □



$\mathrm{TL}^{\text{SERE}} \equiv \mathrm{TL}[\diamond\!\!\to] \equiv \mathrm{TL}[\mathtt{F}^\alpha] \equiv \mathrm{TL}[\bigcirc, \mu]$
EXPSPACE-complete
$\mathrm{TL}[\bigcirc, \mathtt{U}, \mathtt{Cl}] \equiv \mathrm{TL}[\mathtt{U}, \mathtt{Cl}]$
$\mathrm{TL}[\bigcirc, \mathtt{U}]$
$\mathrm{TL}[\bigcirc, \mathtt{F}, \mathtt{Cl}] \equiv \mathrm{TL}[\mathtt{F}, \mathtt{Cl}]$
PSPACE-complete
$\mathrm{TL}[\mathtt{U}]$
$\mathrm{TL}[\bigcirc, \mathtt{F}]$
$\mathrm{TL}[\bigcirc, \mathtt{Cl}] \equiv \mathrm{TL}[\mathtt{Cl}]$
$\in$ EXPSPACE
PSPACE-hard
NP-complete
$\mathrm{TL}[\mathtt{F}]$
$\mathrm{TL}[\bigcirc]$

Fig. 1. Expressive power and complexity of fragments of $\mathrm{TL}^{\text{SERE}}$

5 Summary and Conclusion

Fig. 1 shows the relationship between fragments of $\mathrm{TL}^{\text{SERE}}$ w.r.t. relative expressive power. Note that $\mathrm{TL}^{\text{SERE}}$ and the alike are equi-expressive to Monadic Second-Order Logic, resp. NBA or ω-regular expressions. LTL, i.e. $\mathrm{TL}[\bigcirc, \mathtt{U}]$, is equi-expressive to First-Order Logic or ω-star-free expressions.

Strict inclusions are marked using dashed lines. The strictness of these has been shown in [4] for the part below $\mathrm{TL}[\bigcirc, \mathtt{U}]$, and in Thm. 2 and Cor. 2 for the others. Strictness of the two remaining inclusions is still open.

Fig. 1 also gives an overview of the complexity of these fragments' satisfiability problems. Again, for the part below $\mathrm{TL}[\bigcirc, \mathtt{U}]$ this has been shown in [9] already. The other results summarise the findings of the theorems and corollaries in the previous section. The exact complexity of $\mathrm{TL}[\mathtt{Cl}]$ is still open. However, if the size of a (semi-extended) regular expression is measured as its syntactical length, then the problem becomes PSPACE-complete. Note that then the translation from $\mathrm{TL}[\mathtt{Cl}]$ formulas to $\exists\forall$-accepting NBA as shown in Lemma 2 produces NBA of at most exponential size whose emptiness can be checked in PSPACE again.

As for all linear time temporal logics, the satisfiability complexity results immediately carry over to the model checking problem for finite transition systems and the implicit "for all paths" semantics. The complexities are the same for all the fragments since the complexity classes mentioned there are deterministic – apart from the fragments $\mathrm{TL}[\mathtt{F}]$ and $\mathrm{TL}[\bigcirc]$. It is known that their model checking problems are co-NP-complete [9].

Apart from the open questions concerning the strictness of two inclusions, the work presented herein gives rise to some other questions regarding the expressive power and complexity of temporal logics with *extended regular expressions*, i.e. with a complementation operator included. It is known that its emptiness problem is non-elementary but it remains to be seen whether the presence of the temporal operators also lifts the satisfiability problem up the exponential space hierarchy by one level.

Note that in this setting the syntax of formulas contains a clear hierarchical structure: regular expressions can occur within temporal formulas but not vice-versa. It remains to be seen what the corresponding complexity and expressiveness results are when both kinds are allowed to be mixed in a straight-forward way: a temporal formula φ can be seen as an atomic proposition that holds in finite words of length exactly 1. Let TL*[..] denote the resulting fragments. With this mixture, it is easy to answer one of the open questions regarding expressive power: we have $\mathrm{TL}[\bigcirc, \mathtt{U}] \leq \mathrm{TL}^*[\mathtt{F}, \mathtt{Cl}]$ because of $\varphi \mathtt{U} \psi \equiv \mathtt{F}\,\psi \wedge \mathtt{Cl}((\varphi)^*; (\psi); \top^*)$.

Finally, a more thorough treatment of succinctness issues between these logics may be desirable.

References

1. Inc. Accellera Organization: Formal semantics of Accellera property specification language (2004) In Appendix B of `http://www.eda.org/vfv/docs/PSL-v1.1.pdf`
2. Barringer, H., Kuiper, R., Pnueli, A.: A really abstract concurrent model and its temporal logic. In: POPL'86. Conf. Record of the 13th Annual ACM Symp. on Principles of Programming Languages, pp. 173–183. ACM, New York (1986)
3. Bustan, D., Fisman, D., Havlicek, J.: Automata constructions for PSL. Technical Report MCS05-04, The Weizmann Institute of Science (2005)
4. Emerson, E.A., Halpern, J.Y.: Decision procedures and expressiveness in the temporal logic of branching time. Journal of Computer and System Sciences 30, 1–24 (1985)
5. Gabbay, D., Pnueli, A., Shelah, S., Stavi, J.: The temporal analysis of fairness. In: POPL'80. Proc. 7th Symp. on Principles of Programming Languages, pp. 163–173. ACM Press, New York (1980)
6. Henriksen, J.G., Thiagarajan, P.S.: Dynamic linear time temporal logic. Annals of Pure and Applied Logic 96(1–3), 187–207 (1999)
7. Pnueli, A.: The temporal logic of programs. In: FOCS'77. Proc. 18th Symp. on Foundations of Computer Science, Providence, RI, USA, pp. 46–57. IEEE Computer Society Press, Los Alamitos (1977)
8. Savitch, W.J.: Relationships between nondeterministic and deterministic tape complexities. Journal of Computer and System Sciences 4, 177–192 (1970)
9. Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. Journal of the Association for Computing Machinery 32(3), 733–749 (1985)
10. Sistla, A.P., Vardi, M.Y., Wolper, P.: Reasoning about infinite computation paths. In: FOCS'83. Proc. 24th Symp. on Foundations of Computer Science, pp. 185–194. IEEE Computer Society Press, Los Alamitos, CA, USA (1983)
11. Thomas, W.: Star-free regular sets of ω-sequences. Information and Control 42(2), 148–156 (1979)
12. van Emde Boas, P.: The convenience of tilings. In: Sorbi, A. (ed.) Complexity, Logic, and Recursion Theory. Lecture notes in pure and applied mathematics, vol. 187, pp. 331–363. Marcel Dekker, Inc. (1997)
13. Vardi, M.Y.: A temporal fixpoint calculus. In: ACM (ed.) POPL'88. Proc. Conf. on Principles of Programming Languages, pp. 250–259. ACM Press, NY, USA (1988)
14. Wolper, P.: Temporal logic can be more expressive. Information and Control 56, 72–99 (1983)