Difficult configurations – On the complexity of LTrL

Igor Walukiewicz

Difficult configurations — on the complexity of *LTrL*

Igor Walukiewicz[1]
Institute of Informatics, Warsaw University
Banacha 2, 02-097 Warsaw, POLAND

January 6, 2005

Abstract

The complexity of *LTrL*, a global linear time temporal logic over traces is investigated. The logic is global because the truth of a formula is evaluated in a global state, also called configuration. The logic is shown to be non-elementary with the main reason for this complexity being the nesting of until operators in formulas. The fragment of the logic without the until operator is shown to be EXPSPACE-hard.

1 Introduction

Infinite words, which are linear orders on *events*, are often used to model executions of systems. Infinite *traces*, which are partial orders on events, are often used to model concurrent systems when we do not want to put some arbitrary ordering on actions occurring concurrently. A *state* of a system in the linear model is just a prefix of an infinite word; it represents the actions that have already happened. A state of a system in the trace model is a *configuration*, i.e., a finite downwards closed set of events that already happened.

Temporal logics over traces come in two sorts: a *local* and a *global* one. The truth of a formula in a *local logic* is evaluated in an event, the truth of a formula in a *global logic* is evaluated in a configuration. Global logics have the advantage of talking directly about configurations hence potentially it is easier to write specifications in them.

[1]Supported by Polish KBN grant No. 8 T11C 002 11

In this paper we investigate the complexity of *LTrL*, a global temporal logic over traces proposed in [TW97]. Originally this logic had special formulas of the form $\langle a^{-1}\rangle \underline{\mathrm{tt}}$. We do not need these formulas for the lower bounds in this paper. Diekert and Gastin has recently shown [DG00] that even without these formulas the logic is expressively complete with respect to first-order logic.

We show that the satisfiability problem for *LTrL* is non-elementary. As it turns out, it is the nesting of until operators that gives such a high complexity. This makes it natural to ask what is the complexity of the logic without the until operator. We investigate a global logic, $LTrL^{-}$, containing only "for some configuration in the future" modality and "next step" modalities. We show that this logic is EXPSPACE-hard. These results give also the bounds on the model checking problem for the logics in question.

The presented results show that the complexity of global logics is bigger than the complexity of local logics. It is well known that LTL, a linear temporal logic over infinite words, is PSPACE-complete. It is still PSPACE-complete if we have just "some time in the future" operator instead of the until operator [SC85]. Local temporal logics for traces proposed in [APP95, Ram96, Thi94] have also PSPACE complexity. It is not known what kinds of global properties are expressible in these local logics. More recently [Nie98, Wal00] some variants of the mu-calculus were proposed. These are expressively complete with respect to monadic second-order logic on traces and have PSPACE complexity. Our results show that expressing global trace properties in these logics, if at all possible, will require big formulas.

There are not many results known about the complexity of global logics. Some undecidability results were obtained in [KPRP95, Pen92]. The most relevant here is the paper by Alur, McMillan and Peled [AMP98]. In this paper an EXPSPACE upper bound is shown for the logic $\mathrm{ISTL}^{\diamond}$. This fragment corresponds to $LTrL^{-}$ with atomic propositions but without next modalities. We conjecture that $LTrL^{-}$ can be decided in EXPSPACE too.

Let us finish this introduction with some remarks on a more general context of this paper. From the verification point of view, traces are interesting for at least two reasons. First, as the development of the trace theory shows [DR95], they are "like words" because most of the properties of words have their counterparts in traces. The generalization from words to traces is interesting because it is far from trivial and it requires new methods and new insights. Next, traces can model systems more faithfully than words as they do not introduce ad hoc dependencies. Because of this, traces promise to offer some help in coping with the state explosion problem [Val92, God96, Pel96].

If we agree that modeling systems with traces is a good enough idea to try then the immediate question is: how to express properties of traces. For this we must understand the complexity of checking properties over traces. Instead of talking about particular properties it is often better to design a logic and talk about a set of properties definable in this logic. A twenty year long development of temporal logics suggests strong candidates for the classes of properties we want to express – the classes of properties expressible in first-order and monadic second-order logic over traces represented as dependence graphs. Here we focus on properties expressible in first-order logic. This class of properties has many different characterizations [EM93] and is a natural counterpart of the class of properties expressible in *LTL* over words. The next question then is: with what kinds of operators we want to express this class of properties. *LTL* and first-order logic can express exactly the same properties of infinite words but, often, *LTL* is preferred because of its low complexity. This low complexity would be useless if it was not often the case that the properties we want to express can be written as small *LTL* formulas. To have this feature in the trace world it seems reasonable to base a logic on configurations and not events. Unfortunately, the present paper shows that one has to be very careful with operators one allows unless one is willing to cope with very high complexity.

In the next section we give the necessary definitions and notations. In Section 3 we describe the proof of the non-elementary lower bound for *LTrL*. In Section 4 we show the EXPSPACE lower bound for the fragment, $LTrL^-$, of *LTrL*.

2 Preliminaries

A (Mazurkiewicz) *trace alphabet* is a pair (Σ, I) where Σ is a finite set of actions and $I \subseteq \Sigma \times \Sigma$ is an irreflexive and symmetric *independence relation*. Its complement $D = (\Sigma \times \Sigma) - I$ is called the dependency relation.

We shall view (Mazurkiewicz) trace, over an alphabet (Σ, I), as a Σ-labelled poset with special properties. Let $(E, \leq, \lambda)$ be a Σ-labelled poset, i.e., $(E, \leq)$ is a poset and $\lambda : E \to \Sigma$ is a labelling function. For $Y \subseteq E$ we define $\downarrow Y = \{x : \exists y \in Y.\ x \leq y\}$ and $\uparrow Y = \{x : \exists y \in Y.\ y \leq x\}$. In case $Y = \{y\}$ is a singleton we shall write $\downarrow y$ $(\uparrow y)$ instead of $\downarrow \{y\}$ $(\uparrow \{y\})$. We also let $\lessdot$ be the immediate successor relation: $x \lessdot y$ iff $x < y$ and for all $z \in E$, $x \leq z \leq y$ implies $x = z$ or $z = y$.

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

(T1) $\forall e \in E.\quad {\downarrow} e$ is a finite set.
(T2) $\forall e, e' \in E.\ \ e \lessdot e' \ \Rightarrow\ (\lambda(e), \lambda(e')) \in D$.
(T3) $\forall e, e' \in E.\ \ (\lambda(e), \lambda(e')) \in D \ \Rightarrow\ e \leq e'$ or $e' \leq e$.

We shall refer to members of E as *events*. All our traces will be infinite, i.e., will have infinitely many events. The set of infinite traces over (Σ, I) is denoted by $TR(\Sigma, I)$.

Let $T = (E, \leq, \lambda)$ be a trace. A *configuration* is a finite subset $C \subseteq E$ such that $C = {\downarrow} C$. We let $\mathit{Conf}(T)$ be the set of configurations of T and we let C, C', C'' range over $\mathit{Conf}(T)$. Note that $\emptyset$, the empty set, is a configuration and ${\downarrow} e$ is a configuration for every $e \in E$. Finally, the transition relation $\rightarrow_T \subseteq \mathit{Conf}(T) \times \Sigma \times \mathit{Conf}(T)$ is given by: $C \overset{a}{\rightarrow}_T C'$ iff there exists $e \in E$ such that $\lambda(e) = a$ and $e \notin C$ and $C' = C \cup \{e\}$. It is easy to see that if $C \overset{a}{\rightarrow}_T C'$ and $C \overset{a}{\rightarrow}_T C''$ then $C' = C''$.

The set of formulas of our linear time temporal logic of traces (*LTrL*) is defined as follows:

$$\mathit{LTrL}(\Sigma, I) ::= \underline{\mathrm{tt}} \mid \neg\alpha \mid \alpha \wedge \beta \mid \langle a \rangle \alpha \mid \alpha \ \mathbb{U}\ \beta$$

where α, β range over formulas and a over actions. In the original definition of *LTrL* there were also formulas of the form $\langle a^{-1} \rangle \underline{\mathrm{tt}}$. These formulas are not needed for the lower bounds we present here. Recently Diekert and Gastin [DG00] has shown that these formulas do not increase the expressive power of the logic.

A model of *LTrL* is a trace $T = (E, \leq, \lambda)$. The relation $T, C \models \alpha$ will denote that formula $\alpha \in \mathit{LTrL}(\Sigma, I)$ is satisfied at configuration $C \in \mathit{Conf}(T)$. This notion is defined via:

- $T, C \models \underline{\mathrm{tt}}$. Furthermore $\neg$ and $\wedge$ are interpreted in the usual way.
- $T, C \models \langle a \rangle \alpha$ iff $\exists C' \in \mathit{Conf}(T).\ C \overset{a}{\rightarrow}_T C'$ and $T, C' \models \alpha$.
- $T, C \models \alpha \ \mathbb{U}\ \beta$ iff $\exists C' \in \mathit{Conf}(T).\ C \subseteq C'$ and $T, C' \models \beta$ and $\forall C'' \in \mathit{Conf}(T).\ C \subseteq C'' \subset C'$ implies $T, C'' \models \alpha$.

We will write $T \vDash \alpha$ for $T, \emptyset \vDash \alpha$. The semantics of until allows us to express "sometime" and "always" modalities:

$$\mathbb{E}\alpha \equiv \underline{\mathrm{tt}} \ \mathbb{U}\ \alpha \qquad \mathbb{A}\alpha \equiv \neg\mathbb{E}\neg\alpha$$

which have the expected semantics:

- $T, C \vDash \mathbb{E}\alpha$ iff $\exists C' \in \mathit{Conf}(T).\ C \subseteq C'$ and $T, C' \models \alpha$.

- $T, C \models \mathbb{A}\alpha$ iff $\forall C' \in \mathit{Conf}(T).\ C \subseteq C'$ implies $T, C' \models \alpha$.

For a sequence of actions $v_1 \dots v_n \in \Sigma^*$ we will write $\langle v_1 \dots v_n \rangle \alpha$ instead of $\langle v_1 \rangle \langle v_2 \rangle \dots \langle v_n \rangle \alpha$.

3 The complexity of *LTrL*

In this section we show a non-elementary lower bound for the complexity of the satisfiability problem for *LTrL*. Let *Tower* stand for the "tower of exponentials" function, i.e., $\mathit{Tower}(0, n) = n$ and $\mathit{Tower}(k+1, n) = 2^{\mathit{Tower}(k,n)}$.

Given a $\lambda x.\, \mathit{Tower}(m, x)$ space bounded Turing machine M and a word w, we will construct a $\mathcal{O}(|w|2^{\mathcal{O}(m)} + |w| + |M|)$ size formula that is satisfiable iff w is accepted by M. This will show that the satisfiability problem for *LTrL* cannot be solved in time bounded by the function $\lambda x.\, \mathit{Tower}(m, x)$ for any fixed m. On the other hand the satisfiability problem can be solved in time $\lambda m.\, \mathit{Tower}(m, 2)$. This follows from the translation of *LTrL* into first-order logic over infinite words [TW97].

The plan of this section is the following. Our first goal is to show, on an example, how to construct big counters with small formulas. Even before the example we will introduce some extensions to our logic and show how to code these extensions in the basic language. After the example we will formally define what a big counter is and we will say what kind of formulas we can write. Finally, we will explain how to use big counters to code long computations of Turing machines.

Preliminary definitions and notations

For $i = 0, 1, 2$ let $\Sigma_i = \{a_i, b_i\}$ and let $\overline{\Sigma}_i = \{\overline{a}_i, \overline{b}_i\}$. Put $\Delta = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ and similarly for $\overline{\Delta}$. Our trace alphabet will be $(\Delta \cup \overline{\Delta}, I)$ where I says that two letters are independent iff one has a bar and the other does not, i.e., $I = (\Delta \times \overline{\Delta}) \cup (\overline{\Delta} \times \Delta)$.

In the formulas below we will use the construction $\langle \Sigma \rangle \alpha$ for a set of letters $\Sigma \subseteq \Delta$ or $\Sigma \subseteq \overline{\Delta}$. This just an abbreviation for $\bigvee_{\sigma \in \Sigma} \langle \sigma \rangle \alpha$. Of course if we want to replace $\langle \Sigma \rangle \alpha$ with what it stands for, we may obtain an exponentially bigger formula. This is not a problem for us because we want to show a non-elementary lower bound. Anyway, below we will show how to code $\langle \Sigma \rangle \alpha$ more efficiently if we have traces of a special form.

We will also use the construction $\langle \Sigma^* \rangle \alpha$ for a set of letters $\Sigma \subseteq \Delta$ or $\Sigma \subseteq \overline{\Delta}$. The meaning of such a formula is that $\langle v \rangle \alpha$ holds for some $v \in \Sigma^*$. We need to require some special form of a trace to encode this formula with the constructions we have in *LTrL*. Every trace over the alphabet we are working with consists of two, independent from each other, sequences of events (one over Δ and one over $\overline{\Delta}$). Assume that we have two more letters $e, \overline{e}$. Letter e depends only on letters from Δ and letter $\overline{e}$ depends only on letters from $\overline{\Delta}$. We will force our traces to have e on every even position of the Δ sequence and $\overline{e}$ on every even position of the $\overline{\Delta}$ sequence. So traces we will consider will look as follows:

$$\begin{gathered} e\sigma_0 e\sigma_1 \cdots e\sigma_i \cdots \\ \overline{e}\rho_0\overline{e}\rho_1 \cdots \overline{e}\rho_i \cdots \end{gathered}$$

where $\sigma_0, \sigma_1, \ldots \in \Delta$ and $\rho_0, \rho_1, \ldots \in \overline{\Delta}$. It is easy to write a formula forcing the trace to be of this shape. Over such traces the formula $\langle(\Sigma \cup \{e\})^*\rangle\alpha$ is equivalent to $\big(\langle\overline{e}\rangle\underline{\mathrm{tt}} \wedge \langle\Sigma \cup \{e\}\rangle\underline{\mathrm{tt}}\big)\ \mathbb{U}\,(\alpha \wedge \langle e\rangle\langle\overline{e}\rangle\underline{\mathrm{tt}})$. Strictly speaking it is equivalent in configurations satisfying $\langle e\rangle\langle\overline{e}\rangle\underline{\mathrm{tt}}$. We could avoid this problem but anyway we will be interested only in configurations satisfying $\langle e\rangle\langle\overline{e}\rangle\underline{\mathrm{tt}}$. Let us also mention that if we have this form of trace then there is a more efficient way to code $\langle\Sigma\rangle\alpha$. In case $\Sigma \subseteq \Delta$, we can define it with the formula: $\langle e\rangle\big(\langle\overline{e}\rangle\underline{\mathrm{tt}} \wedge \neg\langle e\rangle\underline{\mathrm{tt}} \wedge \bigwedge_{a\notin(\Delta-\Sigma)} \neg\langle a\rangle\underline{\mathrm{tt}}\big)\ \mathbb{U}\,(\alpha \wedge \langle e\rangle\underline{\mathrm{tt}})$. Because α appears only once in this formula, we avoid an exponential blowup caused by the previous translation.

To make the presentation easier we will forget about e and $\overline{e}$ letters and use $\langle\Sigma^*\rangle\alpha$ construct as if we had it in our language. Nevertheless all formulas we will write below can be translated to the core language without $\langle\Sigma^*\rangle\alpha$ construct. First step in this translation is to add a formula requiring that every second letter in the upper sequence is e and every second letter in the lower sequence is $\overline{e}$. Then we need to make translated formulas to see only configurations starting with $e\overline{e}$; or in other words configurations satisfying $\langle e\rangle\langle\overline{e}\rangle\underline{\mathrm{tt}}$. For this we replace each $\langle a\rangle\beta$ by $\langle ea\rangle\beta$ and similarly for the letters with bars. We replace $\alpha\ \mathbb{U}\ \beta$ with $(\langle e\rangle\langle\overline{e}\rangle\underline{\mathrm{tt}} \Rightarrow \alpha)\ \mathbb{U}\,(\beta \wedge \langle e\rangle\langle\overline{e}\rangle\underline{\mathrm{tt}})$. Finally, we replace each $\langle\Sigma^*\rangle\alpha$ by its definition from the above.

Counters: an example

After this preliminary definitions, let us start with the description of the construction. A word $l \in (\Sigma_0)^n$ can be considered as a counter when we identify a_0 with 0 and b_0 with 1. The value of such a counter $\sigma_0 \ldots \sigma_{n-1} \in (\Sigma_0)^n$ is $\sum_{i=0,\ldots,n-1} \sigma_i 2^i$. (Note that the most significant digit is to the right.) Similarly an element of $(\overline{\Sigma}_0)^n$ can be regarded as a counter.

Consider the following formulas:

$$\begin{aligned}
counter_0 \equiv& \langle\Sigma_0\rangle^n\langle\overline{\Sigma}_0\rangle^n\langle\Sigma_1\rangle\langle\overline{\Sigma}_1\rangle\underline{\mathrm{tt}}\\
same_0 \equiv& \bigwedge_{i=0,\dots,n-1}\langle\Sigma_0\rangle^i\langle\overline{\Sigma}_0\rangle^i\big(\langle a_0\rangle\langle\overline{a}_0\rangle\underline{\mathrm{tt}}\vee\langle b_0\rangle\langle\overline{b}_0\rangle\underline{\mathrm{tt}}\big)\\
first_0 \equiv& \langle a_0\rangle^n\langle\overline{a}_0\rangle^n\langle\Sigma_1\rangle\langle\overline{\Sigma}_1\rangle\underline{\mathrm{tt}}
\end{aligned}$$

Recall that the letters from Δ are independent from the letters from $\overline{\Delta}$ so, for example, $\langle a_0\rangle\langle\overline{a}_0\rangle\underline{\mathrm{tt}}$ is equivalent to $\langle\overline{a}_0\rangle\langle a_0\rangle\underline{\mathrm{tt}}$. The formula $counter_0$ says that a trace starts with two counters, one over Σ_0 and one over $\overline{\Sigma}_0$. After these counters, there are letters from Σ_1 and $\overline{\Sigma}_1$ which will be used later for defining bigger counters. The formula $same_0$ says that the counters at the beginning of a trace represent the same values. One can also write, a slightly longer, formula $next_0$ saying that the value of the Σ_0 counter is one plus the value of the $\overline{\Sigma}_0$ counter. Finally, the formula $first_0$ just says that the values of the two counters are 0. Similarly we can write a formula last_0 saying the that values of the two counters are 2^n-1.

Now we want to write a formula $counter_1$ forcing the beginning of a trace to consist of two exponentially longer counters. We call them level 1 counters as opposed to level 0 counters defined above. For level 1 counter the beginning of a trace will have the form

$$\begin{matrix} l_0\sigma_0 l_1\sigma_1\dots l_k\sigma_k\sigma\\ \overline{l}_0\rho_0\overline{l}_1\rho_1\dots\overline{l}_k\rho_k\rho\end{matrix} \tag{1}$$

where $k=2^n-1$; $l_i\in(\Sigma_0)^n$ and $\overline{l}_i\in(\overline{\Sigma}_0)^n$ are counters representing successive numbers; $\sigma_i\in\Sigma_1$, $\rho_i\in\overline{\Sigma}_1$, $\sigma\in\Sigma_2$, $\rho\in\overline{\Sigma}_2$ (for $i=0,\dots,k$).

$$\begin{aligned}
counter_1 \equiv& first_0 \wedge(\alpha\ \mathbb{U}\ \langle\Sigma_2\rangle\langle\overline{\Sigma}_2\rangle\underline{\mathrm{tt}})\\
\alpha \equiv& same_0 \Rightarrow (\mathrm{last}_0\wedge\beta_1)\vee(\neg\,\mathrm{last}_0\wedge\beta_2)\\
\beta_1 \equiv& \langle\Sigma_0\rangle^n\langle\overline{\Sigma}_0\rangle^n\langle\Sigma_1\rangle\langle\overline{\Sigma}_1\rangle\langle\Sigma_2\rangle\langle\overline{\Sigma}_2\rangle\underline{\mathrm{tt}}\\
\beta_2 \equiv& \langle\Sigma_0\rangle^n\langle\Sigma_1\rangle\big(\,next_0\wedge\langle\overline{\Sigma}_0\rangle^n\langle\overline{\Sigma}_1\rangle\,same_0\,\big)
\end{aligned}$$

The first conjunct of formula $counter_1$ says that a trace should begin with two level 0 counters representing 0 followed by σ_0 and ρ_0. The second conjunct of $counter_1$ says that α should be true until we reach (uniquely determined) configuration from which it is possible to do $\langle\Sigma_2\rangle\langle\overline{\Sigma}_2\rangle$. Formula α says that whenever we are in a configuration ahead of which there are two level 0 counters with the same values then either these counters represent the maximal values and β_1 holds or otherwise β_2 holds. Formula β_1 takes care about the end of the counter. It says that after the counters we should have $\sigma_k, \rho_k, \sigma, \rho$. Formula β_2 says that the value of the next Σ_0 counter is bigger by one than the value of the current $\overline{\Sigma}_0$ counter and is equal to the value of the next $\overline{\Sigma}_0$ counter.

A σ-counter, i.e., the upper line of the trace (1) represents the number $\sum_{i=0,\dots,k} \sigma_i 2^i$ (with the convention that a stands for 0 and b stands for 1). Similarly for the ρ-counter, i.e., the lower line of the trace. We can force the value of the σ-counter to have the same value as the ρ-counter. This can be done using the formula same_1:

$$\begin{aligned} \mathit{same}_1 &\equiv \delta \;\mathbb{U}\; \langle \Sigma_2 \rangle \langle \Sigma_2 \rangle \underline{\mathtt{tt}} \\ \delta &\equiv \mathit{same}_0 \Rightarrow \langle \Sigma_0 \rangle^n \langle \overline{\Sigma}_0 \rangle^n (\langle a_1 \rangle \langle \overline{a}_1 \rangle \underline{\mathtt{tt}} \vee \langle b_1 \rangle \langle \overline{b}_1 \rangle \underline{\mathtt{tt}}) \end{aligned}$$

This formula says that whenever a configuration starts with two counters representing the same value of level 0 counters then these counters are followed either by two letters representing 0 or by two letters representing 1.

With a slightly more elaborate formula, next_1, we can force the value of the σ-counter to be one plus the value of the ρ-counter.

This way we have obtained exponentially bigger counters than the ones we have started with. We have also means to test whether two counters are equal and to test that the value of one counter is one plus the value of the other. It is easy to write formulas saying that the a counter has value 0 or that it has the maximal possible value. At this point we can iterate the whole construction to get even bigger counters.

Counters: definition

For describing inductive construction of bigger and bigger counters we need a precise definition of what a counter is. To simplify matters, a counter of level 0 does not count to n as in our example above but just to 1. A *counter of level* $n > 0$ is a trace with the prefix of the form:

$$\begin{array}{c} l_0 \sigma_0 l_1 \sigma_1 \dots l_k \sigma_k \sigma \\ \overline{l}_0 \rho_0 \overline{l}_1 \rho_1 \dots \overline{l}_k \rho_k \rho \end{array} \tag{2}$$

where $k = \mathit{Tower}(n, 2) - 1$. For each $i = 0, \dots, k$, a trace $l_i \sigma_i \overline{l}_i \rho_i$ is a counter of level $n-1$ representing the number i; these counters are over the alphabets $\Delta_{n-1} = \bigcup_{i=1,\dots n-1} \Sigma_i$ and $\overline{\Delta}_{n-1}$. Letters σ_i are from $\Sigma_n = \{a_n, b_n\}$, letters

ρ_i are from $\overline{\Sigma}_n$. Finally $\sigma \in \Sigma_{n+1}$, $\rho \in \overline{\Sigma}_{n+1}$. We use $a, \overline{a}$ with indices to represent 0 and $b, \overline{b}$ with indices to represent 1. So the sequence $\sigma_0, \ldots, \sigma_k$ represents the number $\sum_{i=0}^{k} \sigma_i 2^i$.

We can construct formulas: (i) *counter*(n) saying that a trace starts with the counter of level n; (ii) *same*(n) saying that if a trace is of the form (2) then $\sigma_0, \ldots, \sigma_k$ and $\rho_0, \ldots, \rho_k$ represent the same numbers; (iii) *next*(n) saying that the number represented by $\sigma_0, \ldots, \sigma_k$ is one plus the number represented by $\rho_0, \ldots, \rho_k$. Below we only present the counter formula. The remaining formulas are given in Appendix A.

$$\begin{aligned}
\mathit{counter}(0) \equiv& \langle \Sigma_0 \rangle \langle \overline{\Sigma}_0 \rangle \langle \Sigma_1 \rangle \langle \overline{\Sigma}_1 \rangle \underline{\mathrm{tt}} \\
\mathit{counter}(n+1) \equiv& \mathit{first}(n) \wedge (\alpha \; \mathbb{U} \, \langle \Sigma_{n+2} \rangle \langle \overline{\Sigma}_{n+2} \rangle \underline{\mathrm{tt}}) \\
\alpha(n) \equiv& (\mathit{counter}(n) \wedge \mathit{same}(n)) \Rightarrow \\
& \quad \big[(\mathrm{last}(n) \wedge \beta_1(n+1)) \vee (\neg \, \mathrm{last}(n) \wedge \beta_2(n+1)) \big] \\
\beta_1(n+1) \equiv& \langle (\Delta_n \cup \overline{\Delta}_n)^* \rangle \langle \Sigma_{n+1} \rangle \langle \overline{\Sigma}_{n+1} \rangle \langle \Sigma_{n+2} \rangle \langle \overline{\Sigma}_{n+2} \rangle \underline{\mathrm{tt}} \\
\beta_2(n+1) \equiv& \langle \Delta_n^* \rangle \langle \Sigma_{n+1} \rangle \big(\, \mathit{next}(n) \wedge \langle \overline{\Delta}_n^* \rangle \langle \overline{\Sigma}_n + 1 \rangle \, \mathit{same}(n) \big)
\end{aligned}$$

We have:

Lemma 1 For every trace T and $n \in \mathbb{N}$. T is a counter of level n iff $T \vDash \mathit{counter}(n)$. The size of the formula *counter*(n) is $2^{\mathcal{O}(n)}$.

The proof of the lemma is given in Appendix A.

Encoding of Turing Machines

Let $m \geq 0$ and let M be a $\lambda x.\ \mathit{Tower}(m, x)$ space bounded Turing machine. For a given word $w \in \{0,1\}^*$ we are going to write a formula *Accept* that is satisfiable iff $w \in L(M)$. Configurations of M on w can be of length $\mathit{Tower}(m, |w|)$, so we need counters able to count up to this number. Let n be the smallest number such that $\mathit{Tower}(n, 2) \geq \mathit{Tower}(m, |w|)$. Clearly $n \leq m + \log(|w|)$.

Let Q be the set of states of M, Γ its tape alphabet, q_I, q_F its initial and finial sates respectively. The blank symbol is denoted by B. A configuration is a word $\vdash v q v' \dashv$ with v representing symbols to the left of the head, v' representing symbols to the right, and the head looking at the first symbol of v'. We use symbols $\vdash$ and $\dashv$ as end markers. Let $\Omega = Q \cup \Gamma \cup \{\vdash, \dashv\}$ be the alphabet needed to write down configurations.

We can now define our trace alphabet. Recall that Δ_n stands for $\bigcup_{i=0,\dots,n} \Sigma_n$ and $\overline{\Delta}_n$ denotes the corresponding set of letters with bars. Our trace alphabet is $(\Delta_n \cup \Omega \cup \overline{\Delta}_n \cup \overline{\Omega} \cup \{\$, \overline{\$}\}, I)$, where I says that two letters are independent iff one has a bar and the other doesn't.

The first step in constructing the formula *Accept* is to write a formula *Conf* saying that a prefix of a trace is of the form:

$$\begin{array}{c} l_0\sigma_0^0 l_1\sigma_1^0 \dots l_k\sigma_k^0 \$ \\ \overline{l}_0\rho_0^0 \overline{l}_1\rho_1^0 \dots \overline{l}_k\rho_k^0 \overline{\$} \end{array} \tag{3}$$

where $k = \mathit{Tower}(n, 2)$, the letters σ_i come from Ω, letters ρ_i come from $\overline{\Omega}$ and $\$, \overline{\$}$ are special end markers. This form is exactly the same as the form of the counter (2) but with letters σ_i and ρ_i from a different alphabet. Formula *Conf* can be obtained by a simple modification of formula $\mathit{counter}(n+1)$ (replace Σ_{n+1} by Ω and Σ_{n+2} by $\{\$\}$).

To write the formula *Accept* we will need tools to define initial configuration, and to say that one configuration is obtained from the other in one step of M.

We start with a formula defining the initial configuration.

$$\begin{aligned} \mathit{Init} \equiv{}& \mathit{Conf} \wedge \mathit{Same} \\ & \wedge \langle (\Delta_n)^* \vdash (\Delta_n)^* q_I (\Delta_n)^* w_1 \dots (\Delta_n)^* w_{|w|} (\Delta_n)^* \dashv \rangle\, \mathit{Blanks} \\ \mathit{Same} \equiv{}& \mathit{Conf} \wedge (\delta \;\mathbb{U}\; \langle \$\overline{\$} \rangle \underline{\mathrm{tt}}) \\ \delta \equiv{}& \mathit{same}(n) \Rightarrow \langle \Delta_n^* \rangle \langle \overline{\Delta}_n^* \rangle \bigvee_{\sigma \in \Omega} \langle \sigma \rangle \langle \overline{\sigma} \rangle \underline{\mathrm{tt}} \\ \mathit{Blanks} \equiv{}& \langle \Delta_n \cup \{B\} \rangle \underline{\mathrm{tt}} \;\mathbb{U}\; \langle \$ \rangle \underline{\mathrm{tt}} \end{aligned}$$

Formula *Conf* forces the beginning of the trace to be of shape (3). Formula *Same* requires that for every $i = 0, \dots, k$ the letter ρ_i^0 is the letter σ_i^0 with a bar over it. The last conjunct of formula *Init* requires that the first letters $\sigma_0^0 \cdots \sigma_{|w|+3}^0$ form the string $\vdash q_I w \dashv$ representing the initial configuration. The rest of the symbols σ_i^0 is forced to be blanks.

Next we write the formula *Comp* which requires that each configuration is followed by a configuration representing the next step of the simulated machine. The intention is that the trace should look like:

$$\begin{array}{cccc} l_0\sigma_0^0 l_1\sigma_1^0 \dots l_k\sigma_k^0 \$ & \cdots & l_0\sigma_0^i l_1\sigma_1^i \dots l_k\sigma_k^i \$ & \cdots \\ \overline{l}_0\rho_0^0 \overline{l}_1\rho_1^0 \dots \overline{l}_k\rho_k^0 \$ & \cdots & \overline{l}_0\rho_0^i \overline{l}_1\rho_1^i \dots \overline{l}_k\rho_k^i \$ & \cdots \end{array} \tag{4}$$

with $\sigma_0^i \dots \sigma_k^i$ representing i-th configuration of the machine and each ρ_j^i being the same letter as σ_j^i but with a bar (for all $i > 0$ and $j = 0, \dots, k$).

This formula can be written as:

$$\begin{aligned} \mathit{Comp} \equiv& \alpha \ \mathbb{U} \,(\mathit{Conf} \wedge \langle \Delta_n^* \rangle \langle q_F \rangle \underline{\mathrm{tt}}) \\ \alpha \equiv& \mathit{Same} \Rightarrow \langle (\Delta_n \cup \Omega)^* \rangle \langle \$ \rangle \big(\mathit{Step} \wedge \langle (\overline{\Delta}_n \cup \overline{\Omega})^* \rangle \langle \overline{\$} \rangle (\mathit{Conf} \wedge \mathit{Same}) \big) \end{aligned}$$

The formula says that if we are in a trace configuration where the upper and the lower sequence represent the same machine configuration then the next machine configuration on the upper sequence should be obtainable in one step of the machine (this is the role of formula *Step*) and the next configuration on the lower sequence should be the same as in the upper sequence (this is guaranteed by the conjunction $\mathit{Conf} \wedge \mathit{Same}$).

Formula *Step*, that we have not presented, looks at three consecutive letters of the lower configuration (i.e., the configuration written in $\overline{\Omega}$), consults the transition table of the machine and decides what letters should there be in the upper configuration. We need to look at three letters at the time because a letter may change at some position if and only if it is adjacent to a state.

The formula *Accept* is $\mathit{Init} \wedge \mathit{Comp}$. We obtain:

Lemma 2 Formula *Accept* is satisfiable iff $w \in L(M)$

In the construction of *Accept* we have used the alphabet depending on the machine. Fortunately the traces we are interested in consist always of two independent sequences of events. Hence we can code big alphabets that we have used, with a four letter alphabet – two letters for each of the sequences.

Finally, let us calculate the size of the formula *Accept*. The formula *Conf* is of the same size as $\mathit{counter}(n+1)$ hence of the size $2^{\mathcal{O}(n)}$. Also $\mathit{same}(n)$ is of the size $2^{\mathcal{O}(n)}$. This makes the size of *Init* to be $\mathcal{O}(|w| + 2^{\mathcal{O}(n)})$. The only new element in the formula *Comp* is the formula *Step*. This formula encodes the transition function of the machine and uses $\mathit{same}(n)$. Hence the size of *Comp* is $\mathcal{O}(|M| + 2^{\mathcal{O}(n)})$. This makes the whole formula *Accept* to be of the size $\mathcal{O}(2^{\mathcal{O}(n)} + |w| + |M|)$. Finally comes the duty of removing $\langle \Delta \rangle \alpha$ and $\langle \Delta^* \rangle \alpha$ constructs but this causes only linear blowup.

Summarizing, for a $\lambda x.\, \mathit{Tower}(m, x)$ space bounded machine M and a word w we construct a $\mathcal{O}(|w| 2^{\mathcal{O}(m)} + |w| + |M|)$ size formula *Accept* that is satisfiable iff $w \in L(M)$. This implies:

Theorem 3 *Let (Σ, I) be a trace alphabet containing letters $a, b, \overline{a}, \overline{b}$ with the only dependencies among them being that between a and b and between $\overline{a}$ and $\overline{b}$. The satisfiability problem for LTrL over (Σ, I) is non-elementary.*

4 A lower bound for the fragment of *LTrL*

As the previous section shows, it is until operator that is responsible for non-elementary complexity. In this section we will deal with *LTrL* without until. Instead of until we will allow "sometime" and "always" modalities ($\mathbb{E}$ and $\mathbb{A}$ respectively) and a new next step modality $\langle\cdot\rangle\alpha$ with the semantics:

$$T, C \vDash \langle\cdot\rangle\alpha \text{ iff } T, C \vDash \langle a\rangle\alpha \text{ for some action } a$$

We call this logic $LTrL^-$. The addition of the new modality requires some justification. One justification is that we don't know the complexity of $LTrL^-$ without this modality. We don't know its complexity even in the case when all the letters dependent on each other. In this case we obtain *LTL*, a linear time temporal logic, but without until, propositional constants and arbitrary next time modality; what is left are $\langle a\rangle$ modalities and sometimes in the future modality. Of course $\langle\cdot\rangle\alpha$ is equivalent to $\bigvee_{a\in\Sigma}\langle a\rangle\alpha$ but this definition shows that the formulas using $\langle\cdot\rangle$ may be exponentially more succinct. Finally, let us say that if we add any form of "trace independent" propositional constants to $LTrL^-$ then we don't need $\langle\cdot\rangle$ modality to obtain the EXPSPACE lower bound.

To encode computations of EXPSPACE Turing machines in traces we will use similar ideas as in the previous section although we will not be able to construct as huge counters as before because for this we need the until operator. Here we will use counter alphabets: $\{a, b\}$, $\{c, d\}$, $\{\overline{a}, \overline{b}\}$ and $\{\overline{c}, \overline{d}\}$. A counter is a word over one of these four alphabets. The interpretation is that a, b stand for 0 and b, d stand for 1.

Let M be a $\lambda x.2^x$ space bounded Turing machine. Let Q be its set of states, Γ its tape alphabet, q_I and q_F its initial and final states respectively. Let $w \in \Gamma^*$ be a word, let n be the length of w and let $k = 2^n - 1$. We are going to write a formula *Accept* having the property:

$$\textit{Accept} \text{ is satisfiable} \quad \text{iff} \quad w \in L(M) \tag{5}$$

The configuration of M is, as before, a string $\vdash vqv' \dashv$, with $v, v' \in \Gamma^*$ and $q \in Q$. We write $\Omega = Q \cup \Gamma \cup \{\vdash, \dashv\}$ for the alphabet needed to write down configurations. Let $\Delta = \{a, b, c, d\}$ be the counter alphabet and let $\overline{\Omega}$ and $\overline{\Delta}$ stand for the corresponding sets of letters with bars over them.

Our trace alphabet is $(\Omega \cup \Delta \cup \overline{\Omega} \cup \overline{\Delta} \cup \{\downarrow, \nearrow\}, I)$ where I is the smallest symmetric relation containing:

$$\big((\Omega \cup \Delta) \times (\overline{\Omega} \cup \overline{\Delta})\big) \cup \big(\{\downarrow\} \times (\overline{\Omega} \cup \overline{\Delta})\big) \cup \big(\{\nearrow\} \times (\Omega \cup \Delta)\big)$$

In words: letters with bars are independent from the letters without bars; the symbol $\downarrow$ depends only on letters without bars and $\nearrow$; the symbol $\nearrow$ depends only on letters with bars and $\downarrow$.

The shape of traces we are after is:

$$\begin{tikzcd}[column sep=small]
l_0^a\sigma_0^0 l_1^a\sigma_1^0\dots l_k^a\sigma_k^0 & \downarrow \arrow[d, dashed] & l_0^c\sigma_0^1 l_1^c\sigma_1^1\dots l_k^c\sigma_k^1 & \downarrow \arrow[d, dashed] & \cdots & \downarrow \arrow[d, dashed] & l_0^a\sigma_0^i l_1^a\sigma_1^i\dots l_k^a\sigma_k^i & \downarrow \arrow[d, dashed] & \cdots \\
l_0^{\overline{a}}\rho_0^0 l_1^{\overline{a}}\rho_1^0\dots l_k^{\overline{a}}\rho_k^0 & \nearrow \arrow[ur, dashed] & l_0^{\overline{c}}\rho_0^1 l_1^{\overline{c}}\rho_1^1\dots l_k^{\overline{c}}\rho_k^1 & \nearrow & \cdots & \nearrow \arrow[ur, dashed] & l_0^{\overline{a}}\rho_0^i l_1^{\overline{a}}\rho_1^i\dots l_k^{\overline{a}}\rho_k^i & \nearrow & \cdots
\end{tikzcd} \tag{6}$$

The dashed arrows represent additional dependencies and the meanings of the components is the following. For every $i = 1, \dots, k$, $l_i^a \in \{a, b\}^n$ is a counter representing number i; similarly for $l_i^{\overline{a}}$, l_i^c $l_i^{\overline{c}}$. Letters $\sigma_j^i, \rho_j^i \in \Omega$ are used to describe configurations. Intuitively $\sigma_0^i \cdots \sigma_k^i$ will represent i-th configuration. Letters $\downarrow, \nearrow$ are used to force synchronization and their role we will explain later.

In our formulas we will use the construction $\langle\Sigma\rangle\alpha$, for some set of letters $\Sigma \subseteq (\Omega \cup \Delta)$. To have a translation of this construction into our core language without causing an exponential blowup we need once again to use the trick with e, $\overline{e}$ letters. We extend the shape of the trace form (6) by adding e as every second letter in the upper sequence and $\overline{e}$ as every second letter in the lower sequence. Having this we can define $\langle\Sigma\rangle\alpha$ construction by $\langle\cdot\rangle(\alpha \wedge \langle\overline{e}\rangle\underline{\mathrm{tt}}) \wedge \bigwedge_{a\not\in\Sigma} \neg\langle a\rangle\underline{\mathrm{tt}}$. This long formula is equivalent to $\langle\Sigma\rangle\alpha$ only in configurations satisfying $\langle\overline{e}\rangle\underline{\mathrm{tt}}$ but these will be the only configurations we will be interested in. As in the previous section, we forget about the complication caused by adding $e, \overline{e}$ letters and pretend that we have $\langle\Sigma\rangle\alpha$ construct from the start. So in the formulas we will write, we will never mention $e, \overline{e}$ letters.

We can write $\mathcal{O}(n)$ size formula *Shape* forcing a trace to be of the form presented in (6) (c.f. Appendix B). We can also write a formula *Init* saying that $\sigma_0^0 \dots \sigma_{|w|+3}^0$ form the initial configuration of M on w which is $\vdash q_I w \dashv$. With a formula of size $\mathcal{O}(n)$, and no until, we cannot say that every σ_i^0 for $i > |w| + 3$ is blank. This is not a problem as we will always look only at the prefix up to the first $\dashv$ symbol and we will assume that M accepts with the empty tape.

For every $i \in \mathbb{N}$, we would like to force $\sigma_0^{i+1} \dots \sigma_k^{i+1}$ to represent the next configuration after $\sigma_0^i \dots \sigma_k^i$. First consider the formula:

$$\mathit{same}(a, \overline{a}) \equiv \bigwedge_{i=0,\dots,n-1} \langle\Delta\rangle^i \langle\overline{\Delta}\rangle^i (\langle a\rangle\langle\overline{a}\rangle\underline{\mathrm{tt}} \vee \langle b\rangle\langle\overline{b}\rangle\underline{\mathrm{tt}})$$

The formula $\mathit{same}(a, \overline{a})$ says that from the current configuration we see two counters, one over $\{a, b\}$ and one over $\{\overline{a}, \overline{b}\}$, representing the same numbers.

Now, we can explain the role of the synchronization letters $\downarrow, \nearrow$. Because of the structure of the trace forced by these letters, if some configuration satisfies $same(a,\overline{a})$ then this configuration must be necessary of the form symbolized by the thick vertical line:

$$\begin{tikzcd}
l_0^a\sigma_0^0 l_1^a\sigma_1^0\dots l_k^a\sigma_k^0 & \downarrow \arrow[d, dashed] & l_0^c\sigma_0^1 l_1^c\sigma_1^1\dots l_k^c\sigma_k^1 & \downarrow \arrow[d, dashed] & \cdots & \downarrow \arrow[d, dashed] & l_0^a\sigma_0^i l_1^a\sigma_1^i\dots & l_j^a\sigma_j^i\dots l_k^a\sigma_k^i & \downarrow \arrow[d, dashed] & \cdots \\
l_0^{\overline{a}}\rho_0^0 l_1^{\overline{a}}\rho_1^0\dots l_k^{\overline{a}}\rho_k^0 & \nearrow \arrow[urr, dashed] & l_0^{\overline{c}}\rho_0^1 l_1^{\overline{c}}\rho_1^1\dots l_k^{\overline{c}}\rho_k^1 & \nearrow & \cdots & \nearrow \arrow[urrr, dashed] & l_0^{\overline{a}}\rho_0^i l_1^{\overline{a}}\rho_1^i\dots & l_j^{\overline{a}}\rho_j^i\dots l_k^{\overline{a}}\rho_k^i & \nearrow & \cdots
\end{tikzcd}$$

for some i and j. That the j's in σ_j^i and ρ_j^i are the same is due to the fact that the counters represent the same value. To see that that the i's are the same suppose $i' \neq i''$. Then both i' and i'' are even as they occur after $\{a,b\}$ and $\{\overline{a},\overline{b}\}$ counters. But then, form the shape of the trace it follows that the positions of $\sigma_j^{i'}$ and $\rho_j^{i''}$ are comparable in the dependency ordering. Hence $\sigma_j^{i'}$ and $\rho_j^{i''}$ cannot be both maximal elements of a configuration.

Next, we can write the formula:

$$\beta(a,\overline{a}) \equiv \mathbb{A}\big(\, same(a,\overline{a}) \Rightarrow \langle\Delta\rangle^n\langle\overline{\Delta}\rangle^n \bigvee_{\sigma\in\Omega}\langle\sigma\rangle\langle\overline{\sigma}\rangle\underline{\mathrm{tt}}\big)$$

This formula says that whenever we see two counters (over $\{a,b\}$ and $\{\overline{a},\overline{b}\}$) representing the same numbers then the letters after them are the same. This way we have $\sigma_j^i = \rho_j^i$ for all even i and all $j \in \{0,\dots,k\}$.

Similarly one can write the formulas $same(c,\overline{c})$ and $\beta(c,\overline{c})$ forcing $\sigma_j^i = \rho_j^i$ for all odd i and all $j \in \{0,\dots,k\}$.

Now, we want to write a formula saying that $\sigma_0^{i+1}\dots\sigma_k^{i+1}$ represents the next configuration after $\rho_0^i\dots\rho_k^i$. For this observe that in order to decide what σ_j^{i+1} should be it is enough to know $\rho_{j-1}^i, \rho_j^i, \rho_{j+1}^i$. We define the formula:

$$Ls(\sigma_1,\sigma_2,\sigma_3) \equiv \langle\Delta\rangle^n\langle\sigma_1\rangle\langle\Delta\rangle^n\langle\sigma_2\rangle\langle\Delta\rangle^n\langle\sigma_3\rangle\underline{\mathrm{tt}}$$

checking that the three consecutive letters from Ω in the upper sequence (i.e., the one written with letters without bars) are $\sigma_1,\sigma_2,\sigma_3$. Similarly we can define $\overline{Ls}$ talking about the lower sequence. Consider the formulas:

$$
\begin{aligned}
\mathit{Step}(c,\overline{a}) &\equiv \mathit{same}(c,\overline{a}) \Rightarrow \gamma_1 \wedge \gamma_2 \wedge \gamma_3 \wedge \gamma_4\\
\gamma_1 &\equiv \bigwedge_{\sigma_1,\sigma_2,\sigma_3\in\Omega-Q} \overline{Ls}(\overline{\sigma}_1,\overline{\sigma}_2,\overline{\sigma}_3) \Rightarrow \langle\Delta\rangle^n\langle\Omega\rangle\langle\Delta\rangle^n\langle\sigma_2\rangle\underline{\mathrm{tt}}\\
\gamma_2 &\equiv \bigwedge_{\sigma_1,\ldots,\sigma_6\in\Omega} (\overline{Ls}(\overline{\sigma}_1,\overline{\sigma}_2,\overline{\sigma}_3) \wedge \mathit{trans}(\sigma_1,\ldots,\sigma_6)) \Rightarrow\\
&\qquad\qquad\qquad\qquad Ls(\sigma_4,\sigma_5,\sigma_6)\\
\gamma_3 &\equiv \langle\overline{\Delta}\rangle^n\langle\overline{\Omega}-\overline{Q}\rangle\langle\overline{\Delta}\rangle^n\langle\overline{\dashv}\rangle \Rightarrow \langle\Delta\rangle^n\langle\Omega\rangle\langle\Delta\rangle^n\langle\dashv\rangle\\
\gamma_4 &\equiv \mathit{first}(\overline{a}) \Rightarrow \langle\Delta\rangle^n\langle\vdash\rangle
\end{aligned}
$$

The shape of the trace guarantees that the only configurations satisfying $\mathit{same}(c,\overline{a})$ are those ending in ρ^i_j and σ^{i+1}_j for some $i\in\mathbb{N}$, $j\in\{0,\ldots,k\}$. Formula γ_1 says that if none of $\rho^i_j,\rho^{i+1}_j,\rho^{i+2}_j$ is a state then σ^{i+1}_j should be the same as ρ^{i+1}_j but without a bar. Formula γ_2 takes care of the case when there is a state: we consult the transition function of M encoded in the formula *trans*. Formula γ_3 assures that the end of configuration marker is copied correctly, similarly γ_4 but for the start of configuration marker.

Finally, we can write a formula *Finish* saying that the automaton has reached the final state and its head is at the leftmost position:

$$\mathit{Finish} \equiv \langle\downarrow\rangle\langle\Delta\rangle^n\langle\vdash\rangle\langle\Delta\rangle^n\langle q_F\rangle\underline{\mathrm{tt}}$$

Of course we can assume that if M accepts then it does so with the head at the leftmost position. Our main formula is:

$$\mathit{Accept} \equiv \mathit{Shape} \wedge \mathit{Init} \wedge \beta(a,\overline{a}) \wedge \beta(c,\overline{c}) \wedge \mathit{Step}(c,\overline{a}) \wedge \mathit{Step}(\overline{c},a) \wedge \mathit{Finish}$$

It can be checked that this formula satisfies property (5). As the size of *Accept* is linear in the size of $|M|+|w|$, we obtain:

Theorem 4 *Let (Σ,I) be a trace alphabet that contains six letters $\{a,b,c,d,\downarrow,\nearrow\}$ with the only dependencies between these letters being those between: (a,b), (c,d), $(\downarrow,a)$, $(\downarrow,b)$, $(\nearrow,c)$, $(\nearrow,d)$, $(\downarrow,\nearrow)$. The satisfiability problem for LTrL$^-$ over (Σ,I) is EXPSPACE-hard.*

In [AMP98] an EXPSPACE upper bound is shown for ISTL$^\diamond$. This logic is very similar to *LTrL*$^-$ but uses atomic propositions and does not have next modalities. It is not clear how to extend the EXPSPACE upper bound to *LTrL* although at first sight one can think that replacing next modalities with propositions should not be very complex. Indeed, it is not known if the extension ISTL$^{\diamond,\bigcirc}$ of ISTL$^{\diamond}$ can be decided in EXPSPACE. We conjecture that both *LTrL*$^{-}$ and ISTL$^{\diamond,\bigcirc}$ can be decided in EXPSPACE.

References

[AMP98] R. Alur, K. McMillan, and D. Peled. Deciding global partial-order properties. In *ICALP'98*, number 1443 in LNCS, pages 41–52, 1998.

[APP95] Rajeev Alur, Doron Peled, and Wojciech Penczek. Model-checking of causality properties. In *LICS '95*, pages 90–100, 1995.

[DG00] Volker Diekert and Paul Gastin. LTL is expressively complete for Mazurkiewicz traces. In *ICALP'00*, volume 1853 of *LNCS*, pages 211–222, 2000.

[DR95] V. Diekert and G. Rozenberg, editors. *The Book of Traces.* World Scientific, 1995.

[EM93] Werner Ebinger and Anca Muscholl. Logical definability on infinite traces. In *ICALP '93*, volume 700, pages 335–346, 1993.

[God96] P. Godefroid. *Partial-order methods for the verification of concurrent systems*, volume 1032 of *LNCS*. Springer-Verlag, 1996.

[KPRP95] K.Lodaya, R. Parikh, R.Ramanujam, and P.S.Thiagarajan. A logical study of distributed transition systems. *Information and Computation*, 119(1):91–118, 1995.

[Nie98] Peter Niebert. *A Temporal Logic for the Specification and Verification of Distributed Behaviour.* PhD thesis, Universität Hildesheim, March 1998. Also available as *Informatik-Bericht Nr. 99-02,Institut für Software, Abteilung Programmierung, Technische Universität Braunschweig, Gaußstraße 11, D-38092 Braunschweig/Germany.*

[Pel96] D. Peled. Partial order reduction : model checking using representatives. In *MFCS'96*, volume 1113 of *LNCS*, pages 93–112, 1996.

[Pen92] Wojciech Penczek. On udecidability of propositional temporal logics on trace systems. *Information Processing Letters*, 43:147–153, 1992.

[Ram96] R. Ramanujam. Locally linear time temporal logic. In *LICS '96*, pages 118–128, 1996.

[SC85] A.P. Sistla and E.M. Clarke. The complexity of propositional linear time logic. *J. ACM*, 32:733–749, 1985.

[Thi94] P. S. Thiagarajan. A trace based extension of linear time temporal logic. In *LICS*, pages 438–447, 1994.

[TW97] P. S. Thiagarajan and I. Walukiewicz. An expressively complete linear time temporal logic for Mazurkiewicz traces. In *LICS'97*, pages 183–194, 1997.

[Val92] A. Valmari. A stubborn attack on state explosion. *Formal Methods in System Design*, 1:297–322, 1992.

[Wal00] Igor Walukiewicz. Local logics for traces. Technical Report RS-00-2, BRICS, Aarhus University, 2000. Extended version to appear in Journal of Automata, Languages and Combinatorics.

A The construction of the formula *counter*(n)

For $i \in \mathbb{N}$ we define $\Sigma_i = \{a_i, b_i\}$ and $\overline{\Sigma}_i = \{\overline{a}_i, \overline{b}_i\}$. For every $n \in \mathbb{N}$, we let $\Delta_n = \bigcup_{i=0,\dots,n} \Sigma_n$ and similarly for $\overline{\Delta}_n$. To define counters of level n we will use the trace alphabet $(\Delta_{n+1} \cup \overline{\Delta}_{n+1}, I)$ where $I = (\Delta_{n+1} \times \overline{\Delta}_{n+1}) \cup (\overline{\Delta}_{n+1} \times \Delta_{n+1})$. We use $a, \overline{a}$ with indices to represent 0 and $b, \overline{b}$ with indices to represent 1. So the sequence $\sigma_0, \dots, \sigma_k$ represents the number $\sum_{i=0}^{k} \sigma_i 2^i$.

A counter of level 0 is a trace with a prefix:

$$\begin{matrix} l_0\sigma \\ \overline{l}_0\rho \end{matrix}$$

where $l_0 \in \Sigma_0$, $\overline{l}_0 \in \overline{\Sigma}_0$, $\sigma \in \Sigma_1$ and $\rho \in \overline{\Sigma}_1$.

A *counter of level* $n \geq 0$ is a trace with the prefix of the form:

$$\begin{matrix} l_0\sigma_0 l_1\sigma_1 \dots l_k\sigma_k\sigma \\ \overline{l}_0\rho_0\overline{l}_1\rho_1 \dots \overline{l}_k\rho_k\rho \end{matrix} \tag{7}$$

where:

- $k = \mathit{Tower}(n,2) - 1$;
- $\sigma \in \Sigma_{n+1}$, $\rho \in \overline{\Sigma}_{n+1}$;
- for each $i = 0, \ldots, k$, the trace $\begin{smallmatrix} l_i & \sigma_i \\ \overline{l}_i & \rho_i \end{smallmatrix}$ is a counter of level $n-1$ with l_i and $\overline{l}_i$ representing the number i.

We will say that the number $\sum_{i=0}^{k} \sigma_i 2^i$ is *represented on the upper line* of the counter and the number $\sum_{i=0}^{k} \rho_i 2^i$ is *represented on the lower line* of the counter.

The following formulas define counters and some auxiliary operations on counters:

$$\begin{aligned}
\mathit{counter}(0) \equiv& \langle \Sigma_0 \rangle \langle \overline{\Sigma}_0 \rangle \langle \Sigma_1 \rangle \langle \overline{\Sigma}_1 \rangle \underline{\mathrm{tt}} \\
\mathit{same}(0) \equiv& \langle a_0 \rangle \langle \overline{a}_0 \rangle \underline{\mathrm{tt}} \vee \langle b_0 \rangle \langle \overline{b}_0 \rangle \underline{\mathrm{tt}} \\
\mathit{next}(0) \equiv& \langle b_0 \rangle \langle \overline{a}_0 \rangle \langle \Sigma_1 \rangle \langle \overline{\Sigma}_1 \rangle \underline{\mathrm{tt}} \\
\mathit{first}(0) \equiv& \langle a_0 \rangle \langle \overline{a}_0 \rangle \langle \Sigma_1 \rangle \langle \overline{\Sigma}_1 \rangle \underline{\mathrm{tt}} \\
\mathrm{last}(0) \equiv& \langle b_0 \rangle \langle \overline{b}_0 \rangle \langle \Sigma_1 \rangle \langle \overline{\Sigma}_1 \rangle \underline{\mathrm{tt}}
\end{aligned}$$

$$
\begin{aligned}
\mathit{counter}(n+1) &\equiv \mathit{first}(n) \wedge (\alpha \;\mathbb{U}\; \langle \Sigma_{n+2}\rangle\langle \overline{\Sigma}_{n+2}\rangle \underline{\mathrm{tt}})\\
\alpha(n) &\equiv (\mathit{counter}(n) \wedge \mathit{same}(n)) \Rightarrow \\
&\qquad \big[(\mathrm{last}(n) \wedge \beta_1(n+1)) \vee (\neg\,\mathrm{last}(n) \wedge \beta_2(n+1))\big]\\
\beta_1(n+1) &\equiv \langle (\Delta_n \cup \overline{\Delta}_n)^*\rangle\langle \Sigma_{n+1}\rangle\langle \overline{\Sigma}_{n+1}\rangle\langle \Sigma_{n+2}\rangle\langle \overline{\Sigma}_{n+2}\rangle \underline{\mathrm{tt}}\\
\beta_2(n+1) &\equiv \langle \Delta_n^*\rangle\langle \Sigma_{n+1}\rangle\big(\,\mathit{next}(n) \wedge \langle \overline{\Delta}_n^*\rangle\langle \overline{\Sigma}_n + 1\rangle\; \mathit{same}(n)\big)\\
\mathit{same}(n+1) &\equiv \mathit{counter}(n+1) \wedge (\delta(n+1) \;\mathbb{U}\; \langle \Sigma_{n+2}\rangle\langle \overline{\Sigma}_{n+2}\rangle \underline{\mathrm{tt}})\\
\delta(n+1) &\equiv \mathit{same}(n) \Rightarrow \langle (\Delta_n \cup \overline{\Delta}_n)^*\rangle \bigvee_{\sigma\in\Sigma_{n+1}} \langle \sigma\rangle\langle \overline{\sigma}\rangle \underline{\mathrm{tt}}\\
\mathit{next}(n+1) &\equiv \mathit{counter}(n+1) \wedge (\mathit{zero}\text{-}\overline{\mathit{one}}(n+1) \;\mathbb{U}\; \gamma)\\
\mathit{zero}\text{-}\overline{\mathit{one}}(n+1) &\equiv \mathit{same}(n) \Rightarrow \langle (\Delta_n \cup \overline{\Delta}_n)^*\rangle\langle a_{n+1}\rangle\langle \overline{b}_{n+1}\rangle \underline{\mathrm{tt}}\\
\gamma(n+1) &\equiv \mathit{same}(n) \wedge \langle (\Delta_n \cup \overline{\Delta}_n)^*\rangle\langle a_{n+1}\rangle\langle b_{n+1}\rangle\; \mathit{same}(n)\\
\mathit{first}(n+1) &\equiv \mathit{only}_a(n+1)\\
\mathrm{last}(n+1) &\equiv \mathit{only}_b(n+1)\\
\mathit{only}_a(n) &\equiv \mathit{counter}(n) \wedge \alpha_1(n) \;\mathbb{U}\; \alpha_2(n)\\
\alpha_1(n) &\equiv \neg\,\mathrm{last}(n-1) \wedge \langle \Delta_{n-1} \cup \overline{\Delta}_{n-1} \cup \{a_n, \overline{a}_n\}\rangle \underline{\mathrm{tt}}\\
\alpha_2(n) &\equiv \mathrm{last}(n-1) \wedge \\
&\qquad \langle (\Delta_{n-1} \cup \overline{\Delta}_{n-1})^*\rangle\langle a_n\rangle\langle \overline{a}_n\rangle\langle \Sigma_{n+1}\rangle\langle \overline{\Sigma}_{n+1}\rangle \underline{\mathrm{tt}}
\end{aligned}
$$

Lemma 5 For every trace T and $n \in \mathbb{N}$:

- $T \models \mathit{counter}(n)$ iff T is an n-level counter.
- $T \models \mathit{same}(n)$ iff T is an n-level counter and the upper and the lower lines of the counter represent the same numbers.
- $T \models \mathit{next}(n)$ iff T is an n-level counter and the upper line represents a number bigger by one than the number represented on the lower line.
- $T \models \mathit{first}(n)$ iff T is an n-level counter and both lines represent the number 0.
- $T \models \mathrm{last}(n)$ iff T is an n-level counter and both lines represent the maximal possible number which is $\mathit{Tower}(n, 2) - 1$.

The size of each of the above mentioned formulas is $2^{\mathcal{O}(n)}$.

thmheadfont Proof
The proof follows basically by inspection of the formulas. Let us consider formula *counter*$(n+1)$. The first conjunct guarantees that the trace starts with $l_0\sigma_0\overline{l}_1\rho_1$. Then, formula α guarantees that whenever we are in a configuration from which we see $l_i\sigma_i\overline{l}_i\rho_i$ then after $l_i\sigma_i$ there is $l_{i+1}\sigma_{i+1}$ (this is done by *next*$(n+1)$ part of β_2 formula) and then we have $\overline{l}_{i+1}\rho_{i+1}$ (this is done by *same*(n) part for β_2 formula). Formula β_1 takes care about the end of the trace.

For the formula *same*$(n+1)$ we know that *same*(n) is true in $(n+1)$-level counter exactly in configurations from which we see $l_i\sigma_i\overline{l}_i\rho_i$. Formula δ requires that ρ_i is the same as σ_i but with a bar on the top.

The formula *next*$(n+1)$ guesses a position i such that $\sigma_i = b$ and $\rho_i = \overline{a}$. Then it checks that for all $j < i$ we have $\sigma_j = a$ and $\rho_j = \overline{b}$ and that for all $j > i$ we have that σ_j is the same as ρ_j modulo bar on the top.

Formulas *first*$(n+1)$ and last$(n+1)$ require that all σ_i and ρ_i for $i = 0, \dots, k$ are a's and b's respectively.

For the size of the formulas, observe that each formula of level $n+1$ contains a constant number of symbols and a constant number of occurrences of formulas of level n. $\square$

B The construction of the formula *Shape*

Let n be fixed (it comes from the size of the input of the machine). A counter over $\{a, b\}$ is a word $w \in \{a, b\}^n$. Similarly for $\{\overline{a}, \overline{b}\}$, $\{c, d\}$ and $\{\overline{c}, \overline{d}\}$ counters. A counter $w_1 \dots w_n$ represents a number $\sum_{i=1}^{k} w_i 2^i$ where a, c stand for 0 and b, d stand for 1. Clearly the biggest number that can be represented is $k = 2^n - 1$. Let l_i^a stand for a counter over $\{a, b\}$ representing number i. Similarly for $l_i^{\overline{a}}$, l_i^c and $l_i^{\overline{c}}$.

The required shape of the trace is the following:

$$\begin{tikzcd}
l_0^a\sigma_0^0 l_1^a\sigma_1^0 \dots l_k^a\sigma_k^0 \downarrow \arrow[d, dashed] & l_0^c\sigma_0^1 l_1^c\sigma_1^1 \dots l_k^c\sigma_k^1 \downarrow \arrow[d, dashed] & \cdots \quad \downarrow \arrow[d, dashed] & l_0^a\sigma_0^i l_1^a\sigma_1^i \dots l_k^a\sigma_k^i \downarrow \arrow[d, dashed] & \cdots \\
l_0^{\overline{a}}\rho_0^0 l_1^{\overline{a}}\rho_1^0 \dots l_k^{\overline{a}}\rho_k^0 \nearrow \arrow[ur, dashed] & l_0^{\overline{c}}\rho_0^1 l_1^{\overline{c}}\rho_1^1 \dots l_k^{\overline{c}}\rho_k^1 \nearrow & \cdots \nearrow \arrow[ur, dashed] & l_0^{\overline{a}}\rho_0^i l_1^{\overline{a}}\rho_1^i \dots l_k^{\overline{a}}\rho_k^i \nearrow & \cdots
\end{tikzcd}$$

where dashed arrows represent additional dependencies and $\sigma_j^i, \rho_j^i \in \Omega$ for all $i \leq 0$ and $j = 0, \dots, k$.

Below we present the formula *Shape* defining traces of the required shape.

$$
\begin{aligned}
\mathit{first}(a) \equiv& \langle a\rangle^n \langle \vdash\rangle\\
\mathrm{last}(a) \equiv& \langle b\rangle^n \langle \Omega\rangle\\
\mathit{next}(a) \equiv& (\mathit{counter}(a) \wedge \neg\,\mathrm{last}(a)) \Rightarrow\\
& \qquad\qquad \mathit{add\text{-}one}(a) \wedge \langle\Delta\rangle^n\langle\Omega\rangle\langle\Delta\rangle^n\langle\Omega\rangle\\
\mathit{add\text{-}one}(a) \equiv& \bigvee_{k=0,\dots,n-2} \langle b\rangle^k \langle a\rangle \underline{\mathrm{tt}} \Rightarrow \langle\Delta\rangle^n\langle\Omega\rangle\langle a\rangle^k\langle b\rangle\underline{\mathrm{tt}}\\
& \wedge \Big[\bigwedge_{i=k,\dots,n-1} (\langle\Delta\rangle^i\langle a\rangle \Rightarrow \langle\Delta\rangle^n\langle\Omega\rangle\langle\Delta\rangle^i\langle a\rangle))\\
& \qquad \wedge ((\langle\Delta\rangle^i\langle b\rangle \Rightarrow \langle\Delta\rangle^n\langle\Omega\rangle\langle\Delta\rangle^i\langle b\rangle))\Big]
\end{aligned}
$$

All the above formulas have their counterparts for $\overline{a}$, c, and $\overline{c}$ instead of a.

$$
\begin{aligned}
\mathit{links} \equiv& \big[\,\mathrm{last}(a) \wedge \mathrm{last}(\overline{a}) \Rightarrow \mathit{ends\text{-}with}(\downarrow, \nearrow, c, \overline{c})\big]\\
& \wedge \big[\,\mathrm{last}(a) \wedge \mathrm{last}(\overline{c}) \Rightarrow \mathit{ends\text{-}with}(\nearrow, \downarrow, c, \overline{a})\big]\\
& \wedge \big[\,\mathrm{last}(c) \wedge \mathrm{last}(\overline{a}) \Rightarrow \mathit{ends\text{-}with}(\nearrow, \downarrow, a, \overline{c})\big]\\
& \wedge \big[\,\mathrm{last}(c) \wedge \mathrm{last}(\overline{c}) \Rightarrow \mathit{ends\text{-}with}(\downarrow, \nearrow, a, \overline{a})\big]\\
\mathit{ends\text{-}with}(\sigma, \rho, d_1, d_2) \equiv& \langle\Delta\rangle^n\langle\overline{\Delta}\rangle^n\langle W\rangle\langle\overline{\Omega}\rangle\langle\sigma\rangle\langle\rho\rangle(\mathit{first}(d_1) \wedge \mathit{first}(d_2))
\end{aligned}
$$

$$
\begin{aligned}
\mathit{Shape} \equiv& \mathit{first}(a) \wedge \mathit{first}(\overline{a}) \wedge\\
& \qquad \mathbb{A}\big(\, \mathit{next}(a) \wedge \mathit{next}(\overline{a}) \wedge \mathit{next}(c) \wedge \mathit{next}(\overline{c}) \wedge \mathit{links}\,\big)
\end{aligned}
$$

Lemma 6 For every trace T: $T \vDash \mathit{Shape}$ iff it is of the form as in (6). The size of *Shape* is $\mathcal{O}(n)$.

thmheadfont Proof
The proof follows from inspection of the formulas. Formula *first*(a) holds in configurations from which starts a counter over $\{a, b\}$ representing 0 followed by the start marker. Similarly, last(a) holds in configurations from which starts a counter representing the maximal value.

The formula *next*(a) says that whenever we see a $\{a, b\}$ counter which does not represent the maximal value then it should be followed by a letter from Ω, a counter representing a number bigger by one, and yet another letter from Ω. The *links* formula takes care of the configurations from which we see the counters with the maximal values. It puts $\nearrow$, $\downarrow$ in appropriate order and starts new counters over appropriate letters. Finally the *Shape* formula forces the presence of two counters representing 0 and then says that every counter which has not reached its maximal value should be followed by a bigger counter. Maximal value counters are dealt with by *links* formula.□