Difficult configurations — on the complexity of *LTrL*

(extended abstract)

Igor Walukiewicz*

Institute of Informatics, Warsaw University
Banacha 2, 02-097 Warsaw, POLAND
igw@mimuw.edu.pl

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-complete.

1 Introduction

Infinite words, or linear orders on *events*, are often used to model executions of systems. Infinite *traces*, or 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 a 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 the formula in a *global logic* is evaluated in a configuration. Global logics have the advantage of talking directly about configurations but, as we show here, have high complexity.

In this paper we investigate the complexity of *LTrL*, a global temporal logic over traces proposed in [12]. We show that the full logic is non-elementary. As it turns out, it is the nesting of the until operators that gives such a high complexity. This makes it natural to ask what is the complexity

*Supported by Polish KBN grant No. 8 T11C 002 11

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-complete. These results give also the bounds on the model checking problem for the logics in question.

Our 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 and remains so if we have "some time in the future" operator instead of the until operator [10]. Local temporal logics for traces proposed in [2, 9, 11] have also PSPACE complexity. It is not known what kinds of global properties are expressible in these local logics. 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 [6, 8]. The most relevant here is the paper by Alur, McMillan and Peled [1]. In this paper an EXPSPACE upper bound is shown for a fragment of ISTL. A modification of their argument shows an EXPSPACE upper bound for $LTrL^-$.

Let us finish this introduction with some remarks showing 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 [3], they are "like words" because most of the properties of words have their counterparts in traces. The generalisation 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 also seem to be of some help in coping with the state explosion problem [13, 5, 7].

If we agree that modelling 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 seems to indicate a strong candidate for the class of properties we want to express – the class of properties expressible in first order logic over traces represented as dependence graphs. This class of properties has many different characterisations [4] 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 achieve this last goal in the trace world it seems agreeable 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 high complexity.

This paper is organized as follows. We start with the necessary definitions and notations. In Section 3 we describe the proof of non-elementary lower bound for *LTrL*. In Section 4 we sketch the proof of the EXPSPACE lower bound for the fragment, $LTrL^-$, of *LTrL*. Because the containment of $LTrL^-$ in EXPSPACE follows from some modifications of the argument in [1] we mention the necessary modifications only in the appendix.

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*. $D = (\Sigma \times \Sigma) - I$ is called the dependency relation.

We shall view (Mazurkiewicz) trace, over an alphabet (Σ, I), as a restricted Σ-labelled poset. Let $(E, \leq, \lambda)$ be a Σ-labelled poset. In other words, $(E, \leq)$ 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 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) $\forall e, e' \in E.\quad e \lessdot e' \Rightarrow \lambda(e) D \lambda(e')$.
(T3) $\forall e, e' \in E.\quad \lambda(e) D \lambda(e') \Rightarrow e \leq e'$ 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 $\mathrm{Conf}(T)$ be the set of configurations of T and let C, C', C'' range over $\mathrm{Conf}(T)$. Note that $\emptyset$, the empty set, is a configuration and $\downarrow e$ is a configuration for every $e \in E$. Finally, the transition relation $\to_T \subseteq \mathrm{Conf}(T) \times \Sigma \times \mathrm{Conf}(T)$ is given by: $C \xrightarrow{a}_T C'$ 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 $C \xrightarrow{a}_T C'$ and $C \xrightarrow{a}_T C''$ then $C' = C''$.

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

$$LTrL(\Sigma, I) ::= \underline{\mathrm{tt}} \mid \sim \alpha \mid \alpha \wedge \beta \mid \langle a \rangle \alpha \mid \alpha \ \mathbb{U}\ \beta \mid \langle a^{-1} \rangle \underline{\mathrm{tt}}$$

Thus the next state modality is indexed by actions. There is also a very restricted version of the previous state modality. This modality will play no role in our considerations, we just mention it in order to be consistent with the original definition.

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

- $T, C \models \underline{\mathrm{tt}}$. Furthermore $\sim$ and $\wedge$ are interpreted in the usual way.
- $T, C \models \langle a \rangle \alpha$ iff $\exists C' \in \mathrm{Conf}(T).\ C \xrightarrow{a}_T C'$ and $T, C' \models \alpha$.
- $T, C \models \alpha \ \mathbb{U}\ \beta$ iff $\exists C' \in \mathrm{Conf}(T).\ C \subseteq C'$ and $T, C' \models \beta$ and $\forall C'' \in \mathrm{Conf}(T).\ C \subseteq C'' \subset C'$ implies $T, C'' \models \alpha$.
- $T, C \models \langle a^{-1} \rangle \underline{\mathrm{tt}}$ iff $\exists C' \in \mathrm{Conf}(T).\ C' \xrightarrow{a}_T C$.

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

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

with the expected semantics:

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

3 The complexity of *LTrL*

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

Given a $\lambda x.\, \mathrm{Tower}(m, x)$ 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 $\lambda x.\, \mathrm{Tower}(m, x)$ for any fixed m. On the other hand the satisfiability problem can be solved in time $\lambda m.\, \mathrm{Tower}(m, 2)$. This follows from the translation of *LTrL* into first order logic over infinite words [12].

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. The actual formulas are presented in the appendix. After this we we will explain how to use big counters to code long computations of Turing machines.

For $i = 0, 1, 2$ let $\Sigma_i = \{a_i, b_i\}$ and let $\overline{\Sigma}_i = \{\overline{a}_i, \overline{b}_i\}$. Let $\Delta = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ 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 have defined above 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 $\overline{e}$ 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:

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

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 $\big(\langle \overline{e} \rangle \underline{\mathrm{tt}} \wedge \langle \Sigma \cup \{e\} \rangle \underline{\mathrm{tt}}\big) \ \mathbb{U}\ \alpha$. Strictly speaking it is equivalent in configurations satisfying $\langle \overline{e} \rangle \underline{\mathrm{tt}}$. 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 the e and $\overline{e}$ letters and use $\langle \Sigma^* \rangle \alpha$ construct as if we had it in our language. To translate what we will write into the language without $\langle \Sigma^* \rangle \alpha$ construct one has to add the formula forcing occurrences of e and $\overline{e}$, replace each $\langle \Sigma^* \rangle \alpha$ by its definition and then replace each $\langle a \rangle \beta$ by $\langle ea \rangle \beta$; moreover some care is also needed with the until operator (we skip the details).

After this preliminary remarks about the language we will use, 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 $\sum_{i=0,\ldots,n-1} \sigma_i 2^i$. (Please note that the most significant digit is to the right.) Similarly an element of $(\overline{\Sigma}_0)^n$ can be considered as a counter.

Consider the following formulas:

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

Please 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 $\mathrm{counter}_0$ 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 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 $\mathrm{counter}_1$ forcing the beginning of a trace to consist of two exponentially long counters, i.e., the beginning of a trace will have the form

$$\begin{array}{l} l_0\sigma_0 l_1\sigma_1 \dots l_k\sigma_k\sigma \\ \bar{l}_0\overline{\rho}_0\bar{l}_1\overline{\rho}_1\dots\bar{l}_k\overline{\rho}_k\overline{\rho} \end{array} \tag{1}$$

where $k = 2^n - 1$; $l_0, \dots, l_k \in (\Sigma_0)^n$ and $\bar{l}_0, \dots, \bar{l}_k \in (\overline{\Sigma}_0)^n$ 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 $i = 0, \dots, k$).

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

The formula $\mathrm{counter}_1$ says that a trace should begin with two level 0 counters representing 0 followed by σ_0 and ρ_0; the rest of the formula 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 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 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 counter as in (1) represents the number $\sum_{i=0,\dots,k} \sigma_i 2^i$. The value of the counter consisting of ρ's can be forced to be the same by the formula same_1:

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

With a slightly more elaborate formula, next_1, we can force the value of the counter represented by σ's to be bigger by one than the value of the counter represented by ρ's.

This way we have obtained exponentially bigger counters than the ones we have started with. We have also means to compare and add one to these counters. Clearly we can also write formulas saying that the value of such a counter is 0 or the maximal possible. At this point we can iterate the whole construction to get even bigger counters.

For describing inductive construction of bigger and bigger counters we need a precise definition 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* $n \geq 0$ is a trace 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 $k = \mathrm{Tower}(n-1, 2) - 1$. 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 $\sigma \in \Sigma_{n+1}$, $\rho \in \overline{\Sigma}_{n+1}$.

In Appendix A we show how to write formulas: (i) $\mathrm{counter}(n)$ saying that a trace starts with the counter of level n; (ii) $\mathrm{same}(n)$ saying that if a trace is of the form (2) then $\sigma_0, \dots, \sigma_k$ and $\rho_0, \dots, \rho_k$ represent the same numbers; (iii) $\mathrm{next}(n)$ saying that the number represented by $\sigma_0, \dots, \sigma_k$ is one plus the number represented by $\rho_0, \dots, \rho_k$.

Let $m \geq 0$ and let M be a $\lambda x. \mathrm{Tower}(m, x)$ 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 $\mathrm{Tower}(m, |w|)$, so we need counters able to count up to this number. Let n be the smallest number such that $\mathrm{Tower}(n, 2) \geq \mathrm{Tower}(m, |w|)$. 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 $\vdash vqv' \dashv$ 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 $\vdash$ and $\dashv$ as end markers. Let $\Omega = Q \cup \Gamma \cup \{\vdash, \dashv\}$ be the alphabet needed to write down configurations. In the formulas below we use Δ_n for $\bigcup_{i=0,\dots,n} \Sigma_n$. As before $\overline{\Delta}_n$ denotes the appropriate set of letters with bars.

Summarising, 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.

First step in constructing the formula Accept is to write a formula Conf saying that a prefix of a trace is of the form as in (2) but the letters σ_i come from Ω, letters ρ_i come from $\overline{\Omega}$, σ is the symbol \$, and ρ is the symbol $\overline{\$}$. This formula can be obtained by a simple modification of the formula counter$(n+1)$ (replace Σ_{n+1} by Ω and Σ_{n+2} by $\{\$\}$).

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

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

The formula Step, that we have not presented, looks at the three consecutive letters of the lower configuration (i.e., the configuration written in $\overline{\Omega}$) 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:

$$\text{Accept} \equiv \text{Init} \wedge \text{Comp}$$

Lemma 1 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 counter$(n+1)$ hence of the size $2^{\mathcal{O}(n)}$ (see Lemma 6 in Appendix A). Also 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 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.

Summarising, for a $\lambda x.\,\mathrm{Tower}(m,x)$ 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 2
Let (Σ, I) be a trace alphabet containing four 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 of 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 gives us the power to reach non-elementary complexity. In this section we will deal with *LTrL* without until. Instead of until we will allow "sometime" and "always" modalities ($\mathbb{E}$, $\mathbb{A}$) 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 good 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 four counter alphabets: $\{a, b\}$, $\{c, d\}$, $\{\overline{a}, \overline{b}\}$ and $\{\overline{c}, \overline{d}\}$. A counter is a word over one of these alphabets. The interpretation is that a and c stand for 0 and b and d stand for 1.

Let M be a 2^n 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 $w \in$

Γ^* be a word, let n be the length of w and let $k = 2^n - 1$. The configuration 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 appropriate 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 depicted in Figure 1. The dashed ar-

$$\begin{array}{lllllll}
l_0^a\sigma_0^0 l_1^a\sigma_1^0 \ldots l_k^a\sigma_k^0 \downarrow & l_0^c\sigma_0^1 l_1^c\sigma_1^1 \ldots l_k^c\sigma_k^1 \downarrow & \cdots & \downarrow l_0^a\sigma_0^i l_1^a\sigma_1^i \ldots l_k^a\sigma_k^i \downarrow & & \cdots \\
l_0^{\overline{a}}\rho_0^0 l_1^{\overline{a}}\rho_1^0 \ldots l_k^{\overline{a}}\rho_k^0 \nearrow & l_0^{\overline{c}}\rho_0^1 l_1^{\overline{c}}\rho_1^1 \ldots l_k^{\overline{c}}\rho_k^1 \nearrow & \cdots \nearrow & l_0^{\overline{a}}\rho_0^i l_1^{\overline{a}}\rho_1^i \ldots l_k^{\overline{a}}\rho_k^i \nearrow & & \cdots
\end{array}$$

Figure 1: The trace shape

rows represent additional dependencies and the meanings of the components is the following. For every $i = 1, \ldots, k$, $l_i^a \in \{a, b\}^n$ is a counter representing number i; similarly for $l_i^{\overline{a}}$, l_i^c $l_i^{\overline{c}}$. Letters $\sigma_j^i, \rho_j^i \in \Omega$ describe configurations. Letters $\downarrow, \nearrow$ are used to force synchronisation 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, $\overline{e}$ actions. To the shape of trace presented in Figure 1 we should add that every second action in the upper sequence is e and every second action in the lower sequence is $\overline{e}$. Having this we can define $\langle\Sigma\rangle\alpha$ construction by $\langle\cdot\rangle(\alpha \wedge \langle\overline{e}\rangle\underline{\mathrm{tt}}) \wedge \bigwedge_{a \notin \Sigma} \neg\langle a\rangle\underline{\mathrm{tt}}$. Once again 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. Also following 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.

In the appendix we describe how to construct a formula Shape forcing a trace to be of the form presented in Figure 1.

We can write a formula Init saying that $\sigma_0^0 \ldots \sigma_{n+3}^0$ form the initial configuration of M on w. 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 $\dashv$.

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 formulas:

$$\begin{aligned}
\mathrm{same}(a,\overline{a}) &\equiv \bigwedge_{i=0,\dots,n-1} \langle\Delta\rangle^i \langle\overline{\Delta}\rangle^i (\langle a\rangle\langle\overline{a}\rangle\underline{\mathrm{tt}} \vee \langle b\rangle\langle\overline{b}\rangle\underline{\mathrm{tt}}) \\
\beta(a,\overline{a}) &\equiv \mathbb{A}\big(\,\mathrm{same}(a,\overline{a}) \Rightarrow \langle\Delta\rangle^n\langle\overline{\Delta}\rangle^n \bigvee_{\sigma\in\Omega} \langle\sigma\rangle\langle\overline{\sigma}\rangle\underline{\mathrm{tt}}\big)
\end{aligned}$$

The formula $\mathrm{same}(a,\overline{a})$ 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 synchronisation letters $\downarrow, \nearrow$. Because of the structure of the trace forced by these letters, if some configuration satisfies $\mathrm{same}(a,\overline{a})$ then this configuration must be necessary of the form symbolised by the thick vertical line:

$$\begin{array}{l|l}
l_0^a\sigma_0^0 l_1^a\sigma_1^0 \dots l_k^a\sigma_k^0 \downarrow\ l_0^c\sigma_0^1 l_1^c\sigma_1^1 \dots l_k^c\sigma_k^1 \downarrow\ \cdots \downarrow l_0^a\sigma_0^i l_1^a\sigma_1^i \dots & l_j^a\sigma_j^i \dots l_k^a\sigma_k^i \downarrow \cdots \\
l_0^{\overline{a}}\rho_0^0 l_1^{\overline{a}}\rho_1^0 \dots l_k^{\overline{a}}\rho_k^0 \nearrow\ l_0^{\overline{c}}\rho_0^1 l_1^{\overline{c}}\rho_1^1 \dots l_k^{\overline{c}}\rho_k^1 \nearrow\ \cdots \nearrow l_0^{\overline{a}}\rho_0^i l_1^{\overline{a}}\rho_1^i \dots & l_j^{\overline{a}}\rho_j^i \dots l_k^{\overline{a}}\rho_k^i \nearrow \cdots
\end{array}$$

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. That the i's are the same is due to the fact that if $i' \neq i''$ then the positions of $\sigma_j^{i'}$ and $\rho_j^{i''}$ are comparable in the dependency ordering of the above trace.

Formula $\beta(a,\overline{a})$ says that whenever we see two counters 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 $\mathrm{same}(c,\overline{c})$ 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 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:

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

Formula γ_1 says that for every position not containing a state on the neighbouring positions the letters should be the same. 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 starting marker. Once again the shape of the trace guarantees that the only configurations satisfying $\mathrm{same}(c,\overline{a})$ are those ending in ρ^i_j and σ^{i+1}_j for some $i \in \mathbb{N}$, $j \in \{0,\dots,k\}$.

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

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

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

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

Lemma 3 Formula Accept is satisfiable iff $w \in L(M)$.

As the size of Accept is linear in the size of $|M| + |w|$ we obtain:

Theorem 4
Let (Σ, I) be a trace alphabet containing 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.

A modification of the argument from [1] shows:

Theorem 5
For arbitrary trace alphabet, LTrL$^-$ is in EXPSPACE.

The proof of this theorem is outlined in the appendix.

Acknowledgements

I thank Doron Peled for giving me his paper [1]. I also thank Manfred Droste for an invitation to a very inspiring workshop on traces.

References

[1] R. Alur, K. McMillan, and D. Peled. Deciding global parial-order properties. submitted, 1997.

[2] R. Alur, D. Peled, and W. Penczek. Model-checking of causality properties. In *LICS '95*, pages 90–100, 1995.

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

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

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

[6] K. Lodaya, R. Parikh, R. Ramanujam, and P. Thiagarajan. A logical study of distributed transition systems. *Information and Computation*, 119:91–118, 1985.

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

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

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

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

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

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

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

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 $\Delta_n = \bigcup_{i=0,\ldots,n} \Sigma_n$ 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 $I = (\Delta_{n+1} \times \overline{\Delta}_{n+1}) \cup (\overline{\Delta}_{n+1} \times \Delta_{n+1})$.

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

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

$$
\begin{aligned}
\mathrm{only}_a(n) \equiv& \,\mathrm{counter}(n) \wedge \alpha_1(n) \;\mathbb{U}\; \alpha_2(n) \\
\alpha_1(n) \equiv& \neg\, \mathrm{last}(n-1) \wedge \langle \Delta_{n-1} \cup \overline{\Delta}_{n-1} \cup \{a_n, \overline{a}_n\} \rangle \underline{\mathrm{tt}} \\
\alpha_2(n) \equiv& \,\mathrm{last}(n-1) \wedge \langle (\Delta_{n-1} \cup \overline{\Delta}_{n-1})^* \rangle \langle a_n \rangle \langle \overline{a}_n \rangle \langle \Sigma_{n+1} \rangle \langle \overline{\Sigma}_{n+1} \rangle \underline{\mathrm{tt}} \\
\mathrm{first}(n) \equiv& \,\mathrm{only}_a(n) \\
\mathrm{last}(n) \equiv& \,\mathrm{only}_b(n)
\end{aligned}
$$

Lemma 6 For every trace T and $n \in \mathbb{N}$. Prefix of T is a counter of level n iff $T \models \mathrm{counter}(n)$. The size of the formula counter(n) is $2^{\mathcal{O}(n)}$.

B The construction of the formula Shape

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

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

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

$$
\mathrm{Shape} \equiv \mathrm{first}(a) \wedge \mathrm{first}(\overline{a}) \wedge \mathbb{A}\big(\,\mathrm{next}(a) \wedge \mathrm{next}(\overline{a}) \wedge \mathrm{next}(c) \wedge \mathrm{next}(\overline{c}) \wedge \mathrm{links}\big)
$$

Lemma 7 For every trace T: $T \vDash \mathrm{Shape}$ iff it is of the form as in Figure 1.

C An EXPSPACE algorithm for $LTrL^-$

In this section we will present an EXPSPACE algorithm for deciding satisfiability of $LTrL^-$ formulas. This way we obtain that for the alphabets as in Theorem 4 $LTrL^-$ is EXPSPACE complete. The proof is by constructing a double exponential size automaton recognising linearizations of traces satisfying a given formula. The standard trick of constructing states of an automaton on demand allows then to reduce the complexity to EXPSPACE. For the rest of this section let us fix a trace alphabet (Σ, I).

A *generalized configuration* is a, not necessary finite, downwards closed set of events. We will sometimes omit the word generalized. It should be not difficult to guess from the context what kind of configuration we mean. In particular the meaning of a formula is still defined only for finite configurations.

Let T be a trace and $\mathbb{E}\alpha$ a formula. A *maximal generalized configuration* of $\mathbb{E}\alpha$ in T is a generalized configuration C such that for every other configuration C': if $C' \subseteq C$ then $T, C' \vDash \mathbb{E}\alpha$; and if $C \subseteq C'$ then $T, C' \nvDash \mathbb{E}\alpha$. Clearly there may be more than one maximal configuration. If $\mathbb{E}\alpha$ is satisfied in T then it has a maximal configuration – in some cases it may be infinite.

The main idea, that comes from [1], is to convert a given formula into an, exponential size, disjunction of normal form formulas. Normal form formulas can have at most one maximal generalized configuration in every trace so they will be easier to check. It will be possible to convert such a disjunction into an equivalent automaton with a single exponential blowup.

Definition 8 Let $\approx$ and $\prec$ stand for, respectively: trace equivalence relation and trace prefix relation on Σ^*. (Please recall that we have fixed a trace alphabet (Σ, I).)

For $k \in \mathbb{N}$, a *k-prefix* of a trace T is any linearisation (respecting the dependency ordering of T) of a finite subtrace of T consisting of all the events $e \in T$ with $\downarrow e$ containing no more than k events.

Let Γ be a, finite, set of $\mathbb{E}$-formulas (formulas starting with $\mathbb{E}$) and $w \in \Sigma^*$. A *w, Γ-type* is a set of formulas containing the formula $\langle w \rangle \underline{tt}$ and, for every $v \prec w$, a formula of the form $\langle v \rangle (\bigwedge \Delta \wedge \neg \bigwedge \Delta')$ with $\Delta, \Delta' \subseteq \Gamma$ being a partition of Γ.

A *k, Γ-type* of a trace T is a w, Γ-type θ such that $T \vDash \theta$, where w is a k-prefix of T. (Recall that $T \vDash \theta$ is an abbreviation of $T, \emptyset \vDash \theta$.)

Lemma 9 For every T and k, a k, Γ-trace of T is unique up to equivalence of formulas.

For a w, Γ-type θ and $a \in \Sigma$, let $\theta \downarrow a$ denote a set:

$$\theta \downarrow a = \{\langle v \rangle \gamma : \exists v'. v' \approx av \text{ and } \langle v' \rangle \gamma \in \theta\}$$

It is easy to see that $\theta \downarrow a$ is a w'-type for w' such that $aw' \approx w$. Moreover if θ is a k, Γ-type of some trace T and $T \vDash \langle a \rangle \underline{tt}$ then $\theta \downarrow a$ is a, superset of, $(k-1), \Gamma$-type of T. If $T \nvDash \langle a \rangle \underline{tt}$ then $\theta \downarrow a$ is equivalent to $\underline{ff}$.

Lemma 10 For every trace T and $k \in \mathbb{N}$, the size of k, Γ-type of T is at most $\mathcal{O}(k|\Gamma|)$.

Lemma 11 Let $\alpha(\Gamma)$ be a formula with no $\mathbb{E}, \mathbb{A}$ operators apart from those in Γ. Formula $\alpha(\Gamma)$ is equivalent to a disjunction of k, Γ-types, where k is the size of α.

Proof
Define $\varphi = \bigvee\{\theta : \ \theta \Rightarrow \alpha(\Gamma) \text{ and } \theta \text{ a } k,\Gamma\text{-type}\}$; here by writing $\theta \Rightarrow \beta$ we mean that this implication is true. We claim that φ is the disjunction required in the lemma.

If $T \vDash \varphi$ then $T \vDash \theta$ for some disjunct θ of φ. Hence by the definition of φ we have $T \vDash \alpha(\Gamma)$. The proof in the opposite direction follows from:

Claim 11.1 If $T \vDash \beta(\Gamma)$ then $\theta \Rightarrow \beta(\Gamma)$ where θ is the k,Γ-type of T and k is at least the size of β.

The proof of the claim is by induction on the structure of β.

Suppose $\beta(\Gamma)$ is a formula from Γ or its negation. In this case we have $\beta(\Gamma) \in \Delta$ for the formula $\bigwedge \Delta \in \theta$.

Suppose $\beta \equiv \beta_1 \wedge \beta_2$. If $T \vDash \beta_1 \wedge \beta_2$ then $T \vDash \beta_1$ and $T \vDash \beta_2$. By the induction assumption $\theta \Rightarrow \beta_1$ and $\theta \Rightarrow \beta_2$, for θ being k,Γ-type of θ.

Suppose $\beta \equiv \beta_1 \vee \beta_2$. If $T \vDash \beta_1$ then, by the induction assumption, $\theta \Rightarrow \beta_1$, for θ being k,Γ-type of θ. Similarly if $T \vDash \beta_2$.

Suppose $\beta = \langle a\rangle\beta_1$. If $T \vDash \langle a\rangle\beta_1$ then $T,\{a\} \vDash \beta_1$. Let θ be the k,Γ-type of T. Clearly $T,\{a\} \vDash \theta\downarrow a$. By induction hypothesis $\theta\downarrow a \Rightarrow \beta_1$. Hence $\langle a\rangle\theta\downarrow a \Rightarrow \langle a\rangle\beta_1$. Finally $\theta \Rightarrow \langle a\rangle\theta\downarrow a$.

Suppose $\beta = [a]\beta_1$. Let θ be the k,Γ-type of T. Assume $T \vDash [a]\underline{\mathrm{ff}}$. As $k > 1$ and $T \vDash \theta$ we have $T \vDash \langle w\rangle\underline{\mathrm{tt}}$ for w the k-prefix of T. By the definition of k-prefix we have $\langle w\rangle\underline{\mathrm{tt}} \Rightarrow [a]\underline{\mathrm{ff}}$. Now assume $T \nvDash [a]\underline{\mathrm{ff}}$. We have $T \vDash \langle a\rangle\beta_1$ and from the above $\theta \Rightarrow \langle a\rangle\beta_1$. Hence $\theta \Rightarrow [a]\beta_1$

Suppose $\beta = \langle\cdot\rangle\beta_1$. If $T \vDash \langle\cdot\rangle\beta_1$ then $T \vDash \langle a\rangle\beta_1$ for some a. Hence $\theta \Rightarrow \langle a\rangle\beta_1$ and $\langle a\rangle\beta_1 \Rightarrow \langle\cdot\rangle\beta_1$ for θ the k,Γ-type of T.

Suppose $\beta = [\cdot]\beta_1$. Assume $T \vDash [\cdot]\beta_1$. Let θ be the k,Γ-type of T. If, for some a, $T \vDash \langle a\rangle\underline{\mathrm{tt}}$ then $T,\{a\} \vDash \beta_1$. Hence $\theta\downarrow a \Rightarrow \beta_1$ by the induction assumption. If $T \nvDash \langle a\rangle\underline{\mathrm{tt}}$ then $\theta\downarrow a$ is equivalent to $\underline{\mathrm{ff}}$. So for every $a \in \Sigma$ we have $[a]\theta\downarrow a \Rightarrow [a]\beta_1$. This gives us $\bigwedge_{a\in\Sigma}[a]\theta\downarrow a \Rightarrow \bigwedge_{a\in\Sigma}[a]\beta_1 \Rightarrow [\cdot]\beta_1$. Finally $\theta \Rightarrow [a]\theta\downarrow a$ for every $a \in \Sigma$. $\square$

Our primary interest will be in formulas in normal form. These formulas will have the property that they will have at most one maximal generalized configuration in every trace.

Definition 12 (Normal formulas) The set of *normal formulas* (NF) is the smallest set containing $\underline{\mathrm{tt}}, \underline{\mathrm{ff}}$ and such that whenever $\alpha, \beta \in NF$ then $\alpha \wedge \beta$, $\langle a\rangle\alpha$, $\mathbb{E}\varphi$ and $\neg\mathbb{E}\varphi$ are in NF.

Please observe that there are no disjunctions, $[a]$ modalities or $\langle\cdot\rangle$ modalities in normal formulas.

Lemma 13 Let α be a normal form formula and let T be a trace. There is at most one maximal generalized configuration of α in T.

Using distributivity of disjunction it is not difficult to show that every formula is equivalent to a disjunction of normal form formulas. We are now going to describe the construction of an automaton accepting linearizations of traces satisfying a given formula. Before this we will need a notion of automaton signalling a generalized configuration.

Let T be a trace and C its generalized configuration. Given a linearization w of T one can *mark out* C from w by marking those positions in w that correspond to the minimal events not in C. This convention is technically easier than, maybe a more natural one, to mark out the maximal events of C. For example to mark out the empty configuration, using our convention we should mark all first events in T.

Definition 14 For automaton to *signal a genralized configuration* it must have a special initial state $q_{\underline{\mathrm{ff}}}$, a special set of states Q_S and a special state q_{end}. If the automaton chooses to start a run from $q_{\underline{\mathrm{ff}}}$ then it signals that there is no maximal configuration in the given trace. If the automaton starts a run from some different initial state and reaches a state from Q_S on reading some letter then it means that the position of this letter is a minimal position outside the configuration. When the last such letter is reached the automaton should enter into the state q_{end}.

The EXPSPACE algorithm for $LTrL^-$ follows from:

Lemma 15 Every $LTrL^-$ formula $\mathbb{E}\alpha$ is equivalent to a disjunction $\bigvee_{i\in I}\alpha_i$ of at most $2^{\mathcal{O}(|\alpha|)}$ normal formulas of size $2^{\mathcal{O}(|\alpha|)}$. For every α_i there exists an automaton $\mathcal{A}_i$ signalling the maximal configurations of α_i, this automaton has the size $2^{\mathcal{O}(\alpha_i)}$

Proof

Without a loss of generality we can assume that there are no negations in α apart from those standing just before $\mathbb{E}$, moreover we assume that there are no $\mathbb{A}$ operators in α. This can be obtained by repeatedly replacing $\mathbb{A}\beta$ subformulas by $\neg\mathbb{E}\neg\beta$ and pushing negation downwards.

Now suppose α is of the form $\alpha'(\Gamma)$ where $\Gamma = \{\gamma_1 \dots, \gamma_n\}$ are all the maximal $\mathbb{E}$ subformulas. Hence there is no $\mathbb{E}$ operator in $\alpha'(\Gamma)$ apart from those in some of the Γ. Of course Γ may be empty.

By induction assumption, each formula γ_i is equivalent to a disjunction $\bigvee_{j\in J_i}\gamma_i^j$ of normal formulas. For each of the disjuncts there is an automaton signalling the maximal configuration of γ_i^j.

From Lemma 11, the formula $\alpha'(\Gamma)$ is equivalent to a disjunction $\bigvee_{i\in I}\theta_i(\Gamma)$. There are at most $2^{\mathcal{O}(|\alpha'|)}$ disjuncts and each disjunct is of size $\mathcal{O}(|\alpha'|)$.

Let us take one such disjunct $\theta(\gamma_1,\dots,\gamma_n)$. We replace each formula γ_i by a disjunction of normal formulas $\theta(\bigvee_{j\in J_1}\gamma_1^j,\dots,\bigvee_{j\in J_n}\gamma_n^j)$. For convenience of notation assume that $\gamma_1,\dots,\gamma_k$ are not negated in θ and $\gamma_{k+1},\dots,\gamma_n$

are. Using distributivity of disjunction we obtain that $\theta(\Gamma)$ is equivalent to $\bigvee_{j1\in J_1,\dots,jk\in J_k} \theta'(\gamma_1^{j1},\dots,\gamma_k^{jk},\bigwedge_{j\in J_{k+1}} \neg\gamma_{k+1}^j,\dots,\bigwedge_{j\in J_n}\neg\gamma_n^j)$ where θ' is obtained from θ by deleting negations. This is a disjunction of normal form formulas. All together we have:

$$\alpha'(\Gamma) \equiv \bigvee_{i\in I} \bigvee_{j1\in J_1,\dots,jk\in J_k} \theta'(\gamma_1^{j1},\dots,\gamma_k^{jk}, \bigwedge_{j\in J_{k+1}} \neg\gamma_{k+1}^j,\dots, \bigwedge_{j\in J_n}\neg\gamma_n^j)$$

This formula consist of $2^{\mathcal{O}(|\alpha'|)}2^{\mathcal{O}(|\gamma_1|)}\dots 2^{\mathcal{O}(|\gamma_k|)} \leq 2^{\mathcal{O}(|\alpha|)}$ disjuncts, each of size $2^{\mathcal{O}(|\alpha|}$.

Let us take one such disjunct $\beta(\delta_1,\dots,\delta_k,\bigwedge_{i\in I_{k+1}}\neg\delta_i\dots,\bigwedge_{i\in I_n}\neg\delta_j)$ (here we made some reindexing to simplify the notation). By induction assumption for every δ_i there is an automaton $\mathcal{A}_i$ signalling the maximal configuration of δ_i. The automaton for $\mathcal{B}$ will contain all these automata and one more component for β.

To find the maximal configuration $\max_\beta$ for β the automaton does the following:

1. Finds the maximal configuration $\max_{\underline{\mathtt{tt}}}$ for $\beta(\delta_1,\dots,\delta_k,\underline{\mathtt{tt}},\dots,\underline{\mathtt{tt}})$.
2. If it does not exist then $\max_\beta$ does not exist.
3. If $\max_{\underline{\mathtt{tt}}}$ is finite then it checks whether it satisfies
$$\beta(\delta_1,\dots,\delta_k, \bigwedge_{i\in I_{k+1}}\neg\delta_i\dots, \bigwedge_{i\in I_n}\neg\delta_j)$$
4. If yes then $\max_\beta$ is $\max_{\underline{\mathtt{tt}}}$. If no then $\max_\beta$ does not exist.
5. If $\max_{\underline{\mathtt{tt}}}$ is infinite then it checks if there exist a finite configuration C with:
$$C \vDash \delta_1\wedge\dots\wedge\delta_k\wedge \bigwedge_{j\in J_{k+1}}\neg\delta^j\wedge\dots\wedge \bigwedge_{j\in J_{k+1}}\neg\delta^j$$
6. If yes then $\max_\beta$ is $\max_{\underline{\mathtt{tt}}}$. If no then $\max_\beta$ does not exist.

The size of states of the automaton $\mathcal{B}$ doing what we have described above is $\mathcal{O}(|\beta|+\sum_{i=1,\dots,n}2^{\mathcal{O}(|\gamma_i|)}) \leq 2^{\mathcal{O}(|\alpha|)}$. $\square$