A linear temporal logic model checking method over finite words with correlated transition attributes

Jean Michel Couvreur and Joaquín Ezpeleta

[1] Laboratoire d'Informatique Fondamental d'Orléans (LIFO)
Université d'Orléans jean-michel.couvreur@univ-orleans.fr
[2] Aragón Institute of Engineering Research (I3A)
Dept. of Computer Sciences and Systems Engineering
University of Zaragoza, Spain ezpeleta@unizar.es

Abstract. Temporal logic model checking techniques are applied, in a natural way, to the analysis of the set of finite traces composing a system log. The specific nature of such traces helps in adapting traditional techniques in order to extend their analysis capabilities. The paper presents an adaption of the classical Timed Propositional Temporal Logic to the case of finite words and considers relations among different attributes corresponding to different events. The introduced approach allows the use of general relations between event attributes by means of freeze quantifiers as well as future and past temporal operators. The paper also presents a decision procedure, as well as a study of its computational complexity.

Keywords: Log analysis, Model checking, Freeze Linear Temporal Logic

1 Introduction

Current information systems usually generate log files to record system and user activities. System logs contain very valuable information that, when properly analyzed, could help in getting a better understanding of the system and user behaviors and then in improving the system. In many (most) cases the log can be seen as a set of *traces*: a trace is a subset of sequentially ordered events which correspond to a process execution. It can correspond, for instance, to the events of a user session in an e-commerce website or database, the events corresponding to the execution of a process in a workflow system, etc.

Process mining [1] is the set of techniques that try to analyze log files looking for trace patterns so as to synthesize a model representing the set of event sequences in the log. In some cases, when the system is governed by a rather closed procedural approach, the model itself can be a quite constraining and closed model (Petri net, BPMN process, etc.) fitting the log. In cases in which the system does not really constrain the user possibilities (open systems), such

constraining models are not really useful (since there can be so many different combinations of event sequences in the log that the obtained model will be a kind of *spaghetti* or *flower* model, depending on the variety of such combinations). For the last cases, a viable approach consists of establishing a set of behavioral properties (described in a high level formalism, such as temporal logic) describing possible model behaviors and then checking which traces in the log satisfy them, looking for what are usually called a *declarative process* (described in an implicit way by the set of formulas).

Conformance checking is the process by which a log and a model (either a procedural model or a declarative one) are compared, so as to get a measure of how well the log and the model are aligned. This paper concentrates on the conformance perspective using a variant of temporal logic for property description and a model checker for conformance checking. Temporal logic has been extensively used in process mining [2, 3]. Initially, only control flow aspects were considered. A case (trace) was defined as an ordered sequence of activities. Later, a multi-perspective point of view was adopted. In this case, each event, besides the activity, could contain additional data, as the time at which the event happened, the resource that executed the activity, the duration, etc. [4–7]. As shown in [6], conformance results can significantly vary when data associated to activities in events is considered: the data perspective should be considered at the same level than the control perspective in order to obtain an accurate model.

Classical LTL temporal logic for declarative process conformance imposes some constraints with respect to the kind of properties one can deal with. Let us consider, as an example, a trace whose events are of the form (ac, re, ts), where ac stands for the activity, re for the resource that executed it and ts for the event time-stamp. It is possible to express by means of a classical LTL formula the property that a concrete activity a executed by a concrete resource r is always (G temporal logic operator) followed by a future (F temporal logic operator) concrete activity b executed by the same concrete resource r: $G((a, r) \rightarrow F((b, r)))$. However, it is not possible to express the property that a concrete activity a is followed in the future by activity b, and both activities are carried out by the same resource (being the resource "any" resource). In the case of finite domains one can transform such formula into the disjunction of a set of formulas (one per resource). However, this is infeasible for dense data domains. Consider, for instance, the necessity of correlating the times at which the considered events happened, so as to ensure that both are in an interval of 30 minutes.

To consider such situations in log analysis, different model checking techniques have been proposed. The work in [8] gives a big step forward towards the full integration of the control and data perspectives, considering the time as a special part of the data associated with events. Authors use Metric First Order Temporal Logic (MFOTL) [9, 10] for the specification of behavioral properties, and propose model checking functions for a subset of MP-Declare[7] patterns.

To alleviate some of the constrains of the preceding methods, in this paper we propose the DLTL temporal logic as an adaption of the *Timed Proposition*

Temporal Logic, TPTL [11], which allows a real integration of the data, control and time perspectives from both, future and past perspectives. The way the logic is defined allows working with any data attributes associated to events as well as general relations among them. The approach can be considered as an integrated multi-perspective conformance checking method. The main contributions in the paper are: 1) the proposal of the DLTL temporal logic able to deal with a whole multi-perspective point of view; 2) the proposal of a model checking algorithm for such logic, with no constrain about the set of formulas that can be analyzed and 3) the space and time complexity characterization of the proposed model checking method.

The paper is organized as follows. Section 2 formally defines the logic and also describes it by means of some intuitive examples. Section 3 proposes a model checking algorithm and evaluates its complexity in time and space. Section 4 shows how the proposed logic and model checking can be applied to the analysis of a log corresponding to a workflow system, used in the literature. Section 5 briefly describes a model checker prototype. Section 6 comments on some related work which concentrate on (timed) temporal logic and model checking approaches. Finally Section 7 establishes some conclusions of the work and gives some future perspectives for its continuation.

2 DLTL

The logic we propose in the paper is based on the the Timed Propositional Temporal Logic, TPTL, [11]. TPTL is a very elegant formalism which extends classical linear temporal logic with a special form of quantification, named *freeze quantification*.

Every freeze quantifier is bound to the time of a particular state. As an example, the property "whenever there is a request p, and variable x is frozen to the current state, the request is followed by a response q at time y, so that y is at most, x+10" is expressed in TPTL by the formula $Gx.(p \rightarrow Fy.(q \wedge (y \leq x + 10)))$ [11].

A TPTL formula can contain as many freeze operators as required. For instance, the following formula states that "every p-state is followed by a q-state which itself is followed by an r-state, and the time difference between the p and r states is at most 10": $Gx.(p \rightarrow F(q \wedge F(y.(r \wedge y \leq x + 10))))$. In this case, since we need to talk about two different points in the trace (p and q states), we use two freeze variables in order to be able to correlate the time values of those states.

The adaption of TPTL that we propose focuses on two different aspects. On the one hand, we generalize the kind of relations between the attributes of the events corresponding to freeze operators. TPTL constrained event correlations to check whether the difference between the attribute values of two freeze variables (positions in the word) belonged to a certain interval. In DLTL such relation is allowed to be more general (as general as any function correlating the attribute values) and also to involve as many freeze variables as required. On the second

hand, DLTL also incorporates past temporal operators. Without them, some interesting properties relating current and past word positions could not be expressed.[3]

Freezing a state by means of a freeze variable x will allow us to talk about attributes of the event at that position, and then establish correlations between attributes of different events by means of relations, of the form "The resource associated to x is different than the resource associated to y" or "The price of such product doubles between events separated more than two days", for instance. The timestamp of an event can be considered as just another attribute. In the case of timestamp attributes we are going to assume they are coherent with the ordering of events in the trace, so that if event e_1 appears before than event e_2, the timestamp of e_1 will be no greater than the one of e_2.

Let us now formally introduce the DLTL logic.

Definition 1. *Let $\mathcal{D}$ be a set, called the* transition domain*; let $\mathcal{V} = \{x_1, x_2, \dots\}$ be a finite set of* freeze variables *and let $\Phi = \{\varphi_1(x^1_1, \dots, x^1_{m_1}), \varphi_2(x^2_1, \dots, x^2_{m_2}), \cdots \mid m_i \geq 0,\ x^j_i \in \mathcal{V},\ \forall i, j\}$ be a finite set of* relations*.*

The set of correct formulas, $\mathcal{F}_{(\mathcal{D},\mathcal{V},\Phi)}$, for the DLTL logic, is inductively defined as follows:

- *$f \in \Phi$ is a correct formula*
- *If f_1 and f_2 are correct formulas, so are $\neg f_1$, $f_1 \wedge f_2$, $X f_1$, $Y f_1$, $f_1\ U\ f_2$, $f_1\ S\ f_2$, $x.f_1$*

In the previous definition, a relation with one variable will be called a *proposition*.

Definition 2. *Let $f \in \mathcal{F}_{(\mathcal{D},\mathcal{V},\Phi)}$ be a correct DLTL formula. A* valuation *v is any mapping from the set of variables in f into $\mathcal{D}$*

We are going to interpret a DLTL formula of $\mathcal{F}_{(\mathcal{D},\mathcal{V},\Phi)}$ over finite words of elements of $\mathcal{D}$, of the form $\sigma = \sigma_1 \cdot \sigma_2 \cdot \ldots \cdot \sigma_n$ (as usual $\mid \sigma \mid$ denotes the length of the word). In order to make notations simpler, in the following, for a giving word σ, when talking about a valuation v we will assume that for any variable x, $v(x)$ is one of the sets in the word, identified by its position in σ and, therefore, $1 \leq v(x) \leq n$.

Definition 3. *Let $f \in \mathcal{F}_{(\mathcal{D},\mathcal{V},\Phi)}$ be a correct DLTL formula; let $\sigma = \sigma_1 \cdot \sigma_2 \cdot \ldots \cdot \sigma_n$ be a finite word over $\mathcal{D}$; let v be a valuation and let i be an index such that $1 \leq i \leq n$. By $\sigma, i \models_v f$ we denote that σ satisfies f for valuation v at position i. This relation is defined as follows:*

- *$\sigma, i \models_v p$ if $p(\sigma_i)$, for any proposition p*

[3] In the original logic, atomic formulas where associated to states. Since we are going to concentrate on log traces, the point of view we adopt associates general data to events.

- $\sigma, i \models_v \varphi(x_1, \ldots, x_m)$ *if* $\varphi(\sigma_{v(x_1)}, \ldots, \sigma_{v(x_m)})$.
- $\sigma, i \models_v \neg f$ *if* $\neg(\sigma, i \models_v f)$
- $\sigma, i \models_v f_1 \wedge f_2$, *for any pair* f_1 *and* f_2, *if* $\sigma, i \models_v f_1$ *and* $\sigma, i \models_v f_2$
- $\sigma, i \models_v Xf$, *for any formula* f, *if* $i < n$ *and* $\sigma, i+1 \models_v f$
- $\sigma, i \models_v Yf$, *for any formula* f, *if* $1 < i$ *and* $\sigma, i-1 \models_v f$
- $\sigma, i \models_v f_1\ U\ f_2$, *for any pair* f_1 *and* f_2, *if there exists an index* $i \leq k \leq n$ *such that* $\sigma, j \models_v f_2$ *and, for any* $i \leq j < k$, $\sigma, j \models_v f_1$
- $\sigma, i \models_v f_1\ S\ f_2$, *for any pair* f_1 *and* f_2, *if there exists an index* $j \leq i$ *such that* $\sigma, j \models_v f_2$ *and, for any* $j+1 \leq k \leq i$, $\sigma, k \models_v f_1$
- $\sigma, i \models_v x.f$, *for any formula* f *and variable* x *if* $\sigma, i \models_{v[x \leftarrow i]} f$, *where* $v[x \leftarrow i]$ *represents the valuation such that* $v[x \leftarrow i](x) = i$, *and has the same value than* v *for the rest of variables.*

The set of operators is extended with the classical abbreviations: $f_1 \vee f_2 \equiv \neg(\neg f_1 \wedge \neg f_2)$, $Ff \equiv true\ U\ f$, $Gf \equiv \neg(F\neg f)$, $f \Rightarrow g \equiv g \vee \neg f$, $f \Leftrightarrow g \equiv (f \Rightarrow g) \wedge (g \Rightarrow g)$, $O\ f \equiv true\ S\ f$, $Hf \equiv \neg(O\ \neg f)$ (here O operator stands for *Once*)

Example 1. As a first example, let us consider a trace of the execution of a process. Let us consider a set of agents, $Ag = \{a, b, c\}$, a set of actions, $Ac = \{req, ack, other\}$, and let $\mathcal{D} = Ag \times Ac \times \mathbb{R}$. Let us now consider the following word, corresponding to a trace of length 5:

$$\sigma = (a, req, 2)(b, req, 4)(a, ack, 6)(c, other, 8)(b, ack, 13)$$

For short, given $d \in \mathcal{D}$, $d.ag$, $d.act$ and $d.t$ will denote the first, second and third components, respectively.

The property that *for each agent, every req is followed by the corresponding ack of the same agent within a given time interval of 8 time units* can be expressed in DLTL this property can be established as follows:

$$f = G(x.(\varphi_1(x) \Rightarrow Fy.(\varphi_2(x, y) \wedge \varphi_3(x) \wedge \varphi_4(x, y))))$$

with $\varphi_1(x) = (x.act = req)$, $\varphi_2(x, y) = (x.ag = y.ag)$, $\varphi_3(x) = (x.act = ack)$ and $\varphi_4(x, y) = (y.t - x.t \leq 8)$, being, in this case, $\Phi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\}$.

In this example, variable x is used to "freeze" a position in the word, while variable y refers to a later position.

Example 2. In applied domains it is very usual to use patterns that correlate two or more different positions in a word. Let us consider, for instance, the previous formula. It is correlating those states with the action *req* with some later states verifying $\varphi_2 \wedge \varphi_3 \wedge \varphi_4$. From an abstract point of view, this can be seen as an instance of

$$\forall x \in REQ, \exists y \in ACK, x \leq y \wedge \phi(x, y)$$

where REQ and ACK represent, respectively, the sets of word positions with *req* and *ack* actions.

Table 1 lists a quite complete set of schemes corresponding to formulas correlating two positions. In the table, A and B correspond to sets, $\mathcal{C}_A$ and

$\mathcal{C}_B$ are characteristic functions for them and $\varphi(x, y)$ is the relation establishing the property data in those positions must satisfy. It is important to include past temporal logic operators, such as Y, O and H. Without them many interesting properties could not be expressed.

Abstract property	DLTL Formula
$\forall x \in A, \forall y \in B, x \leq y \Rightarrow \phi(x, y)$	$G(x \cdot (\mathcal{C}_A(x) \Rightarrow G(y.(\mathcal{C}_B(y) \Rightarrow \varphi(x, y)))))$
$\forall x \in A, \exists y \in B, x \leq y \wedge \varphi(x, y)$	$G(x.(\mathcal{C}_A(x) \Rightarrow F(y.(\mathcal{C}_B(y) \wedge \varphi(x, y)))))$
$\exists x \in A, \forall y \in B, x \leq y \Rightarrow \varphi(x, y)$	$F(x.(\mathcal{C}_A(x) \wedge y.G(\mathcal{C}_B(y) \Rightarrow y.\varphi(x, y)))))$
$\exists x \in A, \exists y \in B, x \leq y \wedge \varphi(x, y)$	$F(x.(\mathcal{C}_A(x) \wedge F(y.(\mathcal{C}_B(y) \wedge \varphi(x, y)))))$
$\forall x \in A, \forall y \in B, x \geq y \Rightarrow \varphi(x, y)$	$H(x \cdot (\mathcal{C}_A(x) \Rightarrow H(y.(\mathcal{C}_B(y) \Rightarrow \varphi(x, y)))))$
$\forall x \in A, \exists y \in B, x \geq y \wedge \varphi(x, y)$	$H(x.(\mathcal{C}_A(x) \Rightarrow O(y.(\mathcal{C}_B(y) \wedge \varphi(x, y)))))$
$\exists x \in A, \forall y \in B, x \geq y \Rightarrow \varphi(x, y)$	$O(x.(\mathcal{C}_A(x) \wedge y.H(\mathcal{C}_B(y) \Rightarrow y.\varphi(x, y)))))$
$\exists x \in A, \exists y \in B, x \geq y \wedge \varphi(x, y)$	$O(x.(\mathcal{C}_A(x) \wedge O(y.(\mathcal{C}_B(y) \wedge \varphi(x, y)))))$
$\forall x \in A, \forall y \in B, \varphi(x, y)$	$G(x.(\mathcal{C}_A(x) \Rightarrow H(y.(\mathcal{C}_B(y) \Rightarrow \varphi(x, y))))\wedge$ $G(y.(\mathcal{C}_B(y) \Rightarrow \varphi(x, y)))))$
$\forall x \in A, \exists y \in B, \varphi(x, y)$	$G(x.(\mathcal{C}_A(x) \Rightarrow O(y.(\mathcal{C}_B(y) \wedge \varphi(x, y))))\wedge$ $F(y.(\mathcal{C}_B(y) \wedge \varphi(x, y)))))$
$\exists x \in A, \forall y \in B, \varphi(x, y)$	$F(x.(\mathcal{C}_A(x) \wedge H(y.(\mathcal{C}_B(y) \Rightarrow \varphi(x, y))))\wedge$ $G(y.(\mathcal{C}_B(y) \Rightarrow \varphi(x, y)))))$
$\exists x \in A, \exists y \in B, \varphi(x, y)$	$F(x.(\mathcal{C}_A(x) \wedge O(y.(\mathcal{C}_B(y) \wedge \varphi(x, y))))\wedge$ $F(y.(\mathcal{C}_B(y) \wedge \varphi(x, y)))))$

Table 1. DLTL formula schemes correlating two word positions

3 The complexity of model checking a DLTL formula

[12] presents a deep and clear study of the complexity of the problem of verifying a TPTL formula against a finite word, which can be directly applied to the case of DLTL formulas. In this section we introduce a detailed description of the problem in DLTL with the aim of pointing out the reasons behind the cost of such evaluation. Besides of proving that it is in PSPACE [12], we prove that it is exponential in time with respect to the number of freeze variables, and linear with respect to the rest of the involved parameters (size of the formula and length of the word).

We first introduce a recursive procedure for model checking DLTL formulas, and then we evaluate the complexity of the method. Since the complexity depends on the evaluation of the relations in Φ we are going to assume that the cost of evaluating such relations is "reasonable".

Checking function $dltl\_sat(\sigma, i, v, f)$ takes as parameters a word, σ, a position in the word, i, a valuation v and a DLTL formula, f. Checking f on the word σ is carried out by means of the evaluation of $dltl\_sat(\sigma, 1, \emptyset, f)$. The algorithm is, basically, a recursive implementation of the inductive definition of DLTL formulas. In the case of parameter i being outside the range of σ, we consider the formula is false. Freeze variables are considered as word position variables. In the case of f being a relation $\varphi(x_1, \ldots, x_m)$, we assume in the evaluation of $dltl\_sat(\sigma, i, v, f)$ that valuation v binds a value for each variable $x_1, \ldots, x_m$.

This way evaluating the function is the same as evaluating $\varphi(v(\sigma_{x_1}), \ldots, v_(\sigma_{x_m}))$. Notice that such evaluation does not depend on parameter i (provided i is in the range of σ). Evaluating a formula $x.f$ for a position i is the same a making $x = i$ in v.

```
function dltl_sat(σ,i,v,f)
    if i ≤ 0 or i > |σ| then
        return false
    elseif f = φ(x_1, ..., x_m) then
        return φ(v(σ_{x_1}), ..., v(σ_{x_m}))
    elseif f = p then
        return p(σ_i)
    elseif f = X f_1 then
        return dltl_sat(σ,i + 1,v,f_1)
    elseif f = f_1 U f_2 then
        return dltl_sat(σ,i,v,f_2)
            or ( dltl_sat (σ,i,v,f_1) and dltl_sat(σ,i + 1,v,f))
    elseif f = Y f_1 then
        return dltl_sat(σ,i − 1,v,f_1)
    elseif f = f_1 S f_2 then
        return dltl_sat(σ,i,v,f_2)
            or ( dltl_sat (σ,i,v,f_1) and dltl_sat(σ,i − 1,v,f))
    else f = x.f_1 then
        local old_x = v[x]
        v[x] = i
        res = dltl_sat (σ,i,v,f_1)
        v[x] = old_x
        return res
end
```

Let us now concentrate on the complexity of the proposed algorithm. The cost clearly depends on the cost of evaluating relations $\varphi(v(\sigma_{x_1}), \ldots, v(\sigma_{x_m}))$. We are going to assume that they are PSPACE with respect to the size of f (as usual, the size is the number of operands and operators in the formula) and the length of σ, $|\sigma|$. With respect to the time, we are going to denote $K_{|f|,|\sigma|}$ a bound for all of them.

The following propositions establish the time and space complexity of $dltl\_sat(\sigma, i, v, f)$.

Proposition 1. *The model checking problem for $\sigma \models f$, where σ is a finite word and f is a DLTL formula, is PSPACE.*

Proof. Evaluating $dltl\_sat(\sigma, 1, \emptyset, f)$ will require, at most, $|f|$ recursive invocations. At each invocation $dltl\_sat(\sigma, i, v, g)$, where g is a subformula of f, valuation v can be passed as a reference to an $|Var|$-indexed array. On the other hand, f is coded by its syntax tree being each subformula g a node. As a consequences, the size of the parameters of each invocation are of constant size (the considered references plus the size of old_x when needed). Provided that we are assuming that evaluating $\varphi(v(\sigma_{x_1}), \ldots, v(\sigma_{x_m}))$ is PSPACE, we can conclude that evaluating $dltl\_sat(\sigma, 1, \emptyset, f)$ is also PSPACE.

In order to obtain a better time execution cost, we use dynamic programming techniques as the way of avoiding recomputing the same subformula more than once for the same parameters.

Proposition 2. *The model checking problem for $\sigma \models f$, where σ is a finite word and f is DLTL formula, can be solved in time*
$O((K_{|f|,|\sigma|} + |Var|) * |\sigma|^{|Var|} * |f| * |\sigma|)$

Proof. Provided that the same subformula is not going to be computed more than once, $|\sigma| * |\sigma|^{|Var|} * |f|$ is an upper bound for the number of invocations. In the case the subformula is $\varphi(v(\sigma_{x_1}), \ldots, v(\sigma_{x_m}))$, the cost is $K_{|f|,|\sigma|}$. When out of the word range, the cost is constant. We have also to consider the cost added by the dynamic programming technique. For that, we can use an array of size $|Var| + 2$, so that the cost of looking for a value is $O(|Var|)$. This way, we can conclude.

Let us now prove that the problem of checking a DLTL formula for a finite word is PSPACE HARD. Let us first prove that the problem of satisfying a QBF (Quantified Boolean Formula) can be translated into a checking problem.

Lemma 1. *Let $\phi(x_1, \ldots, x_{2n})$ be a boolean formula. Let Φ the the following quantified boolean formula:*

$$\Phi = \forall x_1, \exists x_2, \ldots \forall x_{2n-1}, \exists x_{2n}, \phi(x_1, \ldots, x_{2n})$$

Let us consider the word $\sigma = (1, true) \cdot (2, false) \ldots (2 * i + 1, true) \cdot (2i + 2, false) \ldots (2 * n - 1, true) \cdot (2n, false)$ and the following DLTL formula $f = y_0 \cdot L_\forall(1)$ defined as follows (for each transition x in the word, $x.t$ and $x.val$ denote the first and second components, respectively):

$$\begin{aligned} L_\forall(2n+1) &= \phi(y_1.val, \ldots, y_{2n}.val) \\ L_\forall(i) &= G(y_i \cdot (y_i.t - y_{i-1}.t \leq 1 \Rightarrow L_\exists(i+1))) \\ L_\exists(i) &= F(y_i \cdot (y_i.t - y_{i-1}.t \leq 1 \wedge L_\forall(i+1))) \end{aligned}$$

Then Φ is true iff σ fulfills the DLTL formula f.

Proof. We are going to prove, by induction, that

$$\Phi(2i+1)(y_1.t, \ldots, y_{2i}.t) = \forall x_{2i+1}, \exists x_{2i+2}, \ldots \forall x_{2n-1}, \exists x_{2n}, \phi(y_1.val, \ldots, y_{2i}.val, x_{2i+1}, \ldots x_{2n})$$

When $i = n$, $L_\forall(2n+1)$ does not depend on the position in the word, and the equality isi verified everywhere:

$$L_\forall(2n+1) = \Phi(2n+1) = \phi(y_1.val, \ldots, y_{2n}.val)$$

Assuming now the property is satisfied for $i+1$, let us prove that it is also true for i. $L_\forall(2i+1)$ can be expressed in terms of $L_\forall(2i+3)$ as follows:

$$L_\exists(2i+2) = F(y_{2i+2} \cdot (y_{2i+2}.t - y_{2i+1}.t \leq 1 \wedge L_\forall(2i+3)))$$
$$L_\forall(2i+1) = G(y_{2i+1} \cdot (y_{2i+1}.t - y_{2i}.t \leq 1 \Rightarrow L_\exists(2i+2)))$$

Applying induction hypothesis for $L_\exists(2i+2)$ we get:

$$L_\exists(2i+2) = F(y_{2i+2} \cdot (y_{2i+2}.t - y_{2i+1}.t \leq 1 \wedge \Phi(2i+3)))$$

for every position until $2i+2$. When evaluating F and freezing variable y_{2i+2}, only two non-trivial positions have to be considered: either the same position or the next one. Since two consecutive positions cover both boolean values, the formula can be simplified as follows:

$$\begin{aligned} L_\exists(2i+2) &= \Phi(2i+3)(y_1, \ldots, y_{2i+1}, false) \wedge \Phi(2i+1)(y_1, \ldots, y_{2i+1}, true) \\ &= \exists x_{2i+2}, \Phi(2i+3)(y_1, \ldots, y_{2i+1}, x_{2i+2}) \end{aligned}$$

Doing analogously for the formula $L_\forall(2i+1)$ and positions until $2i+1$, we reach the searched result:

$$\begin{aligned} L_\forall(2i+1) &= \forall x_{2i+1} \exists x_{2i+2}, \Phi(2i+3)(y_1, \ldots, x_{2i+1}, x_{2i+2}) \\ &= \Phi(2i+1) \end{aligned}$$

Proposition 3. *The model checking problem for $\sigma \models f$, where σ is a finite word and f is a DLTL formula, is PSPACE-Hard.*

Proof. Immediate from lemma 1

4 An application example

As an application case, let us consider the log described and analyzed in [13][4]. The log corresponds to the trajectories, obtained from the merging of data from the ERP of a Dutch hospital, followed by 1050 patients admitted to the emergency ward, presenting symptoms of a sepsis problem. The total number of events was 15214. Each event is composed of the *activity* (there are 16 different activities, categorized as either medical or logistical activities), as well as additional information (time-stamps, in seconds, of the beginning and end of the activities, data from laboratory tests and triage checklists, etc.). [13] applies different automatic process discovery techniques and obtain different models

The objective of this section is to show how some of the system requirements, including time constraints, can be expressed in terms of DLTL formulas and checked for conformance with the log. In the following, for an event x of a trace, $x.a$ and $x.t$ correspond to the activity and time-stamp in seconds, respectively.

[4] https://doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54

Requirement: "Between *ER Sepsis Triage* and *IV Antibiotics* actions should be less than 1 hour"[5]. Since a-priori we do not know whether there exists any causal relation between the considered activities, we are going to check the requirement as follows. Let

$$
\begin{aligned}
r1\_0 &= F(\text{ER Sepsis Triage}) \ \wedge \ F(\text{IV Antibiotics}) \\
r1\_1 &= F(x.(x.a = \text{ER Sepsis Triage} \ \wedge \\
&\qquad XF(y.(y.a = \text{IV Antibiotics} \ \wedge \ y.t - x.t \leq 3600)))) \\
r1\_2 &= F(x.(x.a = \text{IV Antibiotics} \ \wedge \\
&\qquad XF(y.(y.a = \text{ER Sepsis Triage} \ \wedge \ y.t - x.t \leq 3600))))
\end{aligned}
$$

be the formulas that check how many traces contain both activities, how many execute the second activity no later than one hour after the first and how many execute the first activity no later than one hour after the second one, respectively. Checking $r1\_0$, $r1\_1$ and $r1\_2$ gives, respectively, 823, 342 and 0 positive answers. This means that the requirement is fulfilled in 41.5% cases and, therefore, violated in 58.5% of the cases. Notice also that there is a causal relation between both events, since the *ER Sepsis Triage* always precedes *IV Antibiotics*. This result coincides with the one presented in [13].

Requirement: "Between *ER Sepsis Triage* and *LacticAcid* should be less than 3 hours". Let us now consider the following formulas:

$$
\begin{aligned}
r2\_0 &= F(\text{ER Sepsis Triage}) \\
r2\_1 &= F(\text{ER Sepsis Triage}) \ \wedge \ F(\text{LacticAcid}) \\
r2\_2 &= F(x.(x.a = \text{ER Sepsis Triage} \ \wedge \\
&\qquad F(y.(y.a = \text{LacticAcid} \ \wedge \ y.t - x.t \leq 10800)))) \\
r2\_3 &= F(x.(x.a = \text{ER Sepsis Triage} \ \wedge \\
&\qquad O(y.(y.a = \text{LacticAcid} \ \wedge \ x.t - y.t \leq 10800))))
\end{aligned}
$$

$r2\_0$ gives that there are 1048 cases in which *ER Sepsis Triage* happens, $r2\_1$ is satisfied by 859 cases, $r2\_2$ states that in 711 cases *LacticAcid* appears in the considered period after *ER Sepsis Triage*, while $r2\_3$ proves that in 133 cases *LacticAcid* is in the considered period, but happening before; finally, only 2 cases verify $r2\_2 \wedge r2\_3$. If one just considers those cases in $r2\_1$, the property is held in 98.02%, and violated in only 1.98%. This result is different than the 0.7% reported in [13]. The discrepancy could be explained in the way requirements have been checked. In our case, we directly work with the log, considering every trace. However, [13] checks the requirement against a model extracted from the log using Multi-perspective Process Explorer, which fits 98.3% of traces. On the other, if one considers the rule time

[5] As in [13], we are using "$\leq$" to check the properties, besides "should be less than 1 hour" suggests "$<$" should be used

must be verified for every case in which *ER Sepsis Triage* occurs ($r2\_0$), the property is true in only 80.34% of the cases.

Requirement: Another proposed question is related to the patients returning to the service. Formula $r3\_0 = F(\text{Return ER})$ gives 28% of positive answers (27.8% in [13]). They are also interested in knowing how many of them return within 28 days. This can be checked with the formula

$$r3\_1 = x.(F(y.(y.a = \text{Return ER} \ \wedge \ y.t - x.t \leq 28 * 24 * 3600)))$$

obtaining 8.95% (12.6% in [13]). An interesting result is that only 28.7% of those patients that return within 28 days correspond to patients that verify the constraints of one and three hours previously checked.

5 About the model checking process

In this section, we briefly describe a way of implementing a DLTL model checker. In order to describe the way DLTL formulas can be checked, let us consider again Example 1:

$$f = G(x.((x.act = req) \Rightarrow Fy.((x.ag = y.ag) \wedge (y.act = ack) \wedge (y.t - x.t \leq 8))))$$

Walking over the syntax tree of the formula allows to build the tableau used for checking it, as in Table 2. In this example, first row is annotated, for each column, with the symbolic representation of the formula $\phi(x, y)$ (in fact, this state is reached after analyzing the leaves of the $\wedge$ subtrees). Next, the $y.\phi(x, y)$ is evaluated: for each column c, y must take the value σ_c, giving the corresponding $\phi(x, c)$ symbolic column. For instance $\phi(x, 5) = (x.ag = b) \wedge (y.act = ack) \wedge (13 - x.t \leq 8)$. Next row corresponds to $F(y.\phi(x, y))$, and so on until the complete tree is evaluated. As a result, a vector of true/false values is obtained. The value in position 1 is the result of checking the formula.

pos	1	2	3	4	5
σ	(a,req,2)	(b,req,4)	(a,ack,6)	(c,other,8)	(b,ack,13)
$\phi(x, y)$	$\phi(x, y)$	$\phi(x, y)$	$\phi(x, y)$	$\phi(x, y)$	$\phi(x, y)$
$f_1(x) = y \cdot \phi(x, y)$	$\phi(x, 1)$	$\phi(x, 2)$	$\phi(x, 3)$	$\phi(x, 4)$	$\phi(x, 5)$
$f_2(x) = F(f_1(x))$	$\exists i \geq 1, \phi(x, i)$	$\exists i \geq 2, \phi(x, i)$	$\exists i \geq 3, \phi(x, i)$	$\exists i \geq 4, \phi(x, i)$	$\phi(x, 5)$
$f_3(x) = (x.act = req)$	$x.act = req$	$x.act = req$	$x.act = req$	$x.act = req$	$x.act = req$
$f_4 = x \cdot (f_3(x) \Rightarrow f_2(x))$	$f_3(1) \Rightarrow f_2(1)$ true	$f_3(2) \Rightarrow f_2(2)$ false	$f_3(3) \Rightarrow f_2(3)$ true	$f_3(4) \Rightarrow f_2(4)$ true	$f_3(5) \Rightarrow f_2(5)$ true
$G(f_4)$	false	false	true	true	true

Table 2. Checking tableau for the formula in Example 1

As stated, the required time can be exponential with respect to the number of freeze variables. In order to get insight of the real time required we have carried

out some experiments measuring the user time required for checking formulas with an increasing number of freeze variables. For that, we have considered the following parametrized formula

$$\phi(n) = Fx_1.(Gx_2.(Fx_3.(Gx_4.(\dots Fx_{2n-1}.(Gx_{2n}.(\bigwedge_{i=2}^{2n}(x.t_i - x.t_{i-1} \leq 100)\dots)$$

Figure 1 shows the chart corresponding to checking the formula for different values of parameter n against the sepsis log used in Section 4. The curve is as expected. It fits the exponential $y = 0.9270899856 \cdot e^{0.1030458522 \cdot x}$ ($R^2 = 0.9956898464$, $rss = 297.8737955$). Notice that the method is able to efficiently deal with "many" freeze variables. If we constraint ourselves to a set of usual patterns involving a small set of freeze variables (as it is the case of the DECLARE patterns, for instance) the model checking method is quite efficient (for instance, checking $r2\_3$, which only involves two freeze variables, against the sepsis log, needed 0.088 seconds).

The experiments have been carried out with a prototype of the model checker implemented in lua 5.3, and executed in a Intel(R) Core(TM) i7-4790K CPU @ 4.00GHz computer with a Ubuntu 16.04 operating system.

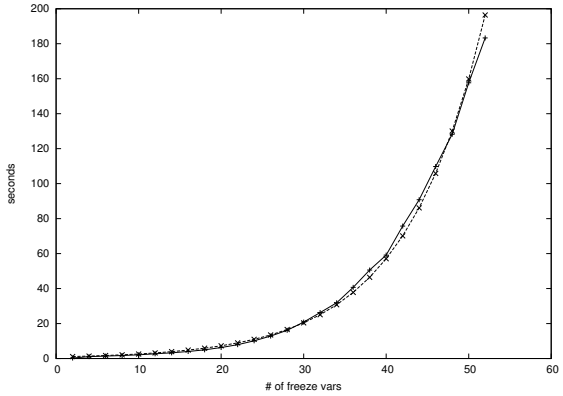

Fig. 1. Time versus number of freeze variables for formula $\phi(n)$, compared to $y = 0.9270899856 \cdot e^{0.1030458522 \cdot x}$ (continuous line corresponds to experimental results)

6 Related work

Temporal logic with data has been used in different domains. [14] proposes Quantified-Free First-Order LTL (QFLTL($\mathbb{R}$)) where transitions, besides atomic propositions, can also contain real data attributes. QFLTL formulas are allowed to include classical operators between real expressions. Global variables can be

used to correlate data of different transitions. Checking a formula is translated into finding intervals of the involved real variables verifying the constraints in the formula. The logic is constrained to some specific event structure and operations, of interest for the concrete domain it is proposed for. Temporal databases, together with temporal logic, have been used as a way to correlate time and data, allowing to analyze data correlations between the values of the database states at different time instants [15, 16].

The addition of freeze variables (also named as *counters* in the domain) to classical LTL, as proposed in TPTL [11] allows correlating values of different points in a word. For the case of more general data in transitions (the term *dataword* is also used in the literature to refer to general words whose elements are data of a given domain), *freezeLTL* [17] is able to deal with correlations between attributes checking the equality of the considered values for a subset of TPTL. For the full TPTL, [12] studies the complexity of model-checking a TPTL formula against a finite word.

In the domain of process mining many works have dealt with conformance checking using LTL as the way of specifying behavioral properties. Since in most cases authors are interested in imposing or finding some process structures, they usually concentrate on a restricted set of patterns which reflect usual and interesting event dependencies. This is the case of the set of patterns in the *Declare* [18] workflow management system. The Declare approach focused on the control perspective, defining a specific set of patterns. Instances of such patterns define specific constrains the system must verify. [7] proposed MP-Declare, an extension of Declare including the data perspective of events. The paper also proposes a checking method for the considered logic, based on SQL.

[19] uses Timed-Declare as the formalism to add time to Declare. They constraint Metric Temporal Logic (MLT) [20] to the set of Declare patterns and adapt it to finite traces. Besides detecting that a constraint has already been violated, the proposed method can be used for the monitoring of the system evolution allowing an early detection that a certain constraint would be violated in the future, allowing for an a-priori guidance to avoid undesired situations.

In [8] the authors propose an approach which allows a multi-perspective point of view in which data and timestamps (those must be natural numbers) of events are considered as two parallel structures (according to [15], they adopt a snapshot perspective). Metric First Order Temporal Logic (MFOTL) [10] (adapted for finite traces) is used as the formalism for the specification of properties. The paper reformulates MP-Declare patterns as MFOTL formulas, and presents a general framework for conformance checking. The framework is based on a general skeleton algorithm, which requires a different instance for each MP-Declare pattern.

The two previous methods, as stated, concentrate on a subset of MP-Declare, and specific methods must be developed for specific patterns, either as a specific function in the second case or as a specific SQL query in the first. On the other hand, given the specific application domain both methods are devoted, the proposed methods do not provide with procedures to check complex formulas

specifying complex relations between formulas, rather than between pairs of events (the *activation* event, which imposes requirement conditions for the *target* event by means of a relation that must be satisfied by the corresponding associated data).

7 Conclusions

The paper has introduced a linear temporal logic able to deal with correlations among different values associated to different points in a finite word. Also, a model checking procedure has been introduced, and its complexity established in terms of the formula and word sizes. The interest of working with finite words comes from the fact the logic is going to be applied to the analysis of system logs. For testing purposes, a model checker prototype has been developed in lua (not described in the paper) which has been used for the application example. The introduced method is general in the sense that there is neither constraint with respect to the set of temporal logic formulas that can be checked nor with respect to the attributes that can be handled by the logic.

One direction for future work is to explore whether the proposed approach can be effectively used for complex logs with complex formulas. In the experiments we have carried out the response time was really short, but deeper analysis is necessary to deduce its applicability to big logs. Then problem of dealing with a big number of traces can be alleviated by parallelizing the checking procedure: just use different parallel processors for dealing with different subsets of traces. The expensive dimensions are the length of the trace and the number of freeze variables.

Acknowledgments

This work was done when J. Ezpeleta was a visiting researcher at the Orléans University, and partially supported by the TIN2014-56633-C3-2-R research project, granted by the Spanish Ministerio de Economía y Competitividad.

References

1. Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. In: Proceedings of the 6th International Conference on Extending Database Technology: Advances in Database Technology. EDBT '98, London, UK, UK, Springer-Verlag (1998) 469–483
2. van der Aalst, W.M.P., de Beer, H.T., van Dongen, B.: Process Mining and Verification of Properties: An Approach Based on Temporal Logic. In: Procedings of the On the Move to Meaningful Internet Systems 2005 (OTM 2005), Agia Napa, Cyprus, October 31 - November 4, 2005. (2005) 130–147
3. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1) (March 2008) 64–95

4. de Leoni, M., van der Aalst, W.: Data-aware process mining: Discovering decisions in processes using alignments. In: Proceedings of the 28th ACM Symposium on Applied Computing (SAC 2013) 18-22 March, Coimbra, Portugal, pp. 113-129. (2013)
5. Räim, M., Ciccio, C., Maggi, F.M., Mecella, M., Mendling, J.: Log-Based Understanding of Business Processes through Temporal Logic Query Checking. In: On the Move to Meaningful Internet Systems: OTM 2014 Conferences: Confederated International Conferences: CoopIS, and ODBASE 2014, Amantea, Italy, October 27-31, 2014, Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg (2014) 75–92
6. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Balanced multi-perspective checking of process conformance. Computing **98**(4) (2016) 407–437
7. Schönig, S., Di Ciccio, C., Maggi, F.M., Mendling, J.: Discovery of multi-perspective declarative processmodels. In: Service-Oriented Computing - 14th International Conference, ICSOC 2016, Banff, AB, Canada, October 10-13, 2016. Volume 9936 of Lecture Notes in Computer Science., Springer (2016) 87–103
8. Burattin, A., Maggi, F.M., Sperduti, A.: Conformance checking based on multi-perspective declarative process models. Expert Systems with Applications **65** (2016) 194–211
9. Basin, D., Klaedtke, F., Müller, S., Pfitzmann, B.: Runtime monitoring of metric first-order temporal properties. In Hariharan, R., Mukund, M., Vinay, V., eds.: Proceedings of the 28th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2008), Dagstuhl, Germany, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2008)
10. Basin, D., Klaedtke, F., Müller, S.: Monitoring security policies with metric first-order temporal logic. In: Proceedings of the 15th ACM Symposium on Access Control Models and Technologies. SACMAT '10, New York, NY, USA, ACM (2010) 23–34
11. Alur, R., Henzinger, T.A.: A really temporal logic. J. ACM **41**(1) (January 1994) 181–203
12. Feng, S., Lohrey, M., Quaas, K.: Path checking for MTL and TPTL over data words. In Potapov, I., ed.: Developments in Language Theory - 19th International Conference, DLT 2015, Liverpool, UK, July 27-30, 2015, Proceedings. Volume 9168 of Lecture Notes in Computer Science., Springer (2015) 326–339
13. Mannhardt, F., Blinde, D.: Analyzing the trajectories of patients with sepsis using process mining. In: RADAR+EMISA 2017, CEUR-WS.org (2017) 72–80
14. Fages, F., Rizk, A.: On temporal logic constraint solving for analyzing numerical data time series. Theoretical Computer Science **408**(1) (2008) 55–65
15. Chomicki, J., Toman, D. In: Temporal Logic in Information Systems. Springer US, Boston, MA (1998) 31–70
16. Chomicki, J., Toman, D. In: Temporal Logic in Database Query Languages. Springer US, Boston, MA (2009) 2987–2991
17. Demri, S., Lazić, R.: Ltl with the freeze quantifier and register automata. ACM Trans. Comput. Logic **10**(3) (April 2009) 16:1–16:30
18. Pesic, M., Schonenberg, H., van der Aalst, W.: DECLARE: Full Support for Loosely-Structured Processes. In: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference. IEEE Computer Society, Washington, DC, USA (2007) 287–

19. Maggi, F.M., Westergaard, M.: Using timed automata for a priori warnings and planning for timed declarative process models. International Journal of Cooperative Information Systems **23**(01) (2014) 1440003
20. Koymans, R.: Specifying real-time properties with metric temporal logic. Real-Time Systems **2**(4) (Nov 1990) 255–299