SMT-based Checking of SOLOIST over Sparse Traces

Marcello M. Bersani[1], Domenico Bianculli[2], Carlo Ghezzi[1], Srđan Krstić[1], and Pierluigi San Pietro[1]

[1] DEEP-SE group - DEIB - Politecnico di Milano, Italy
{bersani,ghezzi,krstic,sanpietr}@elet.polimi.it
[2] SnT Centre - University of Luxembourg, Luxembourg
domenico.bianculli@uni.lu

Abstract. SMT solvers have been recently applied to bounded model checking and satisfiability checking of metric temporal logic. In this paper we consider SOLOIST, an extension of metric temporal logic with aggregate temporal modalities; it has been defined based on a field study on the use of specification patterns in the context of the provisioning of service-based applications. We apply bounded satisfiability checking to perform trace checking of service execution traces against requirements expressed in SOLOIST. In particular, we focus on sparse traces, i.e., traces in which the number of time instants when events occur is very low with respect to the length of the trace.

The main contribution of this paper is an encoding of SOLOIST formulae into formulae of the theory of quantifier-free integer difference logic with uninterpreted function and predicate symbols. This encoding paves the way for efficient checking of SOLOIST formulae over sparse traces using an SMT-based verification toolkit. We report on the evaluation of the proposed encoding, commenting on its scalability and its effectiveness.

1 Introduction

Bounded satisfiability checking [23] (BSC) is a verification technique that complements bounded model checking [9] (BMC): instead of a customary operational model (e.g., a state-transition system) used in BMC, BSC supports the analysis of a *descriptive model*, denoted by a set of temporal logic formulae. With BSC, verification tasks become suitable instances of the satisfiability problem for quite large formulae (written in a certain logic), which comprehend the model of the system to analyze as well as the requirement(s) to verify. BSC has been successfully applied in the context of metric temporal logics and implemented in ZOT [23], a verification toolset based on SAT- and SMT-solvers, developed within our group.

In this paper we apply BSC to trace checking for the language SOLOIST (*SpecificatiOn Language fOr servIce compoSitions inTeractions*) [8], a metric temporal logic with new, additional temporal modalities that support aggregate operations on events occurring in a given time window. SOLOIST has been defined based on the results of a field study [7] on the type of property specification patterns used to express requirements in the context of service-based applications. The study—performed by some of the authors in collaboration with an industrial partner—analyzed more than 900 requirements specifications, extracted both from research papers and industrial data, and led to the identification of a new class of specification patterns, in addition to the ones already known in literature [13, 17]. The new class of patterns is specific to the domain of service provisioning and contains seven patterns, among which there are: *average response time* (hereafter referred to as S1), *count of the number of events* (S2), *average/maximum number of events* (S3/S4).

SOLOIST can be used to specify both functional and quality-of-service requirements of the interactions of a composite service with its partner services. As for the verification of properties expressed in SOLOIST, in [8] we first presented a translation of SOLOIST into LTL that, under certain assumptions, guaranteed its decidability based on well-known results in temporal logic. Nevertheless, this translation was only a proof of concept and was not meant to be used for implementing efficient verification procedures. In subsequent work [18], some of the authors described an approach for performing *trace checking*[3] of service execution traces against requirements expressed in SOLOIST. The approach in [18] defined the *trace checking* problem in terms of the BSC problem for metric temporal logic, where the descriptive model of system executions is represented by traces, while properties are expressed in SOLOIST; in particular, it translates SOLOIST into CLTLB($\mathscr{D}$) [4], an extension of PLTLB (Propositional Linear Temporal Logic with both future and past modalities) augmented with atomic formulae over a constraint system $\mathscr{D}$; the resulting CLTLB($\mathscr{D}$) formula is then checked by ZOT. The main limitation of the approach presented in [18] is that it does not scale well when the trace to check is *sparse*, i.e., when the number of time instants in which events occur is very low with respect to the length of the trace. Notice that the case of sparse traces is *not rare* in the logs of service-based applications. For example, the log used for the *Business Process Intelligence Challenge 2012 (BPIC 2012)* [12] was taken from a Dutch Financial Institute; it contains 13087 traces, whose average number of time instants in which events occur is 20.0347: this represents (on average) the 0.003% of the total number of time instants.

The main contribution of this paper is a new encoding of SOLOIST, targeting formulae of quantifier-free difference logic with uninterpreted function and predicate symbols (QF-EUFIDL), for which there exist efficient decision procedures to be used with SMT solvers. As confirmed by the experimental evaluation we detail in the paper, this new encoding targeting QF-EUFIDL proves to be scalable and effective for checking SOLOIST formulae over sparse traces.

Related Work. There are only few approaches that deal with the verification of properties involving aggregate modalities. Basin et al. [2] define an extension of metric first-order temporal logic that supports aggregation. The language can express aggregate properties over the values of the parameters of relations (corresponding to system events), while SOLOIST expresses aggregate properties on the occurrences of the events. Finkbeiner et al. [15] describe an approach to collect statistics over run-time executions. They use LTL extended with the capability to compute aggregate properties of the trace. However, this specification language provides only limited support for

[3] Trace checking is also called *trace validation* [20] or *history checking* [14].

timing information; unlike SOLOIST, it cannot express properties on a certain subset of an execution trace. This work is also related to approaches for SAT/SMT-based trace checking and bounded model checking, which is usually done over properties expressed in conventional temporal logics. For example, the SAT-based approach for bounded model checking proposed in [24] verifies Metric Temporal Logic (MTL) properties of discrete timed automata. SMT-based techniques like those proposed in [5, 6, 16] deal with verification of MTL over real-valued words.

The rest of the paper is structured as follows. Section 2 provides a brief introduction to SOLOIST and QF-EUFIDL. The main contribution of the paper is presented in Sect. 3, where we present the encoding of SOLOIST into QF-EUFIDL over a finite temporal structure and assess its complexity. Section 4 reports on the evaluation of an implementation of the proposed encoding, performed to assess its scalability and effectiveness. Section 5 gives some concluding remarks.

2 Background

2.1 SOLOIST at a glance

In this section we provide a brief overview of SOLOIST; for the rationale behind the language and a detailed explanation of its semantics see [8].

The syntax of SOLOIST is defined by the following grammar:
$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \mathsf{U}_I \phi \mid \phi \mathsf{S}_I \phi \mid \mathfrak{C}^{K}_{\bowtie n}(\phi) \mid \mathfrak{U}^{K,h}_{\bowtie n}(\phi) \mid \mathfrak{M}^{K,h}_{\bowtie n}(\phi) \mid \mathfrak{D}^{K}_{\bowtie n}(\phi, \phi)$
where $p \in \Pi$, with Π being a finite set of atoms; I is a nonempty interval over $\mathbb{N}$; n, K, h range over $\mathbb{N}$; $\bowtie \in \{<, \leq, \geq, >, =\}$. We restrict the arguments ϕ of modalities $\mathfrak{C}, \mathfrak{U}, \mathfrak{M}, \mathfrak{D}$ to atoms in Π.

The U_I and S_I modalities are, respectively, the metric "*Until*" and "*Since*" operators. Additional temporal modalities can be derived using the usual conventions; for example "*Always*" is defined as $\mathsf{G}_I\phi \equiv \neg(\top \mathsf{U}_I \neg\phi)$ and "*Eventually in the Past*" as $\mathsf{P}_I\phi \equiv \top \mathsf{S}_I \phi$, where $\top$ means "true". The remaining modalities are called *aggregate* modalities and are used to express the specification patterns S1–S4 mentioned above. The $\mathfrak{C}^{K}_{\bowtie n}(\phi)$ modality states a bound (represented by $\bowtie n$) on the number of occurrences of an event ϕ in the previous K time instants: it expresses pattern S2. The $\mathfrak{U}^{K,h}_{\bowtie n}(\phi)$ (respectively, $\mathfrak{M}^{K,h}_{\bowtie n}(\phi)$) modality expresses a bound on the average (respectively, maximum) number of occurrences of an event ϕ, aggregated over the set of right-aligned adjacent non-overlapping subintervals within a time window K; it corresponds to pattern S3 (respectively, S4), as in "the average/maximum number of events per hour in the last ten hours". A subtle difference in the semantics of the $\mathfrak{U}$ and $\mathfrak{M}$ modalities is that $\mathfrak{M}$ considers events in the (possibly empty) tail interval, i.e., the leftmost observation subinterval whose length is less than h, while the $\mathfrak{U}$ modality ignores them. The $\mathfrak{D}^{K}_{\bowtie n}(\phi, \psi)$ modality expresses a bound on the average time elapsed between a pair of specific adjacent events ϕ and ψ occurring in the previous K time instants; it can be used to express pattern S1.

The formal semantics of SOLOIST is defined on timed ω-words [1] over $2^{\Pi} \times \mathbb{N}$. A timed sequence $\tau = \tau_0\tau_1 \ldots$ is an infinite sequence of values $\tau_i \in \mathbb{N}$ with $\tau_i > 0$ satisfying $\tau_i < \tau_{i+1}$, for all $i \geq 0$, i.e., the sequence increases strictly monotonically.

$$\begin{array}{ll}
(w,i) \models p & \text{iff } p \in \sigma_i \\
(w,i) \models \neg\phi & \text{iff } (w,i) \not\models \phi \\
(w,i) \models \phi \wedge \psi & \text{iff } (w,i) \models \phi \wedge (w,i) \models \psi \\
(w,i) \models \phi \mathsf{S}_I \psi & \text{iff for some } j < i, \tau_i - \tau_j \in I, (w,j) \models \psi \text{ and for all } k, j < k < i, (w,k) \models \phi \\
(w,i) \models \phi \mathsf{U}_I \psi & \text{iff for some } j > i, \tau_j - \tau_i \in I, (w,j) \models \psi \text{ and for all } k, i < k < j, (w,k) \models \phi \\
(w,i) \models \mathfrak{C}^{K}_{\bowtie n}(\phi) & \text{iff } c(\tau_i - K, \tau_i, \phi) \bowtie n \text{ and } \tau_i \geq K \\
(w,i) \models \mathfrak{U}^{K,h}_{\bowtie n}(\phi) & \text{iff } \dfrac{c(\tau_i - \lfloor \frac{K}{h} \rfloor h, \tau_i, \phi)}{\lfloor \frac{K}{h} \rfloor} \bowtie n \text{ and } \tau_i \geq K \\
(w,i) \models \mathfrak{M}^{K,h}_{\bowtie n}(\phi) & \text{iff } \max \left\{ \bigcup_{m=0}^{\lfloor \frac{K}{h} \rfloor} \{c(lb(m), rb(m), \phi)\} \right\} \bowtie n \text{ and } \tau_i \geq K \\
(w,i) \models \mathfrak{D}^{K}_{\bowtie n}(\phi, \psi) & \text{iff } \dfrac{\sum_{(s,t) \in d(\phi,\psi,\tau_i,K)} (\tau_t - \tau_s)}{|d(\phi, \psi, \tau_i, K)|} \bowtie n \text{ and } \tau_i \geq K
\end{array}$$

where $c(\tau_a, \tau_b, \phi) = |\{s \mid \tau_a < \tau_s \leq \tau_b \text{ and } (w,s) \models \phi\}|$, $lb(m) = \max\{\tau_i - K, \tau_i - (m+1)h\}$, $rb(m) = \tau_i - mh$, and $d(\phi, \psi, \tau_i, K) = \{(s,t) \mid \tau_i - K < \tau_s \leq \tau_i \text{ and } (w,s) \models \phi, t = \min\{u \mid \tau_s < \tau_u \leq \tau_i, (w,u) \models \psi\}\}$

Fig. 1: Formal semantics of SOLOIST

A timed ω-word over alphabet 2^{Π} is a pair (σ, τ) where $\sigma = \sigma_0\sigma_1\ldots$ is an infinite word over 2^{Π} and τ is a timed sequence. A timed language over 2^{Π} is a set of timed words over the same alphabet. Notice that there is a distinction between the integer position i in the timed ω-word and the corresponding timestamp τ_i. Figure 1 defines the satisfiability relation $(w,i) \models \phi$ for every timed ω-word w, every position $i \geq 0$ and for every SOLOIST formula ϕ. For the sake of simplicity, hereafter we express the $\mathfrak{U}$ modality in terms of the $\mathfrak{C}$ one, based on this definition: $\mathfrak{U}^{K,h}_{\bowtie n}(\phi) \equiv \mathfrak{C}^{\lfloor \frac{K}{h} \rfloor \cdot h}_{\bowtie n \cdot \lfloor \frac{K}{h} \rfloor}(\phi)$, which can be derived from the semantics in Fig. 1.

We remark that the version of SOLOIST presented here is a restriction of the original one in [8]: to simplify the presentation in the next sections, we dropped first-order quantification on finite domains and limited the argument of the $\mathfrak{D}$ modality to only one pair of events; as detailed in [8], these assumptions do not affect the expressiveness of the language.

2.2 QF-EUFIDL

The target language of our encoding is a quantifier free integer difference logic formula with uninterpreted function and predicate symbols (QF-EUFIDL). Since trace checking only deals with finite traces, we require the outcome of the encoding to be a QF-EUFIDL formula that is satisfiable if and only if there exists a *finite* timed word that satisfies the translated SOLOIST formula. Such a logic combines decision procedures from two theories, namely theory of equality and uninterpreted functions (EUF) and theory of integer difference logic (IDL). This combination is shown to be decidable, and the satisfiability problem is NP-complete, according to Nelson-Oppen Theorem [21]. Well-formed EUF formulae conform to the following grammar: $\phi ::= p \mid t = t \mid \neg\phi \mid \phi \vee \phi$, with $t ::= v \mid f(t, \ldots, t)$, where p is an atomic proposition, v is a variable and f is a function. An example is $f(x) = y \wedge x = g(y) \wedge (\neg p \vee q)$, where x and y are variables while p and q are atomic propositions. The decision procedure for this logic combines SAT solving (for the propositional formulae) with an algorithm that checks equalities by building a tree representation of their equivalence classes. Integer difference Logic is a restriction of the theory of linear arithmetic and can be represented with the structure $(\mathbb{Z}, =, (<_d)_{d \in \mathbb{Z}})$, for which decidability has been proven in [11]; each $<_d$ is a binary relation defined as $x <_d y \leftrightarrow x < y + d$, and notations like $x < y$, $x \leq y$, $x \geq y$, $x > y$ and $x = y + d$ are abbreviations for $x <_0 y$, $x <_0 y \vee x = y$, $\neg(x <_0 y)$, $\neg(x <_0 y \vee x = y)$ and $y <_{1-d} x \wedge x <_{d+1} y$, respectively.

Although LTL with arithmetical constraints is proved [11] undecidable over infinite words, and QF-EUFIDL involves variables over discrete infinite domains, our particular use is bounded, because we deal with finite words; hence, the decidability is retained.

3 Encoding SOLOIST into QF-EUFIDL

SOLOIST can be seen as MTL over discrete time, enriched with aggregate modalities. MTL satisfiability checking over discrete time [23] can be efficiently performed by reducing semantics of U_I and S_I to suitable propositional formulae which take advantage from the information about the metric over time defined by I. In [23], however, authors consider ω-words as models for MTL formulae without timestamps. Therefore, the temporal structure required to translate the semantics of a formula such as $\top \mathsf{U}_{[10,10]} \phi$ is at least as long as ten discrete positions, because no timing information is available from the model. In this paper, we devise a new way to represent information about timing constraints defined in metric temporal modalities (including the aggregate ones); this is an improvement on the method proposed in [23]. The encoding presented afterwards is an extension of the one defined in [3], which allows one to capture timed ω-words. As a consequence, models do not require as many discrete positions as needed to build the discrete temporal structure in [23], because the measure of time distances is realized through arithmetical variables that store how much time elapses among consecutive discrete positions. Intuitively, by adding an arithmetical variable $\tau \in \mathbb{N}$ measuring the elapsed time, formula $\top \mathsf{U}_{[10,10]} \phi$ holds at position i if, for instance, at position $i+1$, ϕ holds and the time τ elapsed between position i and position $i+1$ is equal to 10. To realize this counting mechanism with variables and arithmetical operators, we require a language that incorporates arithmetics, hence our choice of QF-EUFIDL as the target language of our encoding.

We use the following QF-EUFIDL structure $(\mathbb{Z}, F, P, V, =, <)$ where F contains functions of the form $f : \mathbb{Z}_0^+ \to \mathbb{Z}$. Each function represents arithmetical variable used in the encoding. Set P contains boolean functions of the form $p : \mathbb{Z}_0^+ \to \{\top, \bot\}$; each of them represents a predicate whose value is defined over a nonnegative integer domain. Set V is a subset of F containing nullary functions returning a value from $\mathbb{Z}$. Using this structure we can define a finite representation of models of SOLOIST formulae. Since our structure is ordered, let $0, 1, 2, \ldots, H$ be a finite linear order, with H corresponding to the length of the finite prefix of the timed ω-word satisfying a SOLOIST formula. The linear order represents a temporal structure and since it is a subset of the domain of both the predicates from P and the functions from F, we can interpret them as having “time dependent” values. On the other hand, we can interpret elements of V as being time invariant, i.e., have constant value over the linear order.

In the encoding, we use the notation $[\![X]\!]$ to denote any additional predicate introduced in P to represent an entity X. We denote with $|X|$ an additional arithmetical variable in F representing an arithmetical entity X. We use $[\![X]\!]_i$ and $|X|_i$ as a shorthand for $[\![X]\!](i)$ and $|X|(i)$, respectively. The truth of $[\![X]\!]_i$ is interpreted as entity X holding at time instant i in an execution trace (or, equivalently, a timed word).

We assume SOLOIST formulae to be in *positive normal form* (PNF). The PNF of a formula is an equivalent formula where negation may only occur on atoms, i.e., atomic propositions (see [22]). PNF can be obtained by propagating the negation towards the atoms, by means of converting a negated operator into its dual version and negating its operand(s). To do so, we introduce the connective $\vee$, dual of $\wedge$, as well as the dual versions of all temporal modalities. The dual of U_I is "*Release*" R_I: $\phi \mathsf{R}_I \psi \equiv \neg(\neg\phi \mathsf{U}_I \neg\psi)$; the dual of S_I is "*Trigger*" T_I: $\phi \mathsf{T}_I \psi \equiv \neg(\neg\phi \mathsf{S}_I \neg\psi)$[4]. A negation in front of one of the $\mathfrak{C}^K_{\bowtie n}, \mathfrak{U}^{K,h}_{\bowtie n}, \mathfrak{M}^{K,h}_{\bowtie n}, \mathfrak{D}^K_{\bowtie n}$ modalities becomes a negation of the relation denoted by the $\bowtie$ symbol, hence no dual version is needed for them.

Let Φ be a SOLOIST formula in PNF. Its encoding is a set of QF-EUFIDL constraints over the predicates from P and functions from F. We introduce a predicate $[\![\varphi]\!]$ for each subformula φ of Φ.

We first define the constraints for timing information. As defined in Sect. 2, the temporal structure contains an integer timestamp. An arithmetical variable $|\tau|$ denotes the absolute time at positions $i = 0 \dots H$. Let $\mathscr{C}_{time}$ be the conjunction of the following constraints:

Position i	Timing information	Description
$0 \dots H-1$	$\|\tau\|_i < \|\tau\|_{i+1}$	strict monotonicity

(1)

Next, we define constraints for atomic propositions and propositional operators; their conjunction is denoted as $\mathscr{C}_{prop}$ (where $\leftrightarrow$ stands for a double implication):

Position i	Propositional operators	Description
$0 \dots H$	$[\![p]\!]_i \leftrightarrow p(i)$	atomic propositions
$0 \dots H$	$[\![\neg p]\!]_i \leftrightarrow \neg p(i)$	negation
$0 \dots H$	$[\![\phi \wedge \psi]\!]_i \leftrightarrow [\![\phi]\!]_i \wedge [\![\psi]\!]_i$	conjunction

(2)

Notice that for any sub-formula of the form $\phi \wedge \psi$ in a SOLOIST formula Φ we add in the resulting encoding, instances of formulae from the third row of (2). This encoding completely conforms to the one in [9].

As for the modality U_I, we add to the encoding, for any subformula of the form[5] $\phi \mathsf{U}_{(a,b)} \psi$ in Φ, the following formulae, denoted as $\mathscr{C}_{temp-until}$:

Position i	Temporal operator	Description
$0 \dots H-1$	$[\![\phi \mathsf{U}_{(a,b)} \psi]\!]_i \leftrightarrow \bigvee_{k=i+1}^{H} ([\![\psi]\!]_k \wedge$ $a < \|\tau\|_k - \|\tau\|_i \wedge \|\tau\|_k - \|\tau\|_i < b \wedge \bigwedge_{p=i+1}^{k-1} [\![\phi]\!]_p)$	"Until"
H	$[\![\phi \mathsf{U}_{(a,b)} \psi]\!]_H \leftrightarrow \bot$	"Until" at position H

(3)

[4] Note that the strict semantics of U_I and S_I preserve the duality of R_I and T_I also on finite words.

[5] A closed interval $[a,b]$ over $\mathbb{N}$ can be expressed as an open one of the form $(a-1, b+1)$.

This is a straightforward encoding of the semantics of the "*Until*" operator. The disjunction in the first row represents a case split on all possible future time instants with respect to i. For each such time instant k a conjunction is created with $[\![\psi]\!]_k$ stating that ψ subformula has to hold at time instant k; moreover, ϕ needs to hold in all instants from $i+1$ to $k-1$, i.e., $\bigwedge_{p=i+1}^{k-1}[\![\phi]\!]_p$. Formula $(a < |\tau|_k - |\tau|_i) \wedge (|\tau|_k - |\tau|_i < b)$ enforces the timing constraint of the $\mathsf{U}_{(a,b)}$ modality, i.e., if $\tau_k - \tau_i \in (a,b)$.

The case for the S_I modality is similar to the above. For any sub-formula of the form S_I in Φ we add to the encoding the following formulae, denoted as $\mathscr{C}_{temp-since}$:

Position i	Temporal operator	Description
0	$[\![\phi \mathsf{S}_{(a,b)} \psi]\!]_0 \leftrightarrow \bot$	"Since" at position 0
$1 \dots H$	$[\![\phi \mathsf{S}_{(a,b)} \psi]\!]_i \leftrightarrow \bigvee_{k=0}^{i-1}([\![\psi]\!]_k \wedge$ $a < \|\tau\|_i - \|\tau\|_k \wedge \|\tau\|_i - \|\tau\|_k < b \wedge \bigwedge_{p=k+1}^{i-1}[\![\phi]\!]_p)$	"Since"

(4)

The conjunction of all formulae from $\mathscr{C}_{temp-until}$ and $\mathscr{C}_{temp-since}$ is denoted as $\mathscr{C}_{temp}$.

The $\mathfrak{C}$ modality expresses a bound on the number of occurrences of a certain event in a given time window; in the encoding, it comes natural to use arithmetical variables as counters of the events. For each subformula of the form $\mathfrak{C}^K_{\bowtie n}(\phi)$, we add an arithmetical variable $|c_\phi|$ to F, constrained with the following formulae:

Position i	$\mathfrak{C}$ modality constraints	Description
0	$\|c_\phi\|_0 = 0$	initialization
$0 \dots H-1$	$[\![\phi]\!]_i \rightarrow (\|c_\phi\|_{i+1} = (\|c_\phi\|_i + 1))$	ϕ occurs at i
$0 \dots H-1$	$\neg[\![\phi]\!]_i \rightarrow (\|c_\phi\|_{i+1} = \|c_\phi\|_i)$	ϕ does not occur at i

(5)

The constraint in the first row initializes the arithmetical variable to zero at time instant 0. The following H constraints (in the second row) force $|c_\phi|$ to increase by 1 at time instant $i+1$, if ϕ occurs at time instant i. The last H constraints from the third row refer to the opposite situation: when there is no occurrence of the event ϕ at time instant i, the value of $|c_\phi|_{i+1}$ is constrained to have the same value as $|c_\phi|_i$. Let us denote, for a $\mathfrak{C}$ modality that has ϕ as a sub-formula, the conjunction of these constraints as $\mathscr{C}_{c-cons}(\phi)$. Besides $\mathscr{C}_{c-cons}(\phi)$, we add to the encoding, for each $i = 0 \dots H$, the following constraints, denoted as $\mathscr{C}_{c-form}(\phi)$:

$$[\![\mathsf{C}^K_{\bowtie n}(\phi)]\!]_i \leftrightarrow \bigvee_{z=0}^{min\{i,K\}} |c_\phi|_{i+1} - |c_\phi|_{i-z} \bowtie n \wedge |\tau|_i - |\tau|_{i-z-1} > K \wedge |\tau|_i - |\tau|_{i-z} \leq K$$

This formula characterizes each time instant i of the temporal structure in which the $\mathfrak{C}$ modality is true. The disjunction is a case split for each position z in the past with respect to the current position i. Notice that, if $K > i$ we need to consider all previous positions in the temporal structure; otherwise, it is enough to consider K previous time instants, since in the worst case all timestamps can increase by at least one. Each case is a conjunction where sub-formula $|\tau|_i - |\tau|_{i-z-1} > K \wedge |\tau|_i - |\tau|_{i-z} \leq K$ determines the correct position on the left side of the time window, while $|c_\phi|_{i+1} - |c_\phi|_{i-z} \bowtie n$ checks that the $\mathfrak{C}$ modality holds in the considered time window.

As for the $\mathfrak{M}$ modality, for each subformula of the form $\mathfrak{M}^{K,h}_{\bowtie n}(\phi)$, we introduce the same arithmetical variable $|c_\phi|$ and the constraint $\mathscr{C}_{c-cons}(\phi)$ (now denoted $\mathscr{C}_{m-cons}(\phi)$) as for the $\mathfrak{C}$ modality. Additionally, we add arithmetical variables $|p_0| \dots |p_{\lfloor \frac{K}{h} \rfloor +1}|$ to the set F for each $\mathfrak{M}$ modality sub-formula of Φ. The encoding of the $\mathfrak{M}$ modality depends on the operator $\bowtie$; for example, when the comparison operator is "<" we have the following constraints, denoted $\mathscr{C}_{m-form}(\phi)$:

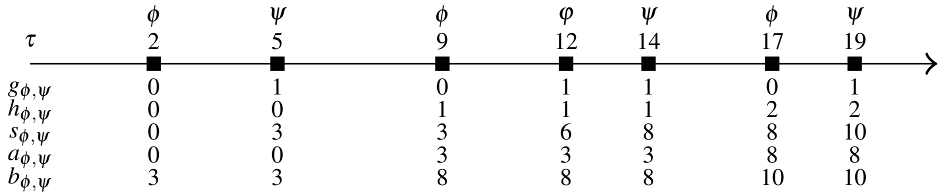
$$\llbracket \mathsf{M}^{K,h}_{<n}(\phi) \rrbracket_i \leftrightarrow \bigwedge_{y=0}^{\lfloor \frac{K}{h} \rfloor} \Big(\bigvee_{z=0}^{min\{i,h\cdot(y+1)\}} \big(|p_{y+1}|_i = |c_\phi|_{i+1} - |c_\phi|_{i-z} \wedge |p_{y+1}|_i - |p_y|_i < n \wedge \\ |\tau|_i - |\tau|_{i-z-1} > (y+1)\cdot h \wedge |\tau|_i - |\tau|_{i-z} \leq (y+1)\cdot h \big) \Big) \wedge |p_0|_i = 0$$

In this formula, in each conjunct y we perform a case split, similar to the case of the $\mathfrak{C}$ modality, but with a different time window: $(y+1)\cdot h$. We assign the result of counting to the variable $|p_{y+1}|$ in each conjunct. Therefore, values $|p_0|_i \dots |p_{\lfloor \frac{K}{h} \rfloor +1}|_i$ contain the number of occurrences of ϕ in time windows $0, h, 2h, \dots, \lfloor \frac{K}{h} \rfloor \cdot h, K$ with respect to position i, respectively. With subformula $|p_{y+1}|_i - |p_y|_i < n$, we check that in each observation subinterval with respect to i there is a bounded number of occurrences. The other cases of $\bowtie$ can be defined in a similar way.

The $\mathfrak{D}$ modality expresses a bound on the average distance between the occurrences of a pair of events in a given time window. Since events can occur multiple times in the temporal structure, a pair of events (ϕ, ψ) may have multiple instances. We call a pair of the form $(\llbracket \phi \rrbracket_i, \llbracket \psi \rrbracket_j)$ *an instance* if there is an occurrence of event ϕ at time instant i and an occurrence of event ψ at time instant j, with $i < j$. We call such instance *open* at time instant q if $i \leq q < j$. Otherwise, the instance is *closed* at time instant q. The *distance* of a closed pair instance is $j - i$; for an open pair at time instant q, the distance is $q - i$. A time window defined for a $\mathfrak{D}^K_{\bowtie n}(\phi, \psi)$ (sub-)formula evaluated at time instant q is bounded by the time instants $q+1$ and $q-K+1$. It has a *left-open* (respectively, *right-open*) pair in position q of a temporal structure, if there is an open instance of (ϕ, ψ) at time instant $q-K+1$ (respectively, $q+1$). Depending on whether a $\mathfrak{D}$ modality (sub-)formula contains either (left- and/ or right-) open pairs or none, there are four distinct cases to take into account for the encoding.

For each subformula of the form $\mathfrak{D}^K_{\bowtie n}(\phi, \psi)$, we add to F five arithmetical variables:

- $|g_{\phi,\psi}|$: it assumes value 1 in the time instants following an occurrence of ϕ and is reset to 0 after an occurrence of ψ. It acts as a flag denoting the time instants during which the event pair instance is open.
- $|h_{\phi,\psi}|$: in each time instant, it contains the number of previously seen closed pair instances. It is increased after every occurrence of ψ.
- $|s_{\phi,\psi}|$: At each time instant, its value corresponds to the sum of distances of all previously occurred pair instances. It is increased after every time instant when either $|g_{\phi,\psi}|$ is 1 or ϕ holds.
- $|a_{\phi,\psi}|$: it keeps track of the sum of the distances of all previously occurred closed pair instances.
- $|b_{\phi,\psi}|$: it has the values that will be assumed by variable $|s_{\phi,\psi}|$ at the next occurrence of ψ (more details below).



	ϕ	ψ	ϕ	φ	ψ	ϕ	ψ
τ	2	5	9	12	14	17	19
$g_{\phi,\psi}$	0	1	0	1	1	0	1
$h_{\phi,\psi}$	0	0	1	1	1	2	2
$s_{\phi,\psi}$	0	3	3	6	8	8	10
$a_{\phi,\psi}$	0	0	3	3	3	8	8
$b_{\phi,\psi}$	3	3	8	8	8	10	10

Fig. 2: Example of trace for the $\mathfrak{D}$ modality, with the corresponding arithmetical variables used in the encoding

Variables $|a_{\phi,\psi}|$, $|b_{\phi,\psi}|$, and $|h_{\phi,\psi}|$ are directly used in the encoding of the $\mathfrak{D}$ modality (sub-)formulae, while variables $|g_{\phi,\psi}|$ and $|s_{\phi,\psi}|$ are helper variables, used to determine the values of the other variables. Figure 2 shows a portion of a trace and the values assumed by these variables: the uppermost row shows instants where *atoms* ϕ, ψ, and φ hold; the second row shows the value of $|\tau|$ at each time instant; the other rows show the values of the variables at each time instant.
For each $\mathfrak{D}^K_{\bowtie n}(\phi,\psi)$ modality sub-formula we define the set of constraints $\mathscr{C}_{d-cons}(\phi,\psi)$:

Position i	$\mathfrak{D}$ modality constraints	Description
0	$\lvert g_{\phi,\psi}\rvert_0 = 0 \wedge \lvert h_{\phi,\psi}\rvert_0 = 0 \wedge \lvert a_{\phi,\psi}\rvert_0 = 0 \wedge \lvert s_{\phi,\psi}\rvert_0 = 0$	variable initialization
0	$[\![B_{eq}]\!]_0$	$\lvert b_{\phi,\psi}\rvert$ initialization
$0\dots H-1$	$[\![\phi]\!]_i \rightarrow (\lvert g_{\phi,\psi}\rvert_{i+1} = 1 \wedge \lvert s_{\phi,\psi}\rvert_{i+1} = \lvert s_{\phi,\psi}\rvert_i + (\lvert\tau\rvert_{i+1} - \lvert\tau\rvert_i) \wedge$ $\lvert h_{\phi,\psi}\rvert_{i+1} = \lvert h_{\phi,\psi}\rvert_i \wedge \lvert a_{\phi,\psi}\rvert_{i+1} = \lvert a_{\phi,\psi}\rvert_i)$	ϕ occurs at i
$0\dots H-1$	$[\![\psi]\!]_i \rightarrow (\lvert g_{\phi,\psi}\rvert_{i+1} = 0 \wedge \lvert h_{\phi,\psi}\rvert_{i+1} = \lvert h_{\phi,\psi}\rvert_i + 1 \wedge$ $\lvert a_{\phi,\psi}\rvert_{i+1} = \lvert s_{\phi,\psi}\rvert_i \wedge \lvert s_{\phi,\psi}\rvert_{i+1} = \lvert s_{\phi,\psi}\rvert_i \wedge$ $\lvert b_{\phi,\psi}\rvert_i = \lvert s_{\phi,\psi}\rvert_i \wedge [\![B_{eq}]\!]_{i+1})$	ψ occurs at i
$0\dots H-1$	$\neg[\![\phi]\!]_i \wedge \neg[\![\psi]\!]_i \rightarrow (\lvert g_{\phi,\psi}\rvert_{i+1} = \lvert g_{\phi,\psi}\rvert_i \wedge \lvert h_{\phi,\psi}\rvert_{i+1} = \lvert h_{\phi,\psi}\rvert_i \wedge$ $\lvert a_{\phi,\psi}\rvert_{i+1} = \lvert a_{\phi,\psi}\rvert_i \wedge$ $(\lvert g_{\phi,\psi}\rvert_i = 1 \rightarrow \lvert s_{\phi,\psi}\rvert_{i+1} =$ $\lvert s_{\phi,\psi}\rvert_i + (\lvert\tau\rvert_{i+1} - \lvert\tau\rvert_i)) \wedge$ $\lvert g_{\phi,\psi}\rvert_i = 0 \rightarrow \lvert s_{\phi,\psi}\rvert_{i+1} = \lvert s_{\phi,\psi}\rvert_i))$	neither ϕ nor ψ occurs at i

(6)

The formula in the first row of (6) initializes all variables at time instant 0 except $|b_{\phi,\psi}|$. In the second row we introduce a new predicate $[\![B_{eq}]\!]$; it has the following constraints:

Position i	$[\![B_{eq}]\!]$ predicate constraints	Description
$0\dots H-1$	$[\![B_{eq}]\!]_i \leftrightarrow [\![\psi]\!]_i \vee ((\lvert b_{\phi,\psi}\rvert_{i+1} = \lvert b_{\phi,\psi}\rvert_i) \wedge [\![B_{eq}]\!]_{i+1})$	propagation of value of $\lvert b_{\phi,\psi}\rvert$
H	$[\![B_{eq}]\!]_H \leftrightarrow \top$	last state constraint

(7)

These constraints force the values of the variables $|b_{\phi,\psi}|_i$ to stay the same in all the consecutive time instants until the first occurrence of ψ or until the end of the trace; the second constraint in (7) deals with traces without occurrences of ψ.

The third constraint in (6) determines the value of variables in the next time instant, upon occurrence of an event ϕ at time instant i. Variable $|g_{\phi,\psi}|_{i+1}$ is set to 1; variable $|s_{\phi,\psi}|_{i+1}$ is incremented by $|\tau|_{i+1} - |\tau|_i$ with respect to value of the variable $|s_{\phi,\psi}|_i$; variables $|h_{\phi,\psi}|_{i+1}$ and $|a_{\phi,\psi}|_{i+1}$ are constrained not to change with respect to value of their counterparts at time instant i. The fourth constraint determines how the variables are updated when an event ψ occurs at time instant i: variable $|g_{\phi,\psi}|_{i+1}$ is set to 0; variables $|b_{\phi,\psi}|_i$, $|a_{\phi,\psi}|_{i+1}$, and $|s_{\phi,\psi}|_{i+1}$ are set to be equal to $|s_{\phi,\psi}|_i$. Moreover, $[\![B_{eq}]\!]_{i+1}$ is constrained to hold, forcing values of $|b_{\phi,\psi}|_j$ to stay the same in all the consecutive time instants $j > i$, until the next occurrence of ψ. The constraints in the fifth row of (6) cover the cases when neither ϕ nor ψ occur at time instant i. In these cases the values of variables $|g_{\phi,\psi}|_{i+1}$, $|h_{\phi,\psi}|_{i+1}$, and $|a_{\phi,\psi}|_{i+1}$ are constrained to have the same value as in their counterparts at i, variable $|b_{\phi,\psi}|_{i+1}$ is unconstrained, while for $|s_{\phi,\psi}|_{i+1}$ we need to distinguish two separate cases. If the last event of the pair is ϕ (denoted by $|g_{\phi,\psi}|_i = 1$), then value of $|s_{\phi,\psi}|_{i+1}$ is $|s_{\phi,\psi}|_i$ incremented by $|\tau|_{i+1} - |\tau|_i$, otherwise it is just $|s_{\phi,\psi}|_i$.

For any sub-formula of the form $\mathsf{D}^K_{\bowtie n}\{(\phi,\psi)\}$ evaluated at time instant i, we add to the encoding the constraint $\mathscr{C}_{d-form}(\phi,\psi)$:

$$
\begin{aligned}
[\![\mathsf{D}^K_{\bowtie n}(\phi,\psi)]\!]_i \leftrightarrow \bigvee_{z=0}^{min\{i,K\}} \Big((\texttt{if}^6\ (|g_{\phi,\psi}|_{i-z} = 1)\ \texttt{then}\ (\tfrac{|a_{\phi,\psi}|_{i+1} - |b_{\phi,\psi}|_{i-z}}{|h_{\phi,\psi}|_{i+1} - |h_{\phi,\psi}|_{i-z} - 1} \bowtie n) \\
\texttt{else}\ (\tfrac{|a_{\phi,\psi}|_{i+1} - |a_{\phi,\psi}|_{i-z}}{|h_{\phi,\psi}|_{i+1} - |h_{\phi,\psi}|_{i-z}} \bowtie n)) \\
\wedge |\tau|_i - |\tau|_{i-z-1} > K \wedge |\tau|_i - |\tau|_{i-z} \le K \Big)
\end{aligned}
$$

In the above formula, the outer disjunction considers all positions that are z time instants in the past with respect to i (i.e., $i - z$) and checks, for each of them, if they fit into the time window using the $|\tau|_i - |\tau|_{i-z-1} > K \wedge |\tau|_i - |\tau|_{i-z} \le K$ formula. If one position does, the rest of the formula considers whether there is an *open* (ϕ,ψ) pair instance at that position which is captured by the $|g_{\phi,\psi}|_{i-z} = 1$ formula. In such a case, we compute the total delay between all pair instances within the time window by subtracting variable $|b_{\phi,\psi}|$ from $|a_{\phi,\psi}|$ at the appropriate positions. Since the value of $|b_{\phi,\psi}|$ at each position contains the value of $|s_{\phi,\psi}|$ at the position of the next occurrence of ψ, we effectively ignore the delay of the left-open pair. Otherwise, we use variable $|a_{\phi,\psi}|$, since it contains the delay from the last *closed* pair instance. Fractions in this formula are used for the sake of clarity, however the actual formula conforms to IDL due to the fact that n is a constant and $\frac{A}{B} = n$ can be written as $A = \underbrace{B + B + \ldots + B}_{\text{n times}}$.

The final QF-EUFIDL formula obtained from the encoding of the input SOLOIST formula Φ is the following conjunction of (possibly empty) formulae, which is supplied to the SMT solver: $[\![\Phi]\!]_0 \wedge \mathscr{C}_{time} \wedge \mathscr{C}_{prop} \wedge \mathscr{C}_{temp} \wedge \mathscr{C}_c \wedge \mathscr{C}_m \wedge \mathscr{C}_d$, where $\mathscr{C}_c \leftrightarrow \mathscr{C}_{c-cons} \wedge \mathscr{C}_{c-form}$, $\mathscr{C}_m \leftrightarrow \mathscr{C}_{m-cons} \wedge \mathscr{C}_{m-form}$ and $\mathscr{C}_d \leftrightarrow \mathscr{C}_{d-cons} \wedge \mathscr{C}_{d-form}$.

[6] "`if` A `then` B `else` C" can be written as $(A \wedge B) \vee (\neg A \wedge C)$

Complexity. We provide an estimation of the size of the QF-EUFIDL formula corresponding to a temporal or aggregating modality of SOLOIST. Although the syntactic complexity of the translation is already known in the case of standard LTL temporal modalities (e.g., [9]), we still provide a measure for U_I and S_I, since we rely on an ad-hoc encoding.

Let us consider first $\phi \mathsf{U}_I \psi$; the case for $\phi \mathsf{S}_I \psi$ is similar. At position $0 \leq i \leq H$, the formula in (3) has size $\mathcal{O}(H-i)^2$. We have then $\sum_{i=0}^{H} \mathcal{O}(H-i)^2 < \mathcal{O}(H^3)$.

Let μ be the maximum constant occurring in the SOLOIST formula and in the trace. One variable $|c_\phi|$ is required for all formulae $\mathfrak{C}^K_{\bowtie n}(\phi)$ with the same argument ϕ. In the worst case, we introduce one variable for each one. At position $0 \leq i \leq H$, formula $[\![\mathsf{C}^K_{\bowtie n}(\phi)]\!]_i$ has size $\mathcal{O}(i)$. We have then $\sum_{i=0}^{H} \mathcal{O}(\log(\mu) i) < \mathcal{O}(\log(\mu) H^2)$. The $\mathfrak{U}$ modality is defined through $\mathfrak{C}$ and, therefore, inherits the same syntactic complexity.

Encoding of formula $\mathfrak{M}^{K,h}_{\bowtie n}(\phi)$ requires one variable $|c_\phi|$. We can reuse variable c_ϕ if in the original SOLOIST formula there are $\mathfrak{M}$ formulae or $\mathfrak{C}$ formulae with the same argument ϕ. Moreover, for each $\mathfrak{M}$ we need also $\lfloor \frac{K}{h} \rfloor + 2$ arithmetical variables $|p_0| \ldots |p_{\lfloor \frac{K}{h} \rfloor + 1}|$. In the worst case, we introduce $\lfloor \frac{K}{h} \rfloor + 3$ variables for each formula $\mathfrak{M}^{K,h}_{\bowtie n}(\phi)$. At position $0 \leq i \leq H$, formula $[\![\mathsf{M}^{K,h}_{\bowtie n}(\phi)]\!]_i$ has size $\mathcal{O}(\log(\mu) \frac{K}{h} \cdot i)$. We have then $\sum_{i=0}^{H} \mathcal{O}(\log(\mu) \frac{K}{h} i) < \mathcal{O}(\log(\mu) \frac{K}{h} H^2)$.

The set of formulae translating $\mathfrak{D}$ is defined by the conjunction of formulae in (6) and (7) in addition to constraint $\mathscr{C}_{d-form}$. For each formula $\mathfrak{D}$ we introduce five variables related to the pair (ϕ, ψ). The size of formulae in (6) and in (7) is $\mathcal{O}(H)$. Constraint $\mathscr{C}_{d-form}$ requires a more careful analysis; notice that its size depends on the parameter n because of the way formula $\frac{a}{b} < n$ is expanded. At position $0 \leq i \leq H$, formula $[\![\mathsf{D}^K_{\bowtie n}(\phi, \psi)]\!]_i$ has size $\mathcal{O}(\log(\mu) in)$. Then, the complexity for $\mathfrak{D}$ is obtained by $\sum_{i=0}^{H} \mathcal{O}(\log(\mu) in) < \mathcal{O}(\log(\mu) n H^2)$.

The size of the QF-EUFIDL encoding of a SOLOIST formula of length λ is $\mathcal{O}(\lambda \log(\mu)(H^3 + \frac{K}{h} H^2 + n H^2))$, as the number of sub-formulae is polynomial in λ, whereas the size of the encoding of a trace is $\mathcal{O}(\log(\mu) H)$. In the worst-case, $K = H, h = 1$, hence the overall size of the QF-EUFIDL formula encoding a trace checking problem is $\mathcal{O}(\lambda \log(\mu) H^3)$. Finally, we notice that the complexity of trace checking SOLOIST formulae is NP-complete. In fact, since size is polynomial and satisfiability of QF-EUFIDL is NP-complete, then the complexity of solving one instance of the problem is NP; NP-hardness may easily obtained by reducing SAT to trace checking SOLOIST formulae.

4 Evaluation

We implemented the encoding as a Common Lisp plugin for the ZOT verification toolset[7]. Before reporting the results of the evaluation of the implementation of the proposed encoding, we first define a metric to characterize the degree of sparseness for the execution traces to be checked. Let ξ be the number of valid time instants in a trace, i.e., the instants in which at least one event occurs. This number corresponds to the number of positions in a timed word modeling the trace. Let ν denote the number of

[7] `http://code.google.com/p/zot/`.

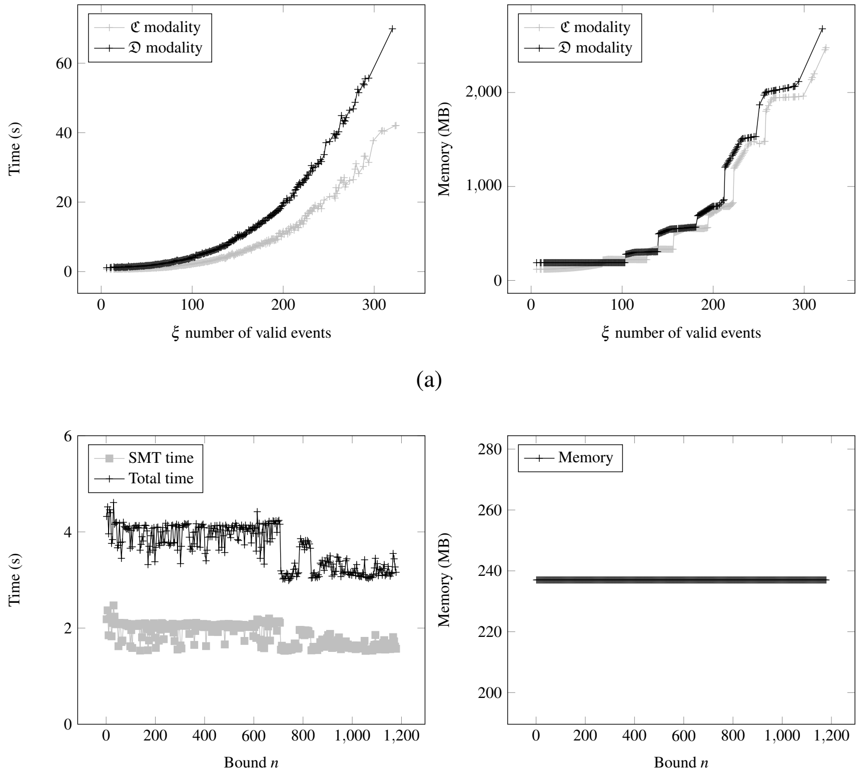
non-valid time instants, i.e., those where no event occurs. Notice that, in timed words, these events are abstracted away by using timestamps. We can use the total length of a trace $\xi + \nu$ to compute the degree of sparseness as $\varsigma = \frac{\xi}{\xi+\nu}$.

Scalability. To show how scalable the proposed encoding is with respect to the parameters mentioned in Sect. 3, we synthesized traces using the PLG (Process Log Generator) tool [10]. This tool can generate traces that conform to the business logic of the process given in input, varying the trace length and the number of valid time instants. We used a variant of the *ATMFrontEnd* business process example from the JBoss jBPM distribution; this process provides customers with some operations to interact with their bank account, such as `query-balance`, `withdraw`, and `deposit`. For space reasons we only report the evaluation of checking properties expressed using the $\mathfrak{C}$ and $\mathfrak{D}$ modalities. We considered the two properties P1: "The number of `query-balance` operations performed in the last 10 minutes is less than 10" and P2: "The average response time of the `query-balance` operation is less than n seconds in the last 6 hours", which can be expressed in SOLOIST as $\mathfrak{C}^{600}_{<10}(QB_s)$ and $\mathfrak{D}^{21600}_{<n}(QB_s, QB_e)$, respectively. Notice that we express time in seconds and use events QB_s and QB_e to denote, respectively, the start and the end of operation `query-balance`.

For both modalities we consider as parameter the number of valid time instants ξ, i.e., the length H of the temporal structure; for the $\mathfrak{D}$ modality we also consider the varying bound n. The plots in Fig. 3a show quadratic increase in memory usage and time with respect to the number of valid time instants, as anticipated in Sect. 3. In addition, the plots in Fig. 3b show that parameter n does not affect the computational time and space. Although in the complexity analysis we theoretically determined that the size of the encoding for the $\mathfrak{D}$ modality linearly depends on n, the evaluation showed that in the actual implementation this does not happen, because the SMT decision procedure supports natively the use of multiplication of terms by a constant. This allows us to write a more concise encoding for $\mathfrak{D}$ modality in $\mathscr{O}(H^2)$.

Application to a Realistic Example. We have applied our approach also to a realistic example, a sample service composition called ACME BOT [19], whose monitoring data are available[8] as part of the "S-Cube Use Case Repository". We reconstructed 9796 execution traces, based on the monitoring data of the corresponding service composition instances. On each of these traces, we performed trace checking with respect to two simple properties, one containing the $\mathfrak{C}$ modality, and the other the $\mathfrak{D}$ modality. In the first case, trace checking took on average 0.672s with a standard deviation of 0.035s and used on average 125.7MB of memory with 0.476MB standard deviation; for the checks with the $\mathfrak{D}$ modality, it took on average 0.813s with 0.032s standard deviation and used on average 127.7MB of memory with 0.476MB standard deviation. On average, each trace had 31.5 valid time instants and a total length of 39341.3; the average degree of sparseness was then 0.08%. This example shows that our approach can efficiently check properties of realistic service compositions.

[8] `http://scube-casestudies.ws.dei.polimi.it/index.php/`.



(a)

(b)

Fig. 3: Scalability of the encoding with respect to: (a) the number of valid time instants ξ in the trace, in the case of the $\mathfrak{C}$ and $\mathfrak{D}$ modalities; (b) bound n, in the case of the $\mathfrak{D}$ modality

Discussion and trade-offs. As you can see from Fig. 3a, our approach can support the checking of traces containing up to 300 valid time instants, using up to 2GB of memory. The strength of the approach is that the number of non-valid time instants in the trace does not affect its scalability. In principle, we can deal with traces of arbitrary length, with varying degrees of sparseness, and still use up to 2GB of memory if the trace contains at most 300 valid time instants. The realistic example described above as well as the process log of the BPI challenge mentioned earlier show that execution traces with a limited number of valid time instants and a low degree of sparseness can be very common in enterprise service-based applications. We compared the performance of the approach proposed in this paper with a previous, not-optimized implementation [18] based on CLTLB($\mathscr{D}$); for the evaluation, we varied the degree of sparseness in the traces and their total length $\xi + \nu$. The approach in [18] does not keep track of timing information and therefore has to enumerate both valid and non-valid time instants. Figure 4 shows the results of this comparison, in terms of time and memory usage: the

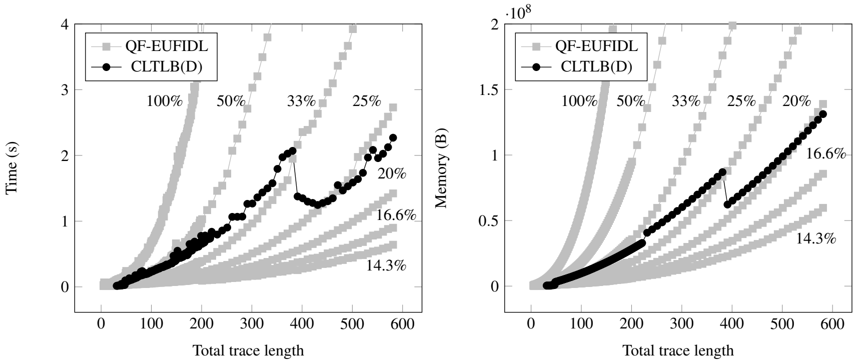


Fig. 4: Tradeoff between the trace checking approach based on CLTLB($\mathscr{D}$) [18] and the one based on the QF-EUFIDL encoding, with respect to the degree of sparseness of the trace

black line shows the scalability of the approach based on CLTLB($\mathscr{D}$) from [18], while the seven gray lines correspond to the QF-EUFIDL-based approach presented in this paper, applied to traces with different degrees of sparseness (100%, 50%, 33%, 25%, 20%, 16.6%, and 14.3%, from left to right, respectively). The results show that the approach presented in this paper is more efficient than the one presented in [18] when the degree of sparseness of input traces is less than 25%.

5 Conclusions

We have shown how trace checking for SOLOIST can be reduced to an existential satisfiability problem for the QF-EUFIDL logic, which can be solved with efficient decision procedures using SMT verifiers. We motivate our approach with two case studies and provide complexity analysis as well as practical evaluation. In the future, we plan to collaborate with an industrial partner, to apply our trace checking approach to more examples of realistic execution traces.

Acknowledgments. This work has been supported by the European Community under the IDEAS-ERC grant agreement no. 227977-SMScom and by the National Research Fund, Luxembourg (FNR/P10/03).

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoreotical Computer Science 126(2), 183–235 (Apr 1994)
2. Basin, D., Klaedtke, F., Marinovic, S., Zălinescu, E.: Monitoring of temporal first-order properties with aggregations. In: Proc. RV'13. LNCS, vol. 8174, pp. 40–58. Springer (2013)

3. Bersani, M.M., Frigeri, A., Morzenti, A., Pradella, M., Rossi, M., San Pietro, P.: Bounded reachability for temporal logic over constraint systems. In: Proc. of TIME'10. pp. 43–50. IEEE Computer Society (2010)
4. Bersani, M.M., Frigeri, A., Rossi, M., San Pietro, P.: Completeness of the bounded satisfiability problem for constraint LTL. In: Proc. of RP 2011. LNCS, vol. 6945, pp. 58–71. Springer (2011)
5. Bersani, M.M., Rossi, M., San Pietro, P.: Deciding continuous-time metric temporal logic with counting modalities. In: Proc. of RP 2013. LNCS, vol. 8169, pp. 70–82. Springer (2013)
6. Bersani, M.M., Rossi, M., San Pietro, P.: On the satisfiability of metric temporal logics over the reals. In: Proc. of AVOCS'13 (2013)
7. Bianculli, D., Ghezzi, C., Pautasso, C., Senti, P.: Specification patterns from research to industry: a case study in service-based applications. In: Proc. of ICSE 2012. pp. 968–976. IEEE Computer Society (2012)
8. Bianculli, D., Ghezzi, C., San Pietro, P.: The tale of SOLOIST: a specification language for service compositions interactions. In: Proc. of FACS'12. LNCS, vol. 7684, pp. 55–72. Springer (2013)
9. Biere, A., Heljanko, K., Junttila, T.A., Latvala, T., Schuppan, V.: Linear encodings of bounded LTL model checking. Logical Methods in Computer Science 2(15) (2006)
10. Burattin, A., Sperduti, A.: PLG: A framework for the generation of business process models and their execution logs. In: BPM Workshops. LNBIP, vol. 66, pp. 214–219. Springer (2011)
11. Demri, S., D'Souza, D.: An automata-theoretic approach to constraint LTL. Inf. Comput. 205(3), 380–415 (Mar 2007)
12. van Dongen, B.: BPI challenge 2012 (2012), `http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f`
13. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Property specification patterns for finite-state verification. In: Proc. of FMSP '98. pp. 7–15. ACM (1998)
14. Felder, M., Morzenti, A.: Validating real-time systems by history-checking TRIO specifications. ACM Trans. Softw. Eng. Methodol. 3(4), 308–339 (Oct 1994)
15. Finkbeiner, B., Sankaranarayanan, S., Sipma, H.: Collecting statistics over runtime executions. Form. Method Syst. Des. 27, 253–274 (2005)
16. Kindermann, R., Junttila, T.A., Niemelä, I.: Bounded model checking of an MITL fragment for timed automata. CoRR abs/1304.7209 (2013)
17. Konrad, S., Cheng, B.H.C.: Real-time specification patterns. In: Proc. of ICSE '05. pp. 372–381. ACM (2005)
18. Krstić, S.: Verification of quantitative properties of service-based applications. Master's thesis, Politecnico di Milano (December 2012)
19. Leitner, P., Hummer, W., Dustdar, S.: A Monitoring Data Set for Evaluating QoS-Aware Service-Based Systems. In: Proc. of PESOS 2012. pp. 67–68 (2012)
20. Mrad, A., Ahmed, S., Hallé, S., Beaudet, E.: Babeltrace: A collection of transducers for trace validation. In: Proc. of RV 2012. LNCS, vol. 7687, pp. 126–130. Springer (2013)
21. Nelson, G., Oppen, D.C.: Simplification by cooperating decision procedures. ACM Trans. Program. Lang. Syst. 1(2), 245–257 (Oct 1979)
22. Pradella, M., Morzenti, A., San Pietro, P.: The symmetry of the past and of the future: bi-infinite time in the verification of temporal properties. In: Proc. of ESEC-FSE '07. pp. 312–320. ACM (2007)
23. Pradella, M., Morzenti, A., San Pietro, P.: Bounded satisfiability checking of metric temporal logic specifications. ACM Trans. Softw. Eng. Methodol. 22(3), 20:1–20:54 (Jul 2013)
24. Wozna-Szczesniak, B., Zbrzezny, A.: Checking MTL properties of discrete timed automata via bounded model checking. In: CS&P. vol. 1032, pp. 469–477. CEUR-WS.org (2013)