Evaluation of temporal datasets via interval temporal logic model checking*

Dario Della Monica[1,2], David de Frutos-Escrig[1], Angelo Montanari[3], Aniello Murano[2], and Guido Sciavicco[4]

1 **Dept. Sistemas Informáticos y Computación, Universidad Complutense de Madrid (Spain)**
ddellamo@ucm.es,defrutos@sip.ucm.es
2 **Dept. of Electrical Engineering and Information Technology, University "Federico II" of Naples, (Italy)**
dario.dellamonica@unina.it,murano@na.infn.it
3 **Dept. of Mathematics, Computer Science, and Physics, University of Udine (Italy)**
angelo.montanari@uniud.it
4 **Dept. of Mathematics and Computer Science, University of Ferrara (Italy)**
guido.sciavicco@unife.it

Abstract

The problem of *temporal dataset evaluation* consists in establishing to what extent a set of temporal data (histories) complies with a given temporal condition. It presents a strong resemblance with the problem of model checking enhanced with the ability of *rating* the compliance degree of a model against a formula. In this paper, we solve the temporal dataset evaluation problem by suitably combining the outcomes of model checking an interval temporal formula against sets of histories (finite interval models), possibly taking into account domain-dependent measures/criteria, like, for instance, sensitivity, specificity, and accuracy. From a technical point of view, the main contribution of the paper is a (deterministic) polynomial time algorithm for interval temporal logic model checking over finite interval models. To the best of our knowledge, this is the first application of interval temporal logic model checking in the area of temporal databases and data mining (rather than in the formal verification setting).

1998 ACM Subject Classification F.4.1 Mathematical Logic, I.2.6 Learning, H.2.1 Logical Design

Keywords and phrases Dataset Evaluation; Temporal Databases; Model Checking; Interval Temporal Logics

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.23

1 Introduction

Temporal databases keep track of the temporal evolution of information by associating one or more temporal dimensions with stored data [10, 19, 37, 38]. One of the fundamental temporal dimensions is *valid* time that associates with each stored fact the time interval at which it is true in the modeled reality. In this paper, we focus on temporal databases

* This work was partially supported by the Italian INdAM-GNCS project *Logics and Automata for Interval Model Checking*. In addition, Dario Della Monica acknowledges the financial support from a Marie Curie INdAM-COFUND-2012 Outgoing Fellowship.

featuring such a dimension (valid-time temporal databases). A valid-time temporal database consists of a set of temporal instances, or *histories*. Borrowing the terminology from the field of machine learning, a set of histories is often referred to as a *temporal dataset* [41].

The *temporal dataset evaluation problem* can be (roughly) defined as the problem of establishing how many histories comply with a given temporal formula or constraint. The evaluation of temporal datasets plays a key role in at least three different application domains: *(i) temporal query processing* [37], where the set of histories that satisfy a given condition must be returned, *(ii) temporal constraint checking* [38], where the set of histories that violate a given constraint must be identified, and *(iii) rule evaluation* [41], where one must determine how many histories, and to which extent, comply with a certain temporal rule. In this paper, we formulate (and solve) this problem as a model checking problem.

Interval temporal logic model checking. In its standard formulation, *model checking* is the problem of verifying whether or not a given formula of some logical language is satisfied by a certain model [9]. One of its most successful applications is the verification of a point-based temporal logic specification (expressed, for instance, by a formula of Linear Temporal Logic [31]) against some reactive system description [32]. It is well known that query evaluation and constraint checking in relational databases can be naturally expressed as model checking problems (see, for instance, [40]). The applicability of model checking techniques for the retrieval and the verification of temporal data has also been explored in the literature, e.g., [4, 35].

Time intervals are commonly used to represent temporal information in (valid-time) temporal databases. On the one hand, they allow one to compactly represent the time periods over which data are valid in the modeled domain [39]; on the other hand, they make it possible to suitably represent inherently interval-based temporal information such as telic facts and temporal aggregations [3, 18]. Accordingly, temporal queries, constraints, and rules can be naturally formulated as formulas of an interval temporal logic to be evaluated over temporal datasets represented as finite interval models. The problem of evaluating a temporal dataset can thus be reduced to the model checking problem for interval temporal logic formulas, making it possible to exploit techniques and tools from logic and formal methods to address and solve problems in temporal databases and data mining (see, for instance, the three aforementioned application domains).

As a matter of fact, there is a little mismatch between the expected outcomes of the two problems: while model checking is a decision problem, which returns "yes" when the model meets the specification and "no" otherwise, the problem of temporal dataset evaluation directly or indirectly returns a set of histories (in rule evaluation, it determines to what extent each single history (resp., the whole set of histories) complies with a certain temporal rule). An actual solution to the problem of temporal dataset evaluation can be obtained by representing a temporal dataset as a set of finite interval models and by suitably combining the outcomes of model checking each single model against the formula. In the case of rule evaluation, some domain-dependent measures for *rating* the compliance degree of the interval model with respect to the formula, like, for instance, sensitivity, specificity, and accuracy [41], are also needed.

Our contribution. The contribution of the present paper is twofold. From a methodological perspective, it proposes interval temporal logic model checking as a viable logical tool for temporal dataset evaluation (in particular, for rule evaluation in valid-time temporal databases), thus establishing a formal connection between the two problems. From a technical point of view, it provides an efficient solution to the finite model checking problem for the well-known Halpern and Shoham's interval temporal logic (HS for short) [15], which features

one modality for each Allen relation [2], by devising a deterministic model checking algorithm that runs in polynomial time, thus proving that the problem is in PTIME.

Related work. In the last years, the model checking problem for interval temporal logic has received an increasing attention as an alternative to the traditional (point-based) temporal logic model checking, which can be recovered as a special case. Model checking for full HS, interpreted over finite Kripke structures according to the *state-based semantics* (we refer here to the terminology introduced in [6]), has been studied in [26, 30]. The authors showed that, under the homegeneity assumption, which constrains a proposition letter to hold over an interval if and only if it holds over each component state, the problem is non-elementarily decidable (EXPSPACE-hardness has been later shown in [5]). Since then, the attention was brought to HS fragments, which are often computationally much better [5, 7, 27, 28, 29]. The model checking problem for some HS fragments extended with epistemic operators has been investigated in [22, 23]. The semantic assumptions for these epistemic HS fragments differ from those of [26, 30] in two important respects, making it difficult to compare the two families of logics: formulas are interpreted over the unwinding of the Kripke structure (*computation-tree-based semantics*, borrowing the terminology from [6]) and interval labeling takes into account only the endpoints of intervals. The latter assumption has been later relaxed by making it possible to use regular expressions to define the labeling of proposition letters over intervals in terms of the component states [24].

A common feature of the application of (interval) model checking to the verification of temporal properties of a reactive system is the encoding of all its possible executions by a finite-state transition system, that is, by a finite Kripke structure, which provides an abstract representation of possibly infinitely many interval models. On the contrary, (valid-time) temporal databases commonly assume a given structure of time. Hence, when used for the evaluation of temporal datasets, interval model checking is applied to finite, concrete interval models and makes no restrictive assumptions on interval labeling such as, for instance, the homogeneity assumption. In fact, representing temporal datasets as suitable Kripke structures over which to evaluate formulas is in principle possible. However, it would be both artificial and computationally inconvenient. As we will show, (finite) interval model checking for temporal dataset evaluation behaves computationally much better than interval model checking for system verification: we devise a polynomial model checking algorithm for full HS, while the complexity of model checking HS fragments against Kripke structures goes from coNP-complete to non-elementary, depending on the particular fragment of HS under consideration [5, 7, 22, 23, 24, 26, 27, 28, 29, 30].

Organization of the paper. The rest of the paper is organized as follows. In the next section, we introduce the logic HS and define the finite interval model checking problem. Then, in Section 3, we illustrate a range of possible applications of temporal dataset evaluation. Finally, in Section 4, we prove that the finite interval model checking problem is in PTIME. Conclusions provide an assessment of the work done and outline future research directions.

2 Preliminaries

Let $\mathbb{D} = \langle D, < \rangle$ be a linear order. A *strict* (resp., *non-strict*) interval over $\mathbb{D}$ is an ordered pair $[x, y]$, where $x, y \in D$ and $x < y$ (resp., $x \leq y$). As it is usually the case with the recent literature, we adopt the *strict semantics*, which admits strict intervals only. Such a choice conforms to the definition of interval given by Allen in [2], but it differs from the one by Halpern and Shoham [15]. We denote by $\mathbb{I}(\mathbb{D})$ the set of (strict) intervals over a linear order $\mathbb{D}$. If we exclude the identity relation, there are 12 different relations between two intervals

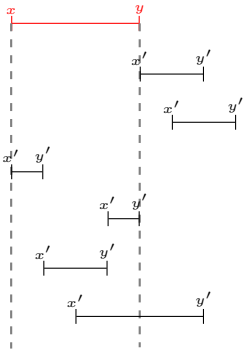
HS	Allen's relations	Graphical representation
$\langle A \rangle$	$[x,y]R_A[x',y'] \Leftrightarrow y = x'$	
$\langle L \rangle$	$[x,y]R_L[x',y'] \Leftrightarrow y < x'$	
$\langle B \rangle$	$[x,y]R_B[x',y'] \Leftrightarrow x = x', y' < y$	
$\langle E \rangle$	$[x,y]R_E[x',y'] \Leftrightarrow y = y', x < x'$	
$\langle D \rangle$	$[x,y]R_D[x',y'] \Leftrightarrow x < x', y' < y$	
$\langle O \rangle$	$[x,y]R_O[x',y'] \Leftrightarrow x < x' < y < y'$	

Figure 1 Allen's interval relations and HS modalities.

in a linear order, often called *Allen's relations* [2]: the six relations R_A (adjacent to), R_L (later than), R_B (begins), R_E (ends), R_D (during), and R_O (overlaps), depicted in Figure 1, and their inverses, that is, $R_{\overline{X}} = (R_X)^{-1}$, for each $X \in \mathcal{A}$, where $\mathcal{A} = \{A, L, B, E, D, O\}$. We associate a universal modality $[X]$ and an existential one $\langle X \rangle$ with each Allen relation R_X. For each $X \in \mathcal{A}$, the *transposes* of the modalities $[X]$ and $\langle X \rangle$ are respectively the modalities $[\overline{X}]$ and $\langle \overline{X} \rangle$, corresponding to the inverse relation $R_{\overline{X}}$ of R_X, and vice versa.

The logic HS. ([15]) Halpern and Shoham's HS can be viewed as a multi-modal logic whose formulas are built from a finite, non-empty set $\mathcal{AP}$ of *atomic propositions* (also referred to as *proposition letters*), the classical Boolean connectives, and a modality for each Allen relation:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle X \rangle \varphi \mid \langle \overline{X} \rangle \varphi,$$

where $p \in \mathcal{AP}$ and $X \in \mathcal{A}$. The other Boolean connectives and the logical constants, e.g., $\rightarrow$ and $\top$, as well as the universal modalities $[X]$, can be defined in the standard way. As shown in [15], the modalities $\langle A \rangle, \langle B \rangle$, and $\langle E \rangle$, along with their transposes, are sufficient to express all the other modalities through a formula of polynomial size. Thus, w.l.o.g., hereafter we restrict ourselves to HS formulas over the set of modalities $\{\langle A \rangle, \langle \overline{A} \rangle, \langle B \rangle, \langle \overline{B} \rangle, \langle E \rangle, \langle \overline{E} \rangle\}$.

The semantics of HS formulas is given in terms of *interval models* $M = \langle \mathbb{D}, V \rangle$, where $\mathbb{D}$ is a linear order and $V : \mathcal{AP} \rightarrow 2^{\mathbb{I}(\mathbb{D})}$ is a *valuation function* which assigns to each atomic proposition $p \in \mathcal{AP}$ the set of intervals $V(p)$ on which p holds. (We indistinctly treat valuation functions as functions from atomic propositions to sets of intervals or, vice versa, from intervals to sets of atomic propositions.) In this work, we are interested in finite structures and thus we restrict our attention to linear orders over finite domains. Any finite linear order $\mathbb{D}$ of size n can be compactly represented by $[n] = \{x \in \mathbb{N} \mid x \leq n\}$. In the following, we will make use of such a representation.

The *truth* of a formula φ on a given interval $[x, y]$ in an interval model M is defined by structural induction on formulas as follows:

- $M, [x, y] \Vdash p$ iff $[x, y] \in V(p)$, for $p \in \mathcal{AP}$;
- $M, [x, y] \Vdash \neg\psi$ iff $M, [x, y] \nVdash \psi$;
- $M, [x, y] \Vdash \psi \vee \xi$ iff $M, [x, y] \Vdash \psi$ or $M, [x, y] \Vdash \xi$;
- $M, [x, y] \Vdash \langle X \rangle \psi$ iff there exists $[w, z]$ such that $[x, y]R_X[w, z]$ and $M, [w, z] \Vdash \psi$.

We denote the modal depth of an HS formula φ by $md(\varphi)$. In [1], it has been shown that, over finite temporal domains, an HS formula φ can be translated into an equivalent one, say it φ', that involves neither $\langle A \rangle$ nor $\langle \overline{A} \rangle$. On the one hand, since the size of φ' may

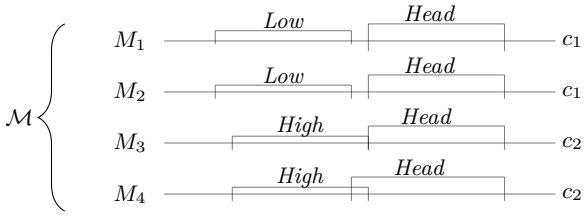


Figure 2 A classification model with four patients partitioned into two classes (c_1 and c_2).

be exponential in the size of φ, we cannot give up modalities $\langle A \rangle$ and $\langle \overline{A} \rangle$ when addressing complexity issues. On the other hand, since $md(\varphi')$ is only linearly larger than $md(\varphi)$, such a translation will come in handy when proving Lemma 1 (bisimulation lemma) in Section 4, allowing us to ignore modalities $\langle A \rangle$ and $\langle \overline{A} \rangle$ in that context.

In the following, we will make use of the *global* modality $[U]$, that allows one to assert a property φ on the entire model. HS is powerful enough to express such a modality as follows:

$$[U]\varphi \stackrel{def}{\equiv} \varphi \wedge \bigwedge_{X \in \mathcal{A}} ([X]\varphi \wedge [\overline{X}]\varphi).$$

The finite HS model checking problem. Given a pair $\mathcal{I} = (M, \varphi)$, where M is a finite interval model and φ is an HS formula, the *finite HS model checking problem* (MC for short) consists in deciding whether $M, [0,1] \Vdash \varphi$. The pair $\mathcal{I} = (M, \varphi)$ is called an *instance* of MC. With a little abuse of the notation, for a given instance (M, φ) of MC, we write $(M, \varphi) \in \mathsf{MC}$ to indicate that $M, [0,1] \Vdash \varphi$. In such a case, we say that MC applied to (M, φ), denoted by $\mathsf{MC}(M, \varphi)$, returns true. We say that two instances $\mathcal{I}, \mathcal{I}'$ of MC are *equivalent*, denoted by $\mathcal{I} \equiv_{\mathsf{MC}} \mathcal{I}'$ whenever $\mathsf{MC}(\mathcal{I})$ returns true if and only if $\mathsf{MC}(\mathcal{I}')$ does. Finally, we say that two HS formulas φ_1 and φ_2 are *equivalent*, denoted by $\varphi_1 \equiv \varphi_2$, if $(M, \varphi_1) \equiv_{\mathsf{MC}} (M, \varphi_2)$ for every interval model M.

3 Temporal Dataset Evaluation

The problem of *temporal dataset evaluation* can be defined as the problem of evaluating how many histories comply with a given temporal formula. A history can be seen as the entire (finite) flow of information relative to a given entity over a bounded time span. As an example, in the medical context, a history is the *medical history* of a patient, that is, the collection of all relevant pieces of information about tests, results, symptoms, and hospitalizations of the patient that occurred during the entire observation period. In Figure 2, we graphically depict (a simplified version of) the history of four patients.

As far as qualitative reasoning is concerned, it turns out to be convenient to assume that all the meaningful events have been suitably *discretized*. As a concrete example, having a fever can be represented by the propositional letter *Low*—meaning lower than 40 degrees—or *High*—meaning higher than or equal to 40 degrees. Similarly, the proposition letter *Head* can be used to indicate the presence of a headache.

The case of fever is an illuminating example of the fact that information about histories is naturally *interval-based* and quite often *non-homogeneous*. Suppose, indeed, that a certain patient is experiencing low fever in an interval $[x, y]$, say, a day, and that during just one hour of that day, that is, over the interval $[w, z]$ strictly contained in $[x, y]$, he/she has an

episode of high fever. A natural choice is to represent such a situation by labeling the interval $[x, y]$ with *Low* and its sub-interval $[w, z]$ with *High*. Notice that such a representation is compatible with a general consistency requirement such as $[U](Low \rightarrow \neg High)$. On the other hand, representing the same pieces of information as three intervals $[x, w], [w, z], [z, y]$ respectively labeled with *Low*, *High*, and *Low*, which would be the case with a point-based representation (or with an interval-based representation under the homogeneity assumption), would be unnatural, and it would entail hiding a potentially important information like: "*the patient presented low fever during the entire day, except for a brief episode of high fever*".

Let us denote a temporal dataset by $\mathcal{D} = \{M_1, \ldots, M_r\}$, where each history M_i, with $1 \leq i \leq r$, is, in fact, an interval model as defined in Section 2. Temporal dataset evaluation can be naturally formulated as a model checking problem for an interval temporal logic like HS. In the following, we briefly elaborate on its concrete application to temporal query processing, temporal constraint checking, and machine learning (rule evaluation).

Evaluating a query over a temporal database corresponds to extracting the set of histories that satisfy the conditions of the query. Temporal queries are typically expressed in a temporal query language like, for instance, TSQL (Temporal SQL) [37], which features operators for Allen's relations that make it possible to check all possible temporal (interval) relationships between pairs of events. The interval temporal logic HS is powerful enough to capture all TSQL operators, which can actually be expressed as suitable disjunctions of HS modalities. To answer a temporal query, an evaluation of the whole dataset of histories is, in general, needed, that returns precisely those histories on which the model checking problem is positively solved. Therefore, it can be naturally viewed as a dataset evaluation problem. As an example, determining the patients that have experienced a headache immediately after an episode of high fever amounts to looking for those histories M_i such that:

$$M_i, [0, 1] \Vdash \langle L \rangle (High \wedge \langle A \rangle Head).$$

In the example of Figure 2, only M_3 makes the formula true.

As far as *temporal constraint checking* is concerned, in a (temporal) database one may impose various different types of (temporal) constraint at design time [21]. They range from very simple *domain* constraints , like, for instance, "each value of a given attribute must belong to a specific domain", to *key* constraints, that is, existence and uniqueness of the values of the key attributes, and *functional dependencies*, which require some attributes to functionally depend on other ones (in the context of temporal databases, interval-based temporal functional dependencies have been studied in [11]). More advanced temporal constraints, that involve more complex relationships among the values of each history, can be expressed as logical formulas, and checking such constraints corresponds to solving the temporal dataset evaluation problem and identifying precisely those histories for which the model checking problem returns false. Let us consider again the simple example reported in Figure 2. In order to check that data about the values of the fever parameter over time, for any given patient, are consistent, that is, to exclude that there exists a patient such that both *High* and *Low* hold in the same time interval, it suffices to check that the number of histories M_i such that:

$$M_i, [0, 1] \Vdash [U](High \wedge Low)$$

is equal to 0 (this is actually the case with the temporal dataset in Figure 2).

Finally, one of the most interesting and extensively studied problems in *machine learning* is supervised classification. In such a problem, each history is assigned to a *class* c from a finite set $\mathcal{C}$ with the aim of devising a module, usually called *classifier*, that, given a new

(unclassified) history, sets its class with an acceptable degree of correctness. In our running example (see Figure 2), each patient is either class c_1 or class c_2, which can be possibly interpreted as "the patient is cured (at the end of the treatment) or not".

There are several well-known (not always comparable) classifier learning methods. They can be broadly categorized into learners based on *trees*, on *functions*, and on *rule sets*. *Decision tree learners*, such as C4.5 [34], fall into the first category, while a *logistic regressor* [20] is an example of function-based classifier learner. Methods based on rule sets, that is, *rule extraction*, are further partitioned into indirect and direct methods. When an indirect method is adopted, rules are synthesized from an already existing classifier [8, 16, 25], while when a direct method is followed, rules are directly learned [17, 41].

Let us focus on the latter methods (direct rule extraction). A classification *rule* has the form $\varphi \Rightarrow c$, where φ is a logical formula and $c \in \mathcal{C}$. Rules are not implicative formulas (we use the symbol $\Rightarrow$, instead of $\rightarrow$, to stress this fact). Direct methods include methods that pair inductive reasoning and programming languages, such as *inductive logic programming*, and randomized methods, like *evolutionary algorithms*. Direct rule extraction via evolutionary algorithms is a simple, and yet very promising, methodology. Intuitively, a set of random rules is initially produced. Then, at each iteration, the current set is evaluated and, as a result, a new (better) set of rules is built from the old one that takes into account the result of the evaluation. Evolutionary algorithms produce a new solution from the current one by means of *evolutionary operators*, that may be generic and/or specific to the problem at hand. *Multi-objective* evolutionary algorithms take into account more than one evaluation measure of the current solution. As an example, the current set of rules can be evaluated by the accuracy of the rules *and* by their succinctness—see, for instance, [12]. While different approaches based on this idea may differ in the way in which they represent rules, in the set of evolutionary operators, and in the selection strategy, the key element, common to all of them, is the evaluation of a solution, which, in the context of evolutionary algorithms, is called *fitting* function. Such a problem of evaluating the current solution can naturally be viewed as a dataset evaluation problem.

Although there is not a unique, commonly accepted definition, a fairly natural way of computing the fitting degree of a set of rules $R_1, \ldots, R_p$, where R_i is a generic rule of the form $\varphi_i \Rightarrow c_i$, for $1 \leq i \leq p$, is by suitably combining the quantities $\Sigma_{i=1}^{p} \frac{Rec(i)}{C(i)}$ (known as *accuracy*) and $\Sigma_{i=1}^{p} \frac{Rec(i)}{\Phi(i)}$ (*recall*), where $C(i)$ is the number of histories in the class c_i, $\Phi(i)$ is the number of histories that comply with φ_i, and $Rec(i)$ is the number of histories in the class c_i that comply with φ_i [33]. Both accuracy and recall clearly depend on the number of histories M_j such that $M_j, [0,1] \Vdash \varphi_i$. In our running example (Figure 2), two rules can be naturally devised:

$$\langle L \rangle (Low \wedge \langle L \rangle Head) \Rightarrow c_1;$$
$$\langle L \rangle (High \wedge (\langle A \rangle Head \vee \langle O \rangle Head)) \Rightarrow c_2.$$

To summarize, we have shown that the temporal dataset evaluation is a relevant problem that comes into play in a variety of application domains including temporal query processing, temporal constraint checking, and rule evaluation. An efficient solution to the model checking problem for interval temporal logic over finite structures turns out to be a fundamental step towards its effective treatment. In the next section we provide such a solution.

4 Model Checking

The input of classic (point-based) model checking consists of a formula and a Kripke structure generally represented by (a suitable encoding of) the set of its states, along with their labels,

n
$p : [0, 1], [0, 2], [0, 3], [1, 3], [2, 3], \ldots$
$q : [0, 1], [0, 3], \ldots$
$\ldots$

Figure 3 A succinct representation of a finite interval model for HS: the first line specifies the size of the frame (number of points in the model); the next lines encode the valuation function V.

and the set of its transitions—for the model checking of very huge structures, other kinds of solution are used, e.g., *on-the-fly* model checking. Classic model checking is infinite in nature, and infinite paths are finitely encoded in the input structure. In the setting of finite interval model checking, models can be represented simply by specifying the size of the frame (number of points in the model) and then listing, for each proposition letter, the set of intervals over which it is true (see Figure 3).

The fundamental difference between the two frameworks is thus that in the classic model checking problem frame information, that is, states and transitions, must be explicitly represented in the input, while in finite interval models frame information, that is, intervals and their relations, is implicit in the size of the temporal domain (this is because relations among intervals are induced by the underlying linear order). As a consequence, while the size of the representation of a Kripke structure is typically polynomial in the number of states and labels, the size of the representation of a finite interval model may be logarithmic in the number of intervals. As an example, consider an interval model $M = \langle [n], V \rangle$ over $\mathcal{AP} = \{p\}$, where $V(p) = \{[n-1, n]\}$. Its representation consists of the number n (which takes space $O(log(n))$) and the mapping $p \mapsto \{[n-1, n]\}$ (which takes space $O(log(n))$ as well).

We will capture situations like the above one by means of the concept of *sparse* MC *instance*. Intuitively, an instance (M, φ) of MC is sparse if it can be represented by logarithmic space (the notion will be formalized later). Now, the immediate adaptation to the interval setting of the CTL model checking algorithm by Emerson and Clarke [13] would consist, given a model checking instance (M, φ), in labeling each world, i.e., interval, of M with the set of sub-formulas of φ which are true on it. The application of such an algorithm to sparse instances immediately shows its exponential complexity. As an example, checking the formula $\langle A \rangle \langle A \rangle p$ against the model $M = \langle [n], V \rangle$, where $V(p) = \{[n-1, n]\}$, would require labeling with $\langle A \rangle p$ all intervals $[x, n-1]$, with $1 \leq x \leq n-2$, whose number is linear in n, and thus exponential in the size of the representation of M.

In the following, we describe a model checking algorithm that runs in polynomial time on every instance, thus avoiding the above problem.

The model checking algorithm. To keep the complexity under control, the algorithm we are going to describe first executes a preprocessing of a sparse instance, which may be represented in logarithmic space, and generates a non-sparse one; then, it basically applies Emerson and Clarke's solution of the model checking problem.

Hereafter, we fix an HS formula φ over the set of modalities $\{\langle A \rangle, \langle \overline{A} \rangle, \langle B \rangle, \langle \overline{B} \rangle, \langle E \rangle, \langle \overline{E} \rangle\}$ and we let $k = 4 \cdot md(\varphi)$. In addition, let $M = \langle [n], V \rangle$ be a model and let $c \in [n]$ be an element of the domain. We define a transformation $\tau_c(M) = \langle [n'], V' \rangle$, where $n' = n - 1$ and $V' : \mathbb{I}([n']) \to 2^{\mathcal{AP}}$ is defined as follows:

$$V'([w, z]) = \begin{cases} V([w, z]) & \text{if } z < c, \\ V([w, z+1]) & \text{if } w < c \leq z, \\ V([w+1, z+1]) & \text{if } w \geq c. \end{cases}$$

Given an interval model $M = \langle [n], V \rangle$ and a point $a \in [n]$, we say that a is a *useless point* if $V([a, y]) = V([x, a]) = \emptyset$ for all $x, y \in [n]$, with $x < a < y$. Moreover, we say that an interval $[a, b] \in \mathbb{I}(\mathbb{D})$, with $b - a > 2 \cdot (k + 1)^2 + 1$, is a *gap* in $M = \langle [n], V \rangle$ if each $x \in [n]$, with $a \leq x \leq b$, is a useless point in M, while $a - 1$ and $b + 1$ are not. Finally, given a gap $[a, b]$, we call the point $c = a + (k + 1)^2 + 1$ the *center* of the gap.

Given an interval model M, we denote by n_M the cardinality of its domain and by u_M the number of useless points in it. We now show that every instance with a gap can be transformed into an equivalent one, of smaller size, that has no gaps. The proof is based on the notion of bisimulation and the invariance of modal logics under bisimulations.

For every $m \in \mathbb{N}$ and each set $\mathcal{S}$ of HS modalities, an *m-bisimulation for $\mathcal{S}$* between two interval models $M_1 = \langle [n_1], V_1 \rangle$ and $M_2 = \langle [n_2], V_2 \rangle$ is a sequence $\langle Z_m, Z_{m-1}, \ldots, Z_1, Z_0 \rangle$ of binary relations between the intervals in $\mathbb{I}([n_1])$ and those in $\mathbb{I}([n_2])$, that is, $Z_i \subseteq \mathbb{I}([n_1]) \times \mathbb{I}([n_2])$ for $i \in \{0, 1, \ldots, m\}$, such that:

- $V_1([x, y]) = V_2([w, z])$, for all $i \in \{0, 1, \ldots, m\}$ and $([x, y], [w, z]) \in Z_i$ (*local condition*);
- for all $i \in \{1, \ldots, m\}$, $([x, y], [w, z]) \in Z_i$, and $X \in \mathcal{S}$:
  - if $[x, y] R_X [x', y']$, then there is $[w', z']$ such that $([x', y'], [w', z']) \in Z_{i-1}$ and $[w, z] R_X [w', z']$ (*forward condition*), and
  - if $[w, z] R_X [w', z']$, then there is $[x', y']$ such that $([x', y'], [w', z']) \in Z_{i-1}$ and $[x, y] R_X [x', y']$ (*backward condition*).

Two intervals $[x, y]$ and $[w, z]$ are *m-bisimilar under $\mathcal{S}$* if there is an m-bisimulation $\langle Z_m, Z_{m-1}, \ldots, Z_1, Z_0 \rangle$ for $\mathcal{S}$ such that $([x, y], [w, z]) \in Z_m$.

The next lemma is based on the well-known *invariance* of modal logics under bisimulations [14], which can be stated as follows: if $\mathcal{L}$ is a modal logic over a set of modalities $\mathcal{S}$ and there exists an m-bisimulation for $\mathcal{S}$ between two models M_1 and M_2 of $\mathcal{L}$, then, for every pair of m-bisimilar worlds w_1 and w_2, it holds that $M_1, w_1 \Vdash \psi$ if and only if $M_2, w_2 \Vdash \psi$, for every formula ψ of $\mathcal{L}$ with $md(\psi) \leq m$.

▶ **Lemma 1.** *If (M, φ) is an instance of* MC *featuring a gap $[a, b]$ in M, then $(M, \varphi) \equiv_{\mathsf{MC}} (\tau_c(M), \varphi)$, where c is the center of $[a, b]$.*

Proof. Let $M = \langle [n], V \rangle$ and $M' = \tau_c(M) = \langle [n'], V' \rangle$, and recall that $n' = n - 1$. Even though φ is built over the set of modalities $\{\langle A \rangle, \langle \overline{A} \rangle, \langle B \rangle, \langle \overline{B} \rangle, \langle E \rangle, \langle \overline{E} \rangle\}$, it is known from [1] that it can be translated into an equivalent formula φ' over the set of modalities $\{\langle B \rangle, \langle \overline{B} \rangle, \langle E \rangle, \langle \overline{E} \rangle\}$, with $md(\varphi')$ linear in $md(\varphi)$ (in fact, $md(\varphi') \leq k = 4 \cdot md(\varphi)$).

We provide a k-bisimulation for $\{\langle B \rangle, \langle \overline{B} \rangle, \langle E \rangle, \langle \overline{E} \rangle\}$ between M and M' such that $([0, 1], [0, 1]) \in Z_k$. The thesis immediately follows from the aforesaid invariance property.

Let $f : \{0, 1, \ldots, k\} \to \mathbb{N}$ be the mapping defined as follows: $f(i) = (1 + k - i) \cdot (k + 1)$. For each $i \in \{0, 1, \ldots, k\}$ and $([x, y], [w, z]) \in \mathbb{I}([n]) \times \mathbb{I}([n'])$, we state that $([x, y], [w, z]) \in Z_i$ if and only if one of the following conditions hold:

(1) $w = x \wedge z = y \wedge y \leq c + f(i)$, or
(2) $w = x \wedge z = y - 1 \wedge x \leq c + f(i) \wedge y \geq c - f(i) \wedge z - w > i$, or
(3) $w = x - 1 \wedge z = y - 1 \wedge x \geq c - f(i)$.

We first observe that f is monotonically decreasing as $f(i - 1) - f(i) = k + 1 > i$ for all $i \in \{1, \ldots, k\}$. Moreover, it holds that $f(0) = (k + 1)^2$ and thus, by the definitions of gap and center of a gap, we have that:

$$c - a = (k + 1)^2 + 1 > (k + 1)^2 = f(0)$$

and

$$b - c = b - a - (k+1)^2 - 1 > 2 \cdot (k+1)^2 + 1 - (k+1)^2 - 1 = (k+1)^2 = f(0).$$

Therefore, it holds that $a \leq c - f(0) - 1$ and $c + f(0) + 1 \leq b$, and thus, for each $i \in \{0, 1, \ldots, k\}$ and each d in between $c - f(i) - 1$ and $c + f(i) + 1$, that is, $c - f(i) - 1 \leq d \leq c + f(i) + 1$, d is a useless point.

We now show that $\langle Z_k, Z_{k-1}, \ldots, Z_1, Z_0 \rangle$ satisfies local, forward, and backward conditions.

Local condition. Let $([x, y], [w, z]) \in Z_i$ for some $i \in \{1, \ldots, k\}$. We have to show that $V([x, y]) = V'([w, z])$. We must deal with the three possible cases.

- Case (1). If $w = x$, $z = y$, and $y \leq c + f(i)$, then $x < c + f(i)$ and:
  - if $c \leq w < c + f(i)$, then w $(= x)$ is useless and so is $w + 1$, and thus $V'([w, z]) = V([w + 1, z + 1]) = \emptyset = V([x, y])$;
  - if $z < c$, then $V'([w, z]) = V([w, z]) = V([x, y])$;
  - if $w < c$ and $c \leq z \leq c + f(i)$, then z $(= y)$ is useless and so is $z + 1$, and thus $V'([w, z]) = V([w, z + 1]) = \emptyset = V([x, y])$.
- Case (2). If $w = x$, $z = y - 1$, $x \leq c + f(i)$, $y \geq c - f(i)$, and $z - w > i$, then:
  - if $c \leq w \leq c + f(i)$, then w $(= x)$ is useless and so is $w + 1$, and thus $V'([w, z]) = V([w + 1, z + 1]) = \emptyset = V([x, y])$;
  - if $c - f(i) - 1 \leq z < c$, then z is useless and so is y $(= z + 1)$, and thus $V'([w, z]) = V([w, z]) = \emptyset = V([x, y])$;
  - if $w < c$ and $z \geq c$, then $V'([w, z]) = V([w, z + 1]) = V([x, y])$.
- Case (3). If $w = x - 1$, $z = y - 1$, and $x \geq c - f(i)$, then $y > c - f(i)$ and:
  - if $c - f(i) \leq z < c$, then z is useless and so is y $(= z + 1)$, and thus $V'([w, z]) = V([w, z]) = \emptyset = V([x, y])$;
  - if $w \geq c$, then $V'([w, z]) = V([w + 1, z + 1]) = V([x, y])$;
  - if $z \geq c$ and $c - f(i) - 1 \leq w < c$, then w is useless and so is x $(= w + 1)$, and thus $V'([w, z]) = V([w, z + 1]) = \emptyset = V([x, y])$.

Forward condition. Let $([x, y], [w, z]) \in Z_i$, for some $i \in \{1, 2, \ldots, k\}$, and $[x, y] R_X [x', y']$, for some X. We show that there exists $[w', z']$ such that $([x', y'], [w', z']) \in Z_{i-1}$ and $[w, z] R_X [w', z']$. We proceed case by case, taking into account the value of X and the relationship that holds between $[x, y]$ and $[w, z]$:

- Case (1). If $w = x$, $z = y$, and $y \leq c + f(i)$, then $x < c + f(i)$ and:
  - if $X = B$, $X = E$, or $X = \overline{E}$, then we set $w' = x'$ and $z' = y'$; the new points x', y', w', and z' satisfy (1) with respect to $i - 1$, that is, $w' = x'$, $z' = y'$, and $y' \leq c + f(i - 1)$, meaning that $[x', y'] Z_{i-1} [w', z']$;
  - if $X = \overline{B}$, then $x' = x$ and $y' > y$; we set $w' = x'$ and, in order to set the value for z', we distinguish two cases:
    - if $y' \leq c + f(i - 1)$, then we set $z' = y'$ and thus x', y', w', and z' satisfy (1) with respect to $i - 1$;
    - if $y' > c + f(i - 1)$, then we set $z' = y' - 1$ and thus x', y', w', and z' satisfy (2) with respect to $i - 1$; in particular, to see that $z' - w' > i - 1$, observe that $z' - w' = y' - 1 - x > c + f(i - 1) - c - f(i) - 1 = k > i - 1$, and to see that $z' > z$, and thus $[w, z] R_{\overline{B}} [w', z']$, observe that $y' - 1 > c + f(i - 1) - 1 > c - f(i) \geq y$, which implies $z' = y' - 1 > y = z$.
- Case (2). If $w = x$, $z = y - 1$, $x \leq c + f(i)$, $y \geq c - f(i)$, and $z - w > i$, then:

- if $X = \overline{B}$ or $X = \overline{E}$, then we set $w' = x'$ and $z' = y' - 1$, and thus x', y', w', and z' satisfy (2) with respect to $i - 1$;
- if $X = B$, then $x' = x$ and $y' < y$; we set $w' = x'$ and, in order to set the value for z', we distinguish the following cases:
  - if $y' \geq c - f(i-1)$ and $y' - x' > i$, then we set $z' = y' - 1$, and thus x', y', w', and z' satisfy (2) with respect to $i-1$; in particular, it holds that $z' - w' = y' - 1 - x' \geq c - f(i-1) - 1 - c - f(i) = k > i - 1$;
  - if $y' < c - f(i-1)$, then we set $z' = y'$, and thus x', y', w', and z' satisfy (1) with respect to $i - 1$; in particular, to see that $z' < z$, and thus $[w, z]R_B[w', z']$, observe that $z' = y' < c - f(i-1) < c - f(i) - 1 \leq y - 1 = z$;
  - if $y' - x' \leq i$, then we set $z' = y'$, and thus x', y', w', and z' satisfy (1) with respect to $i - 1$; in particular, it holds that $y' \leq x' + i \leq c + f(i) < c + f(i-1)$.

- Case (3). If $w = x - 1$, $z = y - 1$, and $x \geq c - f(i)$, then $y > c - f(i)$ and:
  - if $X = B$, $X = \overline{B}$, or $X = E$, then we set $w' = x' - 1$ and $z' = y' - 1$; the new points x', y', w', and z' satisfy (3) with respect to $i - 1$;
  - if $X = \overline{E}$, then $x' < x$ and $y' = y$; we set $z' = z = y' - 1$ and, in order to set the value for w', we distinguish two cases:
    - if $x' \geq c - f(i-1)$, then we set $w' = x' - 1$, and thus x', y', w', and z' satisfy (3) with respect to $i - 1$;
    - if $x' < c - f(i-1)$, then we set $w' = x'$, and thus x', y', w', and z' satisfy (2) with respect to $i - 1$; in particular, to see that $z' - w' > i - 1$, observe that $z' - w' = y - 1 - x' > c - f(i) - 1 - c + f(i-1) = k > i - 1$, and to see that $w' < w$, and thus $[w, z]R_{\overline{E}}[w', z']$, observe that $w' = x' < c - f(i-1) < c - f(i) - 1 \leq x - 1 = w$.

The backward condition can be proved in a very similar way.

Therefore, the sequence $\langle Z_k, Z_{k-1}, \ldots, Z_1, Z_0 \rangle$ is a k-bisimulation for $\{\langle B \rangle, \langle \overline{B} \rangle, \langle E \rangle, \langle \overline{E} \rangle\}$ between M and M'. In addition, we have that $([0,1],[0,1]) \in Z_k$, and since φ is equivalent to φ' and $md(\varphi') \leq k$, we have that:

$$M, [0,1] \Vdash \varphi \Leftrightarrow M, [0,1] \Vdash \varphi' \Leftrightarrow M', [0,1] \Vdash \varphi' \Leftrightarrow M', [0,1] \Vdash \varphi,$$

and thus the thesis. ◀

We are now ready to formalize the notion of sparse MC instance. We say that an instance $\mathcal{I} = (M, \varphi)$ of MC is *sparse* if

$$u_M > \frac{2 \cdot (k+1)^2 + 2}{2 \cdot (k+1)^2 + 3} \cdot n_M + 1.$$

By making use of Lemma 1, it is possible to transform a sparse instance of MC into an equivalent non-sparse one, as formally stated by the following lemma.

▶ **Lemma 2.** *For every sparse instance of* MC*, there exists an equivalent non-sparse one that is computable in deterministic polynomial time.*

Proof. As a preliminary step, we show that if $\mathcal{I} = (M, \varphi)$ is a sparse instance of MC, then it features a gap $[a, b]$ in M. Assume, towards a contradiction, that $\mathcal{I}$ features no gap and consider the partition of $[n_M]$ into intervals $[(i-1) \cdot \big(2 \cdot (k+1)^2 + 3\big) + 1, i \cdot \big(2 \cdot (k+1)^2 + 3\big)]$, with $1 \leq i \leq \lfloor \frac{n_M}{2 \cdot (k+1)^2 + 3} \rfloor$, plus the interval $[\Big(\lfloor \frac{n_M}{2 \cdot (k+1)^2 + 3} \rfloor\Big) \cdot \big(2 \cdot (k+1)^2 + 3\big) + 1, n_M]$. Since the length of each interval $[(i-1) \cdot \big(2 \cdot (k+1)^2 + 3\big) + 1, i \cdot \big(2 \cdot (k+1)^2 + 3\big)]$, with

Algorithm 1 Transforming a sparse instance into an equivalent non-sparse one.

```
1: function De-Sparsify(M, φ)
2:     while (M = ⟨[n], V⟩, φ) is sparse do
3:         let [a, b] be a gap in M and let b' = a + 2 · (k + 1)^2 + 1
4:         for each p ∈ AP do V'(p) ← ∅
5:         for each p ∈ AP and [w, z] ∈ V(p) do
6:             if z < a then
7:                 V'(p) ← V'(p) ∪ {[w, z]}
8:             else if w < a and z > b then
9:                 V'(p) ← V'(p) ∪ {[w, z − (b − b')]}
10:            else if w > b then
11:                V'(p) ← V'(p) ∪ {[w − (b − b'), z − (b − b')]}
12:        M ← ⟨[n − (b − b')], V'⟩
13:    return (M, φ)
```

$1 \leq i \leq \lfloor \frac{n_M}{2\cdot(k+1)^2+3} \rfloor$, is equal to $2 \cdot (k+1)^2 + 2$, any such interval must contain a non-useless point, otherwise there would be a gap in M. It immediately follows that there are at least $\lfloor \frac{n_M}{2\cdot(k+1)^2+3} \rfloor$ non-useless points in M, and thus it holds that:

$$u_M \leq n_M - \left\lfloor \frac{n_M}{2 \cdot (k+1)^2 + 3} \right\rfloor \leq n_M - \frac{n_M}{2 \cdot (k+1)^2 + 3} + 1 = \frac{2 \cdot (k+1)^2 + 2}{2 \cdot (k+1)^2 + 3} \cdot n_M + 1,$$

which is in contradiction with $\mathcal{I}$ being sparse.

Algorithm 1 computes a non-sparse MC instance that is equivalent to the one given in input. To this end, it iteratively applies a suitable transformation $\tau_{[a,b]}$ to each gap $[a, b]$, until a non-sparse MC instance is obtained.

Given an interval model M and a gap $[a, b]$ in it, the transformation $\tau_{[a,b]}(M)$ returns the pair $\langle [n'], V' \rangle$, where $n' = n - (b - a - 2 \cdot (k+1)^2 - 1)$ and $V' : \mathbb{I}([n']) \to 2^{\mathcal{AP}}$ is such that $V'([w, z])$ is equal to:

$$\begin{cases} V([w, z]) & \text{if } z < a \\ V([w, z + (b - a - 2 \cdot (k+1)^2 - 1)]) & \text{if } w < a \leq z \\ V([w + (b - a - 2 \cdot (k+1)^2 - 1), z + (b - a - 2 \cdot (k+1)^2 - 1)]) & \text{if } w \geq a. \end{cases}$$

It can be easily checked that the model $\tau_{[a,b]}(M)$ returned by one application of $\tau_{[a,b]}$ is equivalent to the one returned by $(b - a - 2 \cdot (k+1)^2 - 1)$ applications of the transformation τ_c, where c is the center of the gap $[a, b]$. Any such application of τ_c produces a model where (the current configuration of the interval) $[a, b]$ is shrunk into the interval $[a, b']$, where $b' = b - 1$ (and thus $b' - a = b - a - 1$). Hence, after $(b - a - 2 \cdot (k+1)^2 - 1) - 1$ applications of τ_c, $[a, b]$ is reduced to the interval $[a, b']$, with $b' = b - (b - a - 2 \cdot (k+1)^2 - 2)$ and $b' - a = b - b + a + 2 \cdot (k+1)^2 + 2 - a = 2 \cdot (k+1)^2 + 2 > 2 \cdot (k+1)^2 + 1$, meaning that $[a, b']$ is still a gap in the resulting model. By Lemma 1, the model returned by an application of τ_c is equivalent to the input model, and thus we have that $(M, \varphi) \equiv_{\mathsf{MC}} (\tau_{[a,b]}(M), \varphi)$. It is worth pointing out that executing $(b - a - 2 \cdot (k+1)^2 - 1)$ times the transformation τ_c, instead of executing $\tau_{[a,b]}$ only once, would result in an algorithm whose execution time is exponential when the instance features exponentially large gaps.

Termination of the algorithm is guaranteed by the fact that $\tau_{[a,b]}$, applied to M, produces a model M' with a reduced number of gaps: a gap $[a,b]$ in M is shrunk into an interval $[a,b']$, which is not a gap in M', as $b'-a = 2\cdot(k+1)^2+1$.

As for the computational complexity, it is not difficult to check that the algorithm runs in polynomial time. Let N be the size of the input. The number of gaps is bounded by $n_M - u_M + 1$ ($n_M - u_M$ is the number of non-useless points—observe that there is at least one non-useless point between any two gaps), thus implying that the body of the outermost loop (line 2) is executed at most N times. Both innermost loops (lines 4 and 5) are clearly executed at most N times as well, giving an overall time complexity of $O(N^2)$. ◀

A non-sparse instance (M,φ) can be represented in space polynomial in n_M. To see that, it suffices to show that $n_M \leq p(N)$, for a polynomial p, where N is the size of the representation of (M,φ). First, we observe that

$$n_M = u_M + e_M$$

where e_M is the number of non-useless points, that is, those points that occur *explicitly* in the model. Since non-useless points are explicitly represented in the model, it clearly holds that $e_M \leq N$. By definition of (non-)sparse instance, we have that (recall that $k \leq 4\cdot md(\varphi) \leq 4\cdot N$):

$$\begin{aligned}
& u_M \leq \tfrac{2\cdot(k+1)^2+2}{2\cdot(k+1)^2+3}\cdot(u_M+e_M)+1 \Leftrightarrow \\
\Leftrightarrow\ & u_M - \tfrac{2\cdot(k+1)^2+2}{2\cdot(k+1)^2+3}\cdot u_M \leq \tfrac{2\cdot(k+1)^2+2}{2\cdot(k+1)^2+3}\cdot e_M + 1 \Leftrightarrow \\
\Leftrightarrow\ & \tfrac{1}{2\cdot(k+1)^2+3}\cdot u_M \leq \tfrac{2\cdot(k+1)^2+2}{2\cdot(k+1)^2+3}\cdot e_M + 1 \Leftrightarrow \\
\Leftrightarrow\ & u_M \leq (2\cdot(k+1)^2+2)\cdot e_M + (2\cdot(k+1)^2+3) \leq \\
& \qquad \leq (2\cdot(4\cdot N+1)^2+2)\cdot N + (2\cdot(4\cdot N+1)^2+3) = \\
& \qquad = 32\cdot N^3 + 48\cdot N^2 + 20\cdot N + 5,
\end{aligned}$$

which means that

$$n_M = u_M + e_M \leq 32\cdot N^3 + 48\cdot N^2 + 21\cdot N + 5.$$

Therefore the number $|\mathbb{I}([n_M])|$ of intervals in M is also bounded by a polynomial in N ($O(N^6)$), thus making it possible to adapt Emerson and Clarke's algorithm to obtain a polynomial model checking algorithm for non-sparse instances.

Algorithm 2 implements such an adaptation. Let us assume φ to be represented as a binary tree and M to be represented as in Figure 3. Moreover, for each sub-formula ψ of φ, let $L(\psi)$ be the set of all intervals of M where ψ holds. For every node of the tree representing φ (corresponding to a sub-formula ψ of φ), the algorithm computes the corresponding set of intervals $L(\psi)$. Initially, we set $L(\psi) = \emptyset$, for each sub-formula ψ of φ which is not a proposition letter, and we set $L(p) = V(p)$ for each proposition letter p. Modalities $\langle\overline{A}\rangle$, $\langle\overline{B}\rangle$, and $\langle\overline{E}\rangle$ are not dealt with by the algorithm as they are specular to the other ones.

▶ **Lemma 3.** *If $\mathcal{I} = (M,\varphi)$ is a non-sparse instance of* MC*, then Algorithm 2 returns* **true** *if and only if $M,[0,1] \Vdash \varphi$. Moreover, it runs in polynomial time.*

Proof. It is immediate to see that Algorithm 2 is sound and complete. In order to show that it runs in polynomial time, we proceed as follows. Let N be the size of the representation of

Algorithm 2 Checking a non-sparse model.

```
1:  function CHECK(M, φ)
2:      for each ψ sub-formula of φ do
3:          if ψ = p then
4:              L(ψ) = V(p)
5:          else
6:              L(ψ) = ∅
7:      for each ψ sub-formula of φ (ordered by increasing size) do
8:          if ψ = ¬τ then
9:              L(ψ) = 𝕀([n]) \ L(τ)
10:         else if ψ = τ ∨ ξ then
11:             L(ψ) = L(τ) ∪ L(ξ)
12:         else if ψ = ⟨A⟩τ then
13:             for [x, y] ∈ L(τ) and for z < x do
14:                 L(ψ) = L(ψ) ∪ {[z, x]}
15:         else if ψ = ⟨B⟩τ then
16:             for [x, y] ∈ L(τ) and for z > y do
17:                 L(ψ) = L(ψ) ∪ {[x, z]}
18:         else if ψ = ⟨E⟩τ then
19:             for [x, y] ∈ L(τ) and for z < x do
20:                 L(ψ) = L(ψ) ∪ {[z, y]}
21:     if [0, 1] ∈ L(φ) then
22:         return True
23:     else
24:         return False
```

the input $\mathcal{I}$. Since $\mathcal{I}$ is non-sparse, the number of intervals in M is polynomial in N, thus providing a polynomial upper bound to the cardinality of $L(\psi)$, for each sub-formula ψ of φ.

The body of the loop at line 7 is executed at most N times (as $|\varphi| \leq N$). Whenever ψ is a Boolean formula, computing $L(\psi)$ takes a linear time in the number $|\mathbb{I}([n_M])|$ of intervals in M. The remaining cases can be efficiently implemented (in $O(N^6)$) by using a symbolic representation. As an example, in order to store the set of intervals on which $\langle A\rangle\tau$ holds, knowing that τ holds on an interval $[x, y]$, it suffices to store the number x, with the intended meaning that it represents all intervals ending at x. By suitably adapting the representation of $L(\psi)$, one is able to guarantee the complexity of these cases to be at most linear in the number of intervals as well. This allows us to conclude that Algorithm 2 runs in $O(N^7)$ time, and thus it is deterministic polynomial. ◀

▶ **Theorem 4.** *The finite interval model checking problem can be solved by a deterministic algorithm that runs in polynomial time in the size of the input.*

To be understood in perspective, such a result must be compared to other classic model checking problems [36]. While model checking of CTL formulas is quadratic (therefore, more efficient than ours), model checking of LTL as well as of CTL* formulas is PSPACE-complete (therefore, much less efficient than ours). Moreover, model checking of HS formulas over Kripke structures goes from coNP-complete to non-elementary, depending on the particular fragment of HS under consideration [26].

5 Conclusions

In this paper, we formally defined the problem of temporal dataset evaluation, and we highlighted the role that finite interval temporal model checking plays in it. We also showed how the problem of temporal dataset evaluation has several applications that range from temporal query answering to temporal constraint checking and rule evaluation, the last one being a key element in various machine learning processes. We identified the finite interval model checking problem for the interval temporal logic HS as the main problem to be solved in this perspective, and we devised an efficient (deterministic polynomial) algorithm for it.

We are currently working on an implementation of the developed model checking procedure, which uses symbolic techniques to obtain better performances, and we plan to integrate such a procedure in an existing module for (temporal) rule extracion [17], based on an evolutionary algorithm, to compute a suitable *fitting function* of a set of temporal rules.

References

1 L. Aceto, D. Della Monica, A. Ingólfsdóttir, A. Montanari, and Guido Sciavicco. On the expressiveness of the interval logic of Allen's relations over finite and discrete linear orders. In *Proc. of the 14th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 8761 of *LNAI*, pages 267–281, 2014.

2 J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

3 M. H. Böhlen, J. Gamper, and C. S. Jensen. How would you like to aggregate your temporal data? In *Proc. of the 13th International Symposium on Temporal Representation and Reasoning (TIME), 15-17 June 2006, Budapest, Hungary*, pages 121–136. IEEE Computer Society, 2006.

4 A. Bottrighi, L. Giordano, G. Molino, S. Montani, P. Terenziani, and M. Torchio. Adopting model checking techniques for clinical guidelines verification. *Artificial Intelligence in Medicine*, 48(1):1–19, 2010.

5 L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. Interval temporal logic model checking: The border between good and bad HS fragments. In *Proc. of the 8th International Joint Conference on Automated Reasoning (IJCAR)*, volume 9706 of *LNCS*, pages 389–405. Springer, 2016.

6 L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. Interval vs. point temporal logic model checking: an expressiveness comparison. In *Proc. of the 36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 65 of *LIPIcs*, pages 26:1–26:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

7 L. Bozzelli, A. Molinari, A. Montanari, A. Peron, and P. Sala. Model checking the logic of allen's relations meets and started-by is p^{np}-complete. In *Proc. of the 7th International Symposium on Games, Automata, Logics and Formal Verification (GandALF)*, volume 226 of *EPTCS*, pages 76–90, 2016.

8 A. Chaves, M. Vellasco, and R. Tanscheit. Fuzzy rules extraction from support vector machines for multi-class classification. *Neural Computing and Applications*, 22(7-8):1571–1580, 2013.

9 E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2002.

10 C. Combi and A. Montanari. Data models with multiple temporal dimensions: Completing the picture. In *Proc. of the 13th International Conference on Advanced Information Systems Engineering (CAiSE)*, volume 2068 of *LNCS*, pages 187–202. Springer, 2001.

11 C. Combi and P. Sala. Mining approximate interval-based temporal dependencies. *Acta Informatica*, 53(6-8):547–585, 2016.

12 K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, London, UK, 2001.

13 E.A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, volume B: formal models and semantics, pages 995–1072. Elsevier MIT Press, 1990.

14 V. Goranko and M. Otto. Model theory of modal logic. In P. Blackburn, J. van Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, pages 249–329. Elsevier, 2007.

15 J.Y. Halpern and Y. Shoham. A propositional modal logic of time intervals. *Journal of the ACM*, 38(4):935–962, 1991.

16 Y. Hayashi, S. Nakano, and S. Fujisawa. Use of the recursive-rule extraction algorithm with continuous attributes to improve diagnostic accuracy in thyroid disease. *Informatics in Medicine Unlocked*, 1:1–8, 2015.

17 F. Jiménez, G. Sánchez, and J. M. Juárez. Multi-objective evolutionary algorithms for fuzzy classification in survival prediction. *Artificial Intelligence in Medicine*, 60(3):197–219, 2014.

18 V. Khatri, S. Ram, R.T. Snodgrass, and P. Terenziani. Capturing telic/atelic temporal data semantics: Generalizing conventional conceptual models. *IEEE Trans. Knowl. Data Eng*, 26(3):528–548, 2014.

19 K. Kulkarni and J.E. Michels. Temporal features in SQL:2011. *ACM SIGMOD Record*, 41(3):34–43, 2012.

20 S. le Cessie and J.C. van Houwelingen. Ridge estimators in logistic regression. *Applied Statistics*, 41(1):191–201, 1992.

21 L. Liu and M. T. Özsu, editors. *Encyclopedia of Database Systems*. Springer NY, 2nd edition, 2017.

22 A. Lomuscio and J. Michaliszyn. An epistemic halpern-shoham logic. In *Proc. of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1010–1016, 2013.

23 A. Lomuscio and J. Michaliszyn. Decidability of model checking multi-agent systems against a class of EHS specifications. In *Proc. of the 21st European Conference on Artificial Intelligence (ECAI)*, pages 543–548, 2014.

24 A. Lomuscio and J. Michaliszyn. Model checking multi-agent systems against epistemic HS specifications with regular expressions. In *Proc. of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 298–308. AAAI Press, 2016.

25 M. Mashayekhi and R. Gras. Rule extraction from random forest: the RF+HC methods. In *Proc. of the 28th Canadian Conference on Advances in Artificial Intelligence*, pages 223–237, 2015.

26 A. Molinari, A. Montanari, A. Murano, G. Perelli, and A. Peron. Checking interval properties of computations. *Acta Informatica*, 53(6-8):587–619, 2016.

27 A. Molinari, A. Montanari, and A. Peron. Complexity of ITL model checking: Some well-behaved fragments of the interval logic HS. In *Proc. of the 22nd International Symposium on Temporal Representation and Reasoning (TIME)*, pages 90–100. IEEE Computer Society, 2015.

28 A. Molinari, A. Montanari, and A. Peron. A model checking procedure for interval temporal logics based on track representatives. In *Proc. of the 24th EACSL Annual Conference on Computer Science Logic (CSL)*, volume 41 of *LIPIcs*, pages 193–210, 2015.

29 A. Molinari, A. Montanari, A. Peron, and P. Sala. Model checking well-behaved fragments of HS: the (almost) final picture. In *Proc. of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 473–483. AAAI Press, 2016.

30 A. Montanari, A. Murano, G. Perelli, and A. Peron. Checking interval properties of computations. In *Proc. of the 21st International Symposium on Temporal Representation and Reasoning (TIME)*, pages 59–68. IEEE Computer Society, 2014.

31 A. Pnueli. The temporal logic of programs. In *Proc. of the 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 46–57, 1977.

32 A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13(1):45 – 60, 1981.

33 D. M. W. Powers. Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

34 J. R. Quinlan. *C45: Programs for Machine Learning*. Morgan Kaufmann, 1992.

35 E. Quintarelli. *Model-Checking Based Data Retrieval, An Application to Semistructured and Temporal Data*, volume 2917 of *LNCS*. Springer, 2004.

36 P. Schnoebelen. The complexity of temporal logic model checking. In *Proc. of the 4th Conference on Advances in Modal Logic*, pages 393–436, 2002.

37 R.T. Snodgrass, editor. *The TSQL2 Temporal Query Language*. Kluwer Academic Publishers, 1995.

38 R.T. Snodgrass, I. Ahn, G. Ariav, D.S. Batory, J. Clifford, C.E. Dyreson, R. Elmasri, F. Grandi, C.S. Jensen, W. Käfer, N. Kline, K. Kulkarni, T.Y.C. Leung, N. Lorentzos, J.F. Roddick, A. Segev, M.D. Soo, and S.M. Sripada. TSQL2 language specification. *SIGMOD Record*, 23(1):65–86, 1994.

39 A. U. Tansel, J. Clifford, S. K. Gadia, S. Jajodia, A. Segev, and R. T. Snodgrass, editors. *Temporal Databases: Theory, Design, and Implementation*. Benjamin/Cummings, 1993.

40 M. Y. Vardi. Model checking for database theoreticians. In *Proc. of the 10th International Conference on Database Theory (ICDT)*, volume 3363 of *LNCS*, pages 1–16. Springer, 2005.

41 I.H. Witten, E. Frank, and M.A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., 3rd edition, 2011.