Implementation of a tableau-based satisfiability checker for HS3 *

Emilio Muñoz-Velasco[1], Guido Sciavicco[2], and Ionel Eduard Stan[2]

[1] Department of Applied Mathematics
University of Málaga (Spain) (ejmunoz@uma.es)
[2] Department of Mathematics and Computer Science
University of Ferrara (Italy)
({guido.sciavicco@|ioneleduard.stan@student.}unife.it)

Abstract. Although there exist several decidable fragments of Halpern and Shoham's interval temporal logic HS, the computational complexity of their satisfiability problem tend to be generally high. Recently, the fragment HS3 of HS, based on coarser-than-Allen's relations, has been introduced, and it has been proven to be not only decidable, but also relatively efficient. In this paper we describe an implementation of a tableau-based satisfiability checker for HS3 interpreted in the class of all finite linear orders.

1 Introduction

Interval Temporal Logics (ITLs) consider time intervals as the primitive ontological entities. This represents an advantage when dealing with some relevant application domains, such as planning and synthesis of controllers, which are characterized by advanced features that are neglected or dealt with in an unsatisfactory way by point-based formalisms. ITLs have been applied in several fields, such as hardware and real-time system verification, language processing, constraint satisfaction and planning, among others [2, 15, 25, 27]. Moreover, due to the fact that temporal logics are considered as the natural basis for temporal extensions of Description Logics [5], several attempts have been made to design interval-based extensions of such formalisms, as in [3, 4, 7, 28]. ITLs can be also considered as the temporal counterpart of TSQL, that is, the temporal extension to the language SQL for databases, included in the standard SQL:2011 [32]. Halpern and Shoham's Modal Logic of Allen's Relations (HS), introduced in [20], is the most prominent representative interval temporal logic, but its satisfiability problem is undecidable when interpreted in almost every interesting class of linearly ordered sets. Various strategies have been considered in the literature to define fragments or variants of HS with a better computational behaviour, including constraining the underlying temporal structure [24], restricting the set of

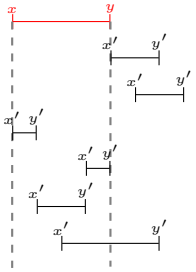
* The authors acknowledge the support from the Italian GNCS Project *"Logics and Automata for Interval Model Checking"* (G. Sciavicco), and the Spanish Project TIN15-70266-C2-P-1 (E. Muñoz-Velasco).

modal operators [1,13], softening the semantics to a reflexive one [23], restricting the nesting of modal operators [10], and restricting the propositional power of the languages [12,14]. The underlying idea in [26] is substituting Allen's Interval Algebra (IA) [2] as the backbone of HS (modal operators in the HS repository can be mapped one-by-one over Allen's interval relations) with a set of jointly exhaustive, mutually exclusive, but coarser, interval relations, originally proposed in Golumbic and Shamir's work [17]. In particular, the coarser algebra IA3 involves three relations: the original *before* and *after*, plus a relation (*intersects*) that can be viewed as the disjunction of all the remaining ones (and therefore is the inverse of itself and includes equality). We call the corresponding modal logic HS3: in [26] its finite satisfiability problem has been shown to be PSPACE-complete by providing a suitable small model theorem.

In this paper we describe the implementation of a satisfiability checker for HS3 interpreted in finite linear orders. The finite satisfiability problem is usually emblematic for the entire range of (discrete) satisfiability problems in interval temporal logics, and so is our tableau-based procedure. On the other hand, checking the (finite) satisfiability of an interval temporal logic formula is essentially different from the same problem for a point-based temporal formula due to the absence of the *Until* operator and the fact that ITLs are not generally susceptible of being treated via fixed-point techniques. This means that most of the work done for (variants of) LTL, including [18, 29, 31] cannot be simply reused, nor compared with this one. Tableau-based procedures for interval temporal logics are not common in the literature. Among the few exceptions, a procedure for the fragment A of HS has been implemented in [9, 22]; the former is an experimental implementation not devoted to computational efficiency, and the latter is an attempt to use an automatic tableaux generator introduced in [30]. The only previous attempt to apply a generic theorem prover to an interval temporal logic can be found in [11], where a tableau-based decision procedure for the fragment D, interpreted over dense linear orders, was developed in LoTREC [16]. Finally, in [8] the authors designed an implementation for a tableau-based procedure for a ITL with the *chop* operator, which is interval-based but employs a semantic strategy, called *locality*, that reduces the truth of a propositional letter over an interval to that of the initial point of that interval.

2 The logic HS and its fragment HS3

Let $\mathbb{D} = \langle D, < \rangle$ be a strict (i.e., irreflexive) linearly ordered set. A *strict interval* (resp., *non-strict interval*) over $\mathbb{D}$ is an ordered pair $[x, y]$, where $x, y \in D$ and $x < y$ (resp., $x \leq y$). In the recent literature, the *strict semantics*, where only strict intervals are considered, is usually adopted. This conforms to the definition of interval adopted by Allen in [2], but differs from the one given by Halpern and Shoham in [20]. If we exclude the identity relation, there are 12 different relations between two intervals 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 Fig. 1, and their inverses, that is,

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

HS3/HS7	Semantics
$\langle AO \rangle$	$\langle AO \rangle \equiv \langle A \rangle \vee \langle O \rangle$
$\langle DBE \rangle$	$\langle DBE \rangle \equiv \langle D \rangle \vee \langle B \rangle \vee \langle E \rangle$
$\langle I \rangle$	$\langle I \rangle \equiv \langle AO \rangle \vee \langle \overline{AO} \rangle \vee \langle DBE \rangle \vee \langle \overline{DBE} \rangle$

Fig. 1. Allen's interval relations, HS modalities, and HS3/HS7 modalities.

$R_{\overline{X}} = (R_X)^{-1}$, for each $X \in \{A, L, B, E, D, O\}$. We interpret interval structures as Kripke structures, with Allen's relations playing the role of the accessibility relations. Thus, we associate a universal modality $[X]$ and an existential modality $\langle X \rangle$ with each Allen relation R_X. For each $X \in \{A, L, B, E, D, O\}$, the *transposes* of the modalities $[X]$ and $\langle X \rangle$ are the modalities $[\overline{X}]$ and $\langle \overline{X} \rangle$, corresponding to the inverse relation $R_{\overline{X}}$ of R_X. Halpern and Shoham's logic HS [20] is a multi-modal logic with formulas built from a finite, non-empty set $\mathcal{AP}$ of atomic propositions (also referred to as proposition letters), the classical propositional connectives, and a pair of modalities for each Allen relation:

$$\varphi ::= \bot \mid p \mid \neg\psi \mid \psi \vee \xi \mid \psi \wedge \xi \mid \langle X \rangle \psi \mid \langle \overline{X} \rangle \psi, \tag{1}$$

where $p \in \mathcal{AP}$ and $X \in \{A, L, B, E, D, O\}$. The other propositional connectives and constants (e.g., $\rightarrow$, and $\top$), as well as the dual modalities (e.g., $[A]\varphi \equiv \neg\langle A \rangle \neg\varphi$), can be derived in the standard way. In general, given any subset $S \subseteq \{Y, \overline{Y} \mid Y \in \{A, L, B, E, D, O\}\}$, one can define the relation:

$$R_S = \bigvee_{X \in S} R_X.$$

The corresponding modal operator can be denoted by simply juxtaposing the original symbols to obtain a string, so that, for example, the modal operator that is the disjunction of Allen's relations *overlaps* and *during* would be denoted by $\langle OD \rangle$. In some cases, such as the relation *intersect*, we introduce a shorthand

for the sake of readability, so that $I = A\overline{A}B\overline{B}E\overline{E}O\overline{O}D\overline{D}$[3]. Well-formed HS3 formulae can be obtained from (1) when $X \in \{L, I\}$; for the sake of completeness, a finer version of HS3 can be defined, called HS7, under the restriction that $X \in \{L, AO, DBE\}$, but in [26] it has been proved that its computational behaviour is the same as the entire HS.

The semantics of HS and HS3 is given in terms of *interval models* $M = \langle \mathbb{I}(\mathbb{D}), V \rangle$, where $\mathbb{D}$ is a linear order, $\mathbb{I}(\mathbb{D})$ is the set of all (strict) intervals over $\mathbb{D}$, and V is a *valuation function* $V : \mathcal{AP} \mapsto 2^{\mathbb{I}(\mathbb{D})}$, which assigns to each atomic proposition $p \in \mathcal{AP}$ the set of intervals $V(p)$ on which p holds. The *truth* of a formula φ on a given interval $[x, y]$ in an interval model M is defined by structural induction on formulae, as follows:

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

Fig. 1 describes the semantics of HS3 operators in terms of that of HS operators. Notice that a distinguishing characteristic of HS, inherited by the fragments considered in this paper, is the fact that the truth of a propositional letter over a given interval has no influence on the truth of the same propositional letter on the intervals contained in it, nor its points. Alternative choices include the *locality* principle, that implies assigning the same truth value to a propositional letter over an interval as over its starting point (see [25] for the introduction of locality in ITLs, and more recent work, such as [21], for an example of a renewed interest in constraining principles). This and other, similar, model-theoretic constraints have been shown useful in several applications and help reduce the complexity of problems such as satisfiability or model-checking, but, here, we follow the most general approach, in which undecidability is the rule and decidability the exception. It is worth observing that HS3 retains sufficient expressive power to define, for example, the *global* operator, and this depends essentially from its modalities being jointly exhaustive:

$$[U]\varphi = \varphi \wedge \bigwedge_{X \in \{L, I\}} ([X]\varphi \wedge [\overline{X}]\varphi).$$

Formulas of HS, and therefore of HS3, can be interpreted over several different classes of interval models. Their frame properties sometimes influence the computational complexity of the satisfiability problem, as witnessed by the recent series of results [1, 13]. Notable classes of linear orders include the class of *all* linear orders, the class of all *finite* linear orders, containing all and only those

[3] This notation should not be confused with the standard notation for fragments of HS, indicated by the set of its modal operators, e.g., $\mathsf{AB\overline{BA}}$, which includes four modal operators, namely, $\langle A \rangle$, $\langle \overline{A} \rangle$, $\langle B \rangle$, and $\langle \overline{B} \rangle$.

linear orders with finitely many points, and the classes of interval models that can be built over notable sets such as $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$, and $\mathbb{R}$. Beside notable exceptions (such as the fragment $\mathsf{ABB\overline{A}}$), fragments of HS tend to behave in a similar way in all finite/discrete cases. Not only is solving in an efficient way the finite satisfiability problem a necessary step towards tackling the satisfiability problem for other, sometimes more interesting, classes of linearly ordered sets, but it is also an important problem on its own. For example, temporal databases use intervals to describe time, and the interval logic HS is their ideal logical counterpart, especially when interpreted in finite domains. Temporal queries, as well as temporal constraints, can be easily expressed in HS, and the problem of establishing whether a query or a constraint is *semantically correct* is, essentially, a finite satisfiability problem. The latter is undecidable in HS; although HS3 is less expressive than HS, some interesting queries and constraints can be expressed in the former [26], and, thus, checked.

3 A tableau-based satisfiability checker for HS3

In [26], a small model theorem for HS3 interpreted in the class of all finite linear orders has been proved, that is, that a formula φ is finitely satisfiable if and only if it has a model with less than

$$2^{|\varphi|\cdot(\log(4\cdot|\varphi|+1)+\log(|\varphi|)+|\varphi|)}$$

distinct points; let us call this number $\mathbb{L}(\varphi)$. Building on it, it has been possible to prove that the finite satisfiability problem for HS3 is PSPACE-complete. This result, in particular, is obtained by showing the correctness and completeness of a PSPACE (non-deterministic) algorithm based on the maximal dimension of a satisfying model for a given formula φ. Such an algorithm is inefficient in nature (although theoretically optimal); in this section, we describe an efficient (but not theoretically optimal) imperative, deterministic implementation of a tableau-based procedure for checking the finite satisfiability of formulas of HS3; the flow diagram of the entire procedure is depicted in Fig. 2. We have chosen to develop a *semantic tableau-based* satisfiability checker for HS3. Algebraically speaking, both the tableau and the formula to be checked are represented as rooted decorated trees. A *rooted directed tree* is a graph $G = (V, E, r)$, where V is nonempty, $E \subseteq V \times V$, $|E| = |V| - 1$, and $r \in V$ is its *root*; every element of V is called a *node*. A *rooted decorated tree* [19] is a rooted directed tree such that there exists a function that associates every node with its *decoration*, which can be thought of as the information carried by that node; when we represent formulae, a decoration is a propositional letter or an operator, and when we represent semantic tableaux, a decoration is the collection of all information needed to expand the tableau or to close it. In any (rooted decorated) tree, nodes without successors are called *leaves*, and every finite path from the root to a leaf is called a *branch*.

Representation. Formulas and tableaux are represented as rooted decorated trees. Focusing on formulas, which correspond to *binary* trees, each node of

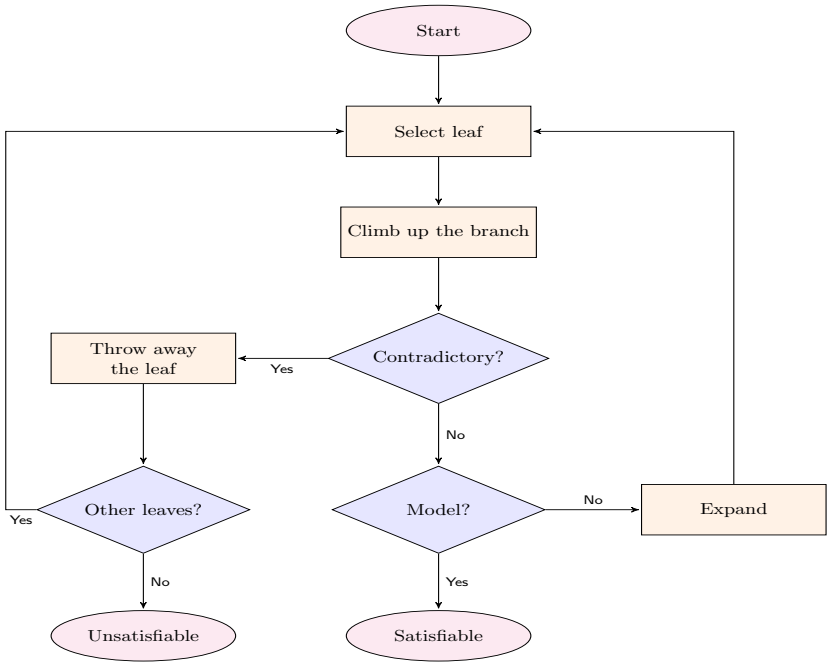


Fig. 2. Flow diagram of the tableau

the tree contains a code that identifies the operator (Boolean or modal, distinguishing between universal and existential), of which a propositional letter is a special case; nodes with a single child have a left child only. A formula is read and contextually transformed by a tree; a simple recursive procedure eliminates all implications and pushes all negations in front of propositional letters, obtaining an equivalent formula in negated normal form. Moreover, since finite satisfiability can be reduced to finite *initial* satisfiability, that is, satisfiability over the initial interval $[0, 1]$, before checking its satisfiability our procedure transforms a formula φ into the formula

$$\varphi \vee \langle L \rangle \varphi \vee \langle I \rangle \varphi,$$

whose initial satisfiability is checked. Indeed, if φ is satisfied on a model M at some interval $[x, y] \neq [0, 1]$, then either $y > 1$ and therefore $\langle L \rangle \varphi$ is satisfied at $[0, 1]$, or $y \leq 1$, and therefore $\langle I \rangle \varphi$ is satisfied at $[0, 1]$.

A tableau is represented as a *k-ary* tree in form of left-child right-sibling. Each node of this tree contains a pointer to the node in the formula tree that represents the sub-formula under analysis, the interval over which it holds, an active/inactive flag, a leaf/internal flag, and the pointers to the left child, the right sibling, and the parent. Domains are represented as totally ordered sets of *floating point* numbers, so that an interval is a pair of floating point numbers. This has a very specific purpose: whenever a new point must be added in be-

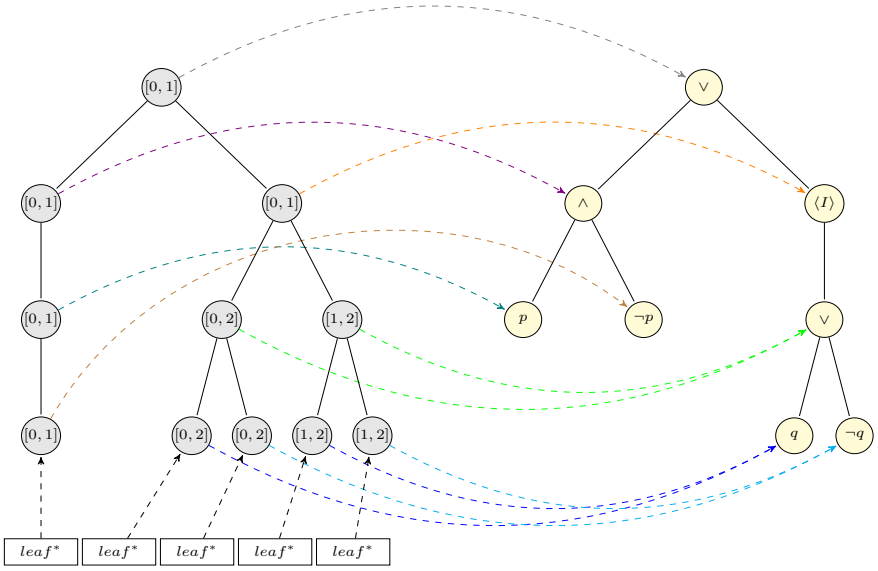


Fig. 3. Interaction between the tableau (lhs) and the tree formula (rhs).

tween a pair of already existing ones, it can be created by simply computing their arithmetic average; nevertheless, a model is always finite by construction. Some tableau nodes are leaves during the construction of the tableau; they (temporally) represent their branch, so that they also store domain information, plus other data that help us choosing the next branch depending on the expansion policy. In addition to the two trees (a formula tree and a tableau tree - the former is fixed during the satisfiability checking process of a given formula, the latter evolves), there are additional data structures used within the tableau expansion. In particular, the collection of all current leaves is stored in a linked list. Each element of such a list points to the tableau node that represents that leaf (and therefore a branch) which may be chosen in the next expansion step. A non-trivial adaptation of a generic tree visit algorithm has been implemented in order to correctly identify, given a tableau node, the set of all and only leaves that belong to the sub-tree rooted at it (see Fig. 3). Finally, a dynamic data structure is built (and destroyed) before each expansion step that allows us to examine the branch on which the to-be-expanded tableau node lies in order to establish if the branch is closed (because it is contradictory), or it represents a model (in which case the procedure stops and returns that the given formula is satisfiable). Such a structure may be thought of as a hash (unordered) set with an efficient (constant time) lookup method, that contains the intervals and the formulas that are (currently) true on them.

Initial tableau and main procedure. The *initial tableau* for a formula φ whose initial satisfiability must be checked is a tableau tree composed by a single

$$\frac{\varphi_1 \vee \varphi_2, [x,y], \mathbb{D}}{\varphi_1, [x,y], \mathbb{D} \mid \varphi_2, [x,y], \mathbb{D}}\ (\mathsf{OR}) \qquad \frac{\varphi_1 \wedge \varphi_2, [x,y], \mathbb{D}}{\begin{array}{c}\varphi_1, [x,y], \mathbb{D} \\ \varphi_2, [x,y], \mathbb{D}\end{array}}\ (\mathsf{AND})$$

$$\frac{\langle X\rangle\varphi, [x,y], \mathbb{D}}{\varphi, \mu_{1,I}^{e,X}([x,y],\mathbb{D}), \mu_{1,D}^{e,X}([x,y],\mathbb{D}) \mid \ldots \mid \varphi, \mu_{\mu^{e,X}([x,y],\mathbb{D}),I}^{e,X}([x,y],\mathbb{D}), \mu_{\mu^{e,X}([x,y],\mathbb{D}),D}^{e,X}([x,y],\mathbb{D})}\ (\mathsf{EXIST})$$

$$\frac{[X]\varphi, [x,y], \mathbb{D}}{\begin{array}{c}\varphi, \mu_1^{u,X}([x,y],\mathbb{D}), \mathbb{D} \\ \varphi, \mu_2^{u,X}([x,y],\mathbb{D}), \mathbb{D} \\ \cdots \\ \varphi, \mu_{\mu^{u,X}([x,y],\mathbb{D})}^{u,X}([x,y],\mathbb{D}), \mathbb{D}\end{array}}\ (\mathsf{UNIV})$$

Table 1. Expansion rules.

tableau node with the following decoration: its formula node pointer points to the root of the formula tree that represents φ, its interval is $[0,1]$, its active flag and its leaf flag are both to 1, its domain is $\{0 < 1\}$, and all other pointers are null. Given a leaf of the current tableau (that is, the branch represented by it), the following two operations are performed: *(i)* branch closing checking and branch model checking, and *(ii)* choosing the next to-be-expanded tableau node. Assuming that the current branch B is not closed and is not a model, the next to-be-expanded tableau node is chosen according to the following policy: it is the active node on B that is closest to the root.

Branch checking. Given a branch B, we check, at the same time, whether it is closed or it is a model, and, during this operation, a structure (H, S, n) is produced; both H (i.e., the current labeled interval structure) and S (i.e., the set of all formulas that should appear somewhere due to some universal modality) are unordered heaps of pairs (node, interval), while n is a node on the branch B. By means of this structure we are able to check if: *(i)* it presents a contradiction, or *(ii)* it is a model. The branch B is *closed* if one of the following two conditions holds: a propositional contradiction is found on it, that is, there exist two nodes in B such that their decorations show $(p, [x,y])$ and $(\neg p, [x,y])$, respectively, or the domain in the decoration of its leaf is greater than $\mathbb{L}(\varphi)$. Notice that when B is contradictory, it may be the case that it presents more than one contradiction. Let $(m_1, m_1'), (m_2, m_2'), \ldots$ be the set of all pairs of contradictory nodes in B in increasing order of distance (i.e., number of edges) from the root; the structure (H, S, n) is built in such a way that n points precisely to m_1'. In this way, we can eliminate from the list of leaves all those that identify a branch B' that share the contradiction (m_1, m_1') with B (there may be more than one such branch). If all leaves are eliminated, the formula is found unsatisfiable. If B is not contradictory, we check whether all active universal modalities on B have already been expanded in all possible intervals: if that is the case, and if, in B, the only active nodes are universal modalities, then we can conclude that B is a model. If B is not a model but it is not contradictory, then n points to the node that is closest to the root and active.

Branch expansion. Expansion rules are described in Tab. 1. Boolean rules are standard, while the rules for modal operators are designed as follows. Let $\mathfrak{D}$ be the set of all finite domains, and let $\mathfrak{I}$ be the set of all intervals in any domain in $\mathfrak{D}$. For a given existential operator $\langle X \rangle$, we define a function:

$$\mu^{e,X} : \mathfrak{I} \times \mathfrak{D} \to \mathbb{N}$$

that for a given pair $([x, y], \mathbb{D})$ returns the number of different intervals in the relation R_X with $[x, y]$ plus the number of new intervals that should be created in the relation R_X in order to explore all qualitatively distinct possibilities (see [19] for a similar approach for a simpler interval temporal logic); for example, $\mu^{e,L}([0, 1], \{0 < 1 < 2\})$ is 5: indeed, if $\langle L \rangle \psi$ holds on $[0, 1]$ and the current domain is $\{0 < 1 < 2\}$, then ψ may hold on some interval $[1.25, 1.75], [1.5, 2], [2, 2.5], [3, 4]$, or $[1.5, 2.5]$; notice that, for example, $[1.25, 1.75]$ is a necessary possibility as it represents a new interval completely between existing points; the same holds for $[3, 4]$. In this way, given $\langle X \rangle \psi$ holding on $[x, y]$ in the finite domain $\mathbb{D}$, the parametric function(s):

$$\mu^{e,X}_{i,I} : \mathfrak{I} \times \mathfrak{D} \to \mathfrak{I}\,,\ \mu^{e,X}_{i,D} : \mathfrak{I} \times \mathfrak{D} \to \mathfrak{D}$$

return, respectively, the i-th interval on which ψ holds and the corresponding i-th domain (not necessarily different from $\mathbb{D}$) in no particular order; following up with the above example, $\mu^{e,L}_{1,I}([0, 1], \{0 < 1 < 2\}) = [1.25, 1.75]$, $\mu^{e,L}_{1,D}([0, 1], \{0 < 1 < 2\}) = \{0 < 1 < 1.25 < 1.75 < 2\}$. Because we are restricting ourselves to initial satisfiability, these functions never return new points between 0 and 1, nor smaller than 0. In this way for each existential operator $\langle X \rangle \psi$ we have an existential disjunctive rule that creates enough branches to search for every possible location for ψ. Dually, for universal operators, we have functions $\mu^{u,X}$ and $\mu^{u,X}_i$ to return all intervals seen from $[x, y]$ via R_X; in this case, domains do not change.

Given the node in B to be expanded, the correct rule is chosen for its expansion, and the result of such a step is applied to *all leaves in the sub-tree* rooted at the chosen node. Potentially, each application of the expansion rules gives rise to new nodes to be attached to such leaves; they are placed all the same level as new leaves in disjunctive rules (i.e., $\vee, \langle X \rangle$), or in sequence (with no particular order) in conjunctive rules (i.e., $\wedge, [X]$). While the application of Boolean rules is completely standard, non-Boolean ones are subject to the following conditions: *(i)* if the $\langle X \rangle$ rule is applied to the node n with the decoration $(\langle X \rangle \psi, [x, y])$, and B *already* contains a pair $(\psi, [z, t])$ where $[x, y] R_X [z, t]$, then n is deactivated without any expansion; *(ii)* similarly, if the $[X]$ rule is applied to the node n with the decoration $([X]\psi, [x, y])$, then, for each pair $(\psi, [z, t])$ where $[x, y] R_X [z, t]$ is already present in B, this pair is not added to the set of nodes that is the result of the expansion. The protocol for active/inactive nodes is designed in such a way that nodes are deactivated after being expanded; in case of universal nodes, they are copied at the end of the branch with the active flag at 1.

Soundness, completeness. We want to argue that our tableau-based procedure is sound and complete; calculating its complexity in the worst-case scenario is not really informative (it is, in fact, doubly exponential in time) considering that the interest in tableau-based methods roots in their efficiency in the average case, as well as their several possible optimizations.

In order to argue that the presented method is complete we need to introduce the following notion. Consider a node n on a tableau for a formula φ, and let $S(n)$ be the set of all decorations on nodes between n and the root; we say that $S(n)$ is *satisfied on an extension of* $\mathbb{D}(n)$, where $\mathbb{D}(n)$ is the domain in the decoration of n, if there exists a model M based on some extension of $\mathbb{D}(n)$ such that, for each $(\psi, [x, y]) \in S(n)$ it is the case that $M, [x, y] \Vdash \psi$. We now show that for every finitely satisfiable formula of HS3 the presented method terminates and returns 'Satisfiable', that is, contra-positively, whenever the procedure closes all branches, the starting formula φ is not finitely satisfiable, by proving, by induction, a stronger claim: for any node n at height h on a tableau for φ, if every branch that contains n is closed, then $S(n)$ is not satisfied on any extension $\mathbb{D}'$ of $\mathbb{D}(n)$ such that $|\mathbb{D}'| \leq \mathbb{L}(\varphi)$; notice that, when n is the root, this is to say that φ is not finitely satisfiable. Now, if $h = 0$, the only branch that contains n also contains two nodes with decorations $(p, [x, y])$ and $(\neg p, [x, y])$, and therefore $S(n)$ is simply not satisfiable, or the number of points ever named in the decorations of its nodes is more than $\mathbb{L}(\varphi)$, for which $S(n)$ can never be satisfied on any extension of $\mathbb{D}(n)$. If $h > 0$, then n has been expanded by some rule, some nodes $n_1, n_2, \ldots$ exist that are descendants of n, and the inductive hypothesis applies to all of them. If the rule that has been applied is Boolean, than the claim follows immediately. If it is the universal rule, then suppose that $([X]\psi, [x, y])$ is in the decoration of n. Every branch that contains n also contains all nodes that are the result of its expansion, and, in particular, some node n' with decoration $(\psi, [z, t])$ for some interval $[z, t]$ such that $[x, y]R_X[z, t]$; if $S(n)$ were satisfiable on some extension of $\mathbb{D}(n)$, then, in particular, $S(n) \cup \{(\psi, [z, t])\} = S(n')$ would be too, but this is in contradiction with the inductive hypothesis. Finally, if it is the existential rule, then suppose that $(\langle X\rangle\psi, [x, y])$ is in the decoration of n. If $S(n)$ were satisfiable on some extension of $\mathbb{D}(n)$, then there would be a model whose domain extends $\mathbb{D}(n)$ such that it satisfies $\langle X\rangle\psi$ on $[x, y]$ and ψ on some $[z, t]$ such that $[x, y]R_X[z, t]$. By construction, there must be some successor n' of n that contains the decoration $(\psi, [z, t])$, independently of z, t being already in $\mathbb{D}(n)$. This means that $S(n')$ would be satisfiable on some extension of $\mathbb{D}(n')$, which is in contradiction with the inductive hypothesis.

To conclude, we must argue that our method is also sound, that is, for every formula φ of HS3 for which it returns 'Satisfiable', there exists a model M such that $M, [0, 1] \Vdash \varphi$. Consider a branch B such that it is not contradictory, all its active nodes are universal, and every node with universal decoration has been already expanded on every possible interval of the domain $\mathbb{D}$ of the branch. Now, let M be a model based on $\mathbb{D}$, and whose valuation function is defined as follows: for each interval $[x, y]$ and each propositional letter p, $[x, y] \in V(p)$ if and only if $(p, [x, y])$ decorates some node on B. We want to prove, by structural induction,

that, for each node n in B with decoration $(\psi, [x, y])$, $M, [x, y] \Vdash \psi$. If ψ is a propositional letter or its negation, we have the result immediately. If ψ is a composite formula, two cases arise: either it is a universal formula, or it is not. In the latter case, the fact that B is not closed implies that n has been expanded, and such expansion has been applied to all branches that contain n: if ψ is a conjunction, then both conjuncts have been included as decorations in nodes of B, if it is a disjunction then at least one disjunct has been included as decoration in some node of B, and, if it is $\psi = \langle X \rangle \xi$, then at least one node in B must be decorated with $(\xi, [z, t])$, for some $[z, t]$ such that $[x, y] R_X [z, t]$; in all cases, the inductive hypothesis applies, so that M must satisfy ψ on $[x, y]$. In the former case, if $\psi = [X]\xi$, since B cannot be further extended, it must be the case that a node n' with decoration $(\xi, [z, t])$ occurs in B for each $[z, t]$ such that $z, t \in \mathbb{D}$ and that $[x, y] R_X [z, t]$, and again, the inductive hypothesis applies.

Theorem 1. *A formula φ of HS3 is finitely satisfiable if and only if the tableau-based method described in Fig. 2, with the rules in Tab. 1, returns 'Satisfiable'.*

Policies. Our procedure is programmed fully object-oriented in C++ standard language with *threads* capabilities. Threads are run in (virtual) parallel, and carry a specific policy for choosing the next leaf to be examined. Each policy is *fair*, that is every branch is eventually examined; the advantage of using different policies is the improved execution time, especially for satisfiable formulae. We have taken into account two key aspects: domain cardinality and branch sparseness. The *sparseness* degree allows us to estimate how many intervals already existing in the domain are actually used; to compute such an estimation, we calculate the average of positive propositional letters assigned to some interval, and defined the sparseness of the branch as the variance of the (ideal) binomial probabilistic variable associated to assigning positive propositions to intervals. We implemented the following policies: *(i)* branches with smaller domain and less sparse first (SBF); *(ii)* branches with longer domain and more sparse first (LBF); *(iii)* branches taken in First-In-First-Out order (FIFO) - that is, the tableau tree is explored depth-first. In our experiments (see next section) we used all such policies; establishing which one of them, if any, is clearly better than the others is an open problem.

4 Experimental results

Because tableau-based (and, in general, decision and semi-decision) procedures for interval temporal logics are not common in the literature, there are no available benchmarks. We have designed a scalable experiment to generate a sequence of (arbitrary) finitely satisfiable formulae, shown in Tab. 2, which are systematically generated for $k = 1, 3, 5, \ldots$, so that their length can be put in relation with the time that our method takes to establish its satisfiability; formulae are generated inductively, and to generate φ_k, we use $\varphi_{k-2}[+2]$, that is, φ_{k-2} where each propositional letter p_i has been replaced by p_{i+2}. We have followed the general guidelines for generating a systematic benchmark for modal logics [6] taking

Finitely satisfiable formulas	
k	formula
φ_1	$p_0 \wedge [I]\neg p_2 \wedge [\overline{L}]\neg p_2 \wedge \langle L\rangle p_1 \wedge [L](p_1 \rightarrow ([I]\neg p_3 \wedge [L]\neg p_3 \wedge \langle \overline{L}\rangle p_2))$
φ_3	$p_0 \wedge [I]\neg p_2 \wedge [\overline{L}]\neg p_2 \wedge \langle L\rangle p_1 \wedge [L](p_1 \rightarrow ([I]\neg p_3 \wedge [L]\neg p_3 \wedge \langle \overline{L}\rangle \varphi_1[+2]))$
φ_5	$p_0 \wedge [I]\neg p_2 \wedge [\overline{L}]\neg p_2 \wedge \langle L\rangle p_1 \wedge [L](p_1 \rightarrow ([I]\neg p_3 \wedge [L]\neg p_3 \wedge \langle \overline{L}\rangle \varphi_3[+2]))$
...	...

Table 2. A benchmark for (finitely) satisfiable formulae of HS3.

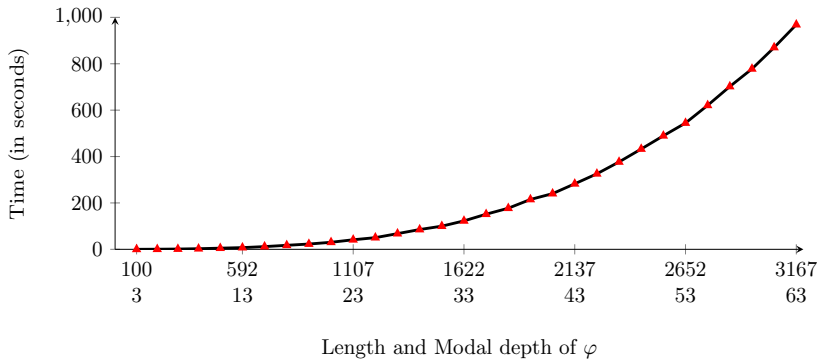


Fig. 4. Elapsed time for satisfiable formulae (by length and by modal depth).

into account, in particular: *(i)* length of the formulas, in terms of the number of symbols, and *(ii)* modal depth of the formulas. The results of this experiment (carried out on an Intel(R) Core(TM) i7-6700HQ, with a clock of 2.60Ghz, four cores, and 16GB RAM), are shown in Fig. 4; as it can be observed, satisfiable formulae have been successfully checked up to 3167 symbols (and modal depth of 63) in less than 17 minutes.

In order to generate a similar benchmark for unsatisfiable formulae, we used a rather straightforward method: we considered an arbitrary set of propositional and modal tautologies (of HS3), and we systematically applied universal substitution to generate longer and longer tautologies; we then tested their negation for satisfiability. It turns out that the elapsed time for testing a satisfiable formula grows proportionally to the length and the modal depth of formulae, while for unsatisfiable ones they differ; therefore there is a single plot in the first case, and two different plots in the second one. The apparent erratic behaviour for unsatisfiable formulas is probably due to the absence of specific optimization policies for this case: therefore, the time needed to establish that a formula is not satisfiable depends too much on the depth at which a contradiction is found.

5 Conclusions

In this paper we have described an efficient implementation of a tableau-based reasoner for the interval temporal logic HS3, whose finite satisfiability problem

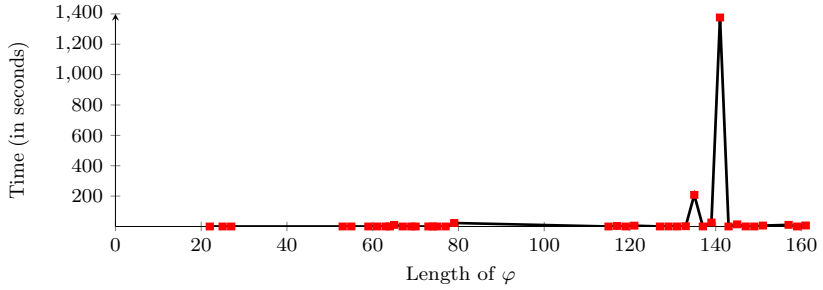


Fig. 5. Elapsed time for finitely unsatisfiable formulae (by length).

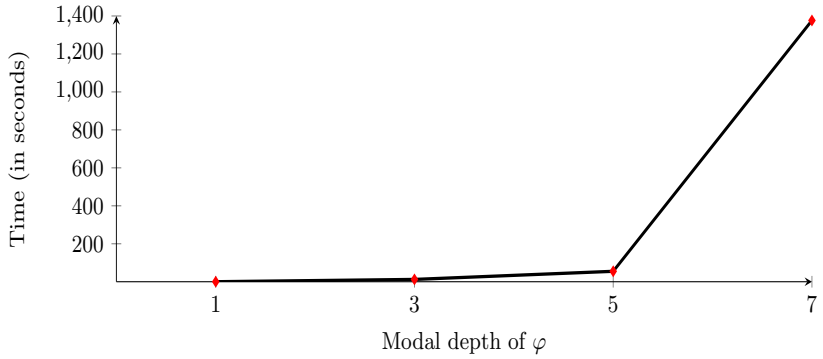


Fig. 6. Elapsed time for finitely unsatisfiable formulae (by modal depth).

had been shown to be PSPACE-complete in [26]. The experiments showed that we are able to check the satisfiability of relatively long formulas with a relatively high modal depth. There are several considerations that can be drawn from our experiments. Tableau-based satisfiability checkers are notoriously not very efficient with formulas with a very relevant propositional component. In the case of unsatisfiable formulas, branch pruning is the main strategy to improve the performance of the reasoner. Our implementation allows for experimenting with rules for branch pruning thanks to the efficient representation of the branch information; since the underlying problem is PSPACE, we expect our implementation to be very susceptible to optimizations. On the other hand, for satisfiable formulas, the main problem relies in branch selection: being able to choose the most promising branch is one of the most important optimizations tools for a tableau-based procedure. As far as branch selection is concerned, our policies can be seen as a first step in this direction, but, as future work, we plan to experiment in innovative and much more aggressive strategies for branch selection. In particular, we are looking into describing this problem as a machine learning problem, and designing an intelligent system that is able to quickly select

the most promising branch based on previous experience on the same problem, improving in this way the performance on satisfiable formulas.

References

1. L. Aceto, D. Della Monica, V. Goranko, A. Ingólfsdóttir, A. Montanari, and G. Sciavicco. A complete classification of the expressiveness of interval logics of Allen's relations: the general and the dense cases. *Acta Informatica*, 53(3):207–246, 2016.
2. J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.
3. A. Artale, D. Bresolin, A. Montanari, V. Ryzhikov, and G. Sciavicco. DL-lite and interval temporal logics: A marriage proposal. In *Proc. of the 21st European Conference of Artificial Intelligence (ECAI)*, pages 957–958, 2014.
4. A. Artale and E. Franconi. A temporal Description Logic for reasoning about actions and plans. *Journal of Artificial Intelligence Reasoning*, 9:463–506, 1998.
5. A. Artale, V. Ryzhikov, R. Kontchakov, and M. Zakharyaschev. A cookbook for temporal conceptual data modelling with Description Logics. *ACM Transaction on Computational Logic*, 15(3):1–50, 2014.
6. P. Balsiger, A. Heuerding, and S. Schwendimann. A benchmark method for the propositional modal logics K, KT, S4. *Journal of Automated Reasoning*, 24(3):297–317, 2000.
7. C. Bettini. Time-dependent concepts: Representation and reasoning using temporal description logics. *Data Knowedge Engeneering*, 22(1):1–38, 1997.
8. H. Bowman and S. J. Thompson. A tableau method for interval temporal logic with projection. In *Proc. of the Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX)*, volume 1397 of *LNCS*, pages 108–123, 1998.
9. D. Bresolin, D. Della Monica, A. Montanari, and G. Sciavicco. A tableau system for Right Propositional Neighborhood Logic over finite linear orders: an implementation. In *Proc. of the 22th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX)*, volume 8123 of *LNCS*, pages 74–80, 2013.
10. D. Bresolin, D. Della Monica, A. Montanari, and G. Sciavicco. The light side of interval temporal logic: the Bernays-Schönfinkel fragment of CDT. *Annals of Mathematics and Artificial Intelligence*, 71(1-3):11–39, 2014.
11. D. Bresolin, V. Goranko, A. Montanari, and P. Sala. Tableaux for logics of subinterval structures over dense orderings. *Journal of Logic and Computation*, 20(1):133–166, 2010.
12. D. Bresolin, A. Kurucz, E. Muñoz-Velasco, V. Ryzhikov, G. Sciavicco, and M. Zakharyaschev. Horn fragments of the Halpern-Shoham interval temporal logic. *ACM Transactions on Computational Logic (TOCL)*, Accepted, 2017.
13. D. Bresolin, D. Della Monica, A. Montanari, P. Sala, and G. Sciavicco. Interval temporal logics over strongly discrete linear orders: Expressiveness and complexity. *Theoretical Computer Science*, 560:269–291, 2014.
14. D. Bresolin, E. Muñoz-Velasco, and G. Sciavicco. Sub-propositional fragments of the interval temporal logic of Allen's relations. In *Proc. of the 14th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 8761 of *LNCS*, pages 122–136, 2014.
15. Z. Chaochen and M. R. Hansen. *Duration Calculus: A Formal Approach to Real-Time Systems*. EATCS: Monographs in Theoretical Computer Science. Springer, 2004.

16. L. Fariñas del Cerro, D. Fauthoux, O. Gasquet, A. Herzig, D. Longin, and F. Massacci. Lotrec : The generic tableau prover for modal and description logics. In *Proc. of the 1st International Joint Conference on Automated Reasoning (IJCAR)*, pages 453–458, 2001.
17. M.C. Golumbic and R. Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *Journal of the ACM*, 40(5):1108–1133, 1993.
18. V. Goranko, A. Kyrilov, and D. Shkatov. Tableau tool for testing satisfiability in LTL: Implementation and experimental analysis. *Electr. Notes Theor. Comput. Sci.*, 262:113–125, 2010.
19. V. Goranko, A. Montanari, and G. Sciavicco. Propositional interval neighborhood temporal logics. *Journal of Universal Computer Science*, 9(9):1137–1167, 2003.
20. J. Halpern and Y. Shoham. A propositional modal logic of time intervals. *Journal of the ACM*, 38(4):935–962, 1991.
21. A. Molinari, A. Montanari, A. Murano, G. Perelli, and A. Peron. Checking interval properties of computations. *Acta Informatica*, 53(6-8):587–619, 2016.
22. D. Della Monica, A. Montanari, G. Sciavicco, and D. Tishkovsky. First steps towards automated synthesis of tableau systems for interval temporal logics. In *Proc. of the 5th International Conference on Computational Logics, Algebras, Programming, Tools, and Benchmarking (COMPUTATION TOOLS)*, pages 32–37, 2014.
23. A. Montanari, I. Pratt-Hartmann, and P. Sala. Decidability of the logics of the reflexive sub-interval and super-interval relations over finite linear orders. In *Proc. of the 17th International Symposium on Temporal Representation and Reasoning (TIME)*, pages 27–34. IEEE Computer Society, 2010.
24. A. Montanari, G. Sciavicco, and N. Vitacolonna. Decidability of interval temporal logics over split-frames via granularity. In *Proc. of the 8th European Conference on Logics in Artificial Intelligence (JELIA)*, volume 2424 of *LNAI*, pages 259–270. Springer, 2002.
25. B. Moszkowski. *Reasoning about digital circuits*. PhD thesis, Dept. of Computer Science, Stanford University, Stanford, CA, 1983.
26. E. Muñoz-Velasco, M. Pelegrín-García, P. Sala, and G. Sciavicco. On coarser interval temporal logics and their satisfiability problem. In *Proc. of the 16th Conference of the Spanish Association for Artificial Intelligence (CAEPIA 2015)*, volume 9422 of *LNAI*, pages 1–11, 2015.
27. I. Pratt-Hartmann. Temporal prepositions and their logic. *Artificial Intelligence*, 166(1–2):1–36, 2005.
28. A. Schmiedel. Temporal terminological logic. In *Proc. of the 8th National Conference on Artificial Intelligence (AAAI)*, pages 640–645. AAAI Press, 1990.
29. S. Schwendimann. A new one-pass tableau calculus for PLTL. In *Proc. of the 4th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*, pages 277–291. Springer, 1998.
30. D. Tishkovsky, R.A. Schmidt, and M. Khodadadi. The tableau prover generator MetTeL^2. In *Proc. of the 13th European Conference on Logics in Artificial Intelligence (JELIA)*, pages 492–495, 2012.
31. M.Y. Vardi and P. Wolper. Automata theoretic techniques for modal logics of programs (extended abstract). In *Proc. of the 16th Annual ACM Symposium on Theory of Computing (STOC 1984)*, pages 446–456, 1984.
32. F. Zemke. What's new in SQL:2011. *SIGMOD Record*, 41(1):67–73, 2012.