An optimal tableau for Right Propositional Neighborhood Logic over trees

Davide Bresolin
Department of Computer Science
University of Verona, Verona, Italy
davide.bresolin@univr.it

Angelo Montanari, Pietro Sala
Department of Mathematics and Computer Science
University of Udine, Udine, Italy
{angelo.montanari|pietro.sala}@dimi.uniud.it

Abstract

Propositional interval temporal logics come into play in many areas of artificial intelligence and computer science. Unfortunately, most of them turned out to be (highly) undecidable. Some positive exceptions, belonging to the classes of neighborhood logics and of logics of subinterval relations, have been recently identified. In this paper, we address the decision problem for the future fragment of Propositional Neighborhood Logic (Right Propositional Neighborhood Logic) interpreted over trees and we positively solve it by providing a tableau-based decision procedure that works in exponential space. Moreover, we prove that the decision problem for the logic is EXPSPACE-hard, thus showing the optimality of the proposed procedure.

1 Introduction

Propositional interval temporal logics play a significant role in computer science, as they provide a natural framework for representing and reasoning about temporal properties in many areas of artificial intelligence, theoretical computer science, and databases. Various propositional interval temporal logics have been proposed in the literature [9]. The most significant ones are Halpern and Shoham's Modal Logic of Time Intervals (HS) [11], Venema's CDT logic, interpreted over linear and partial orderings [10, 17], Moszkowski's Propositional Interval Temporal Logic [14], the propositional interval logics of temporal neighborhood [3, 4, 5, 6, 8], and the temporal logics of subinterval relations [1, 2, 16]. Unfortunately, many of them turned out to be (highly) undecidable.

In the recent years, a number of decidable interval logics have been identified and systematically studied. One can get decidability by making a suitable choice of the interval modalities. This is the case with the $\langle B \rangle \langle \overline{B} \rangle$ (*begins/begun by*) and $\langle E \rangle \langle \overline{E} \rangle$ (*ends/ended by*) fragments of HS [9]. As an alternative, decidability can be achieved by constraining the classes of temporal structures over which the logic is interpreted. This is the case with the so-called Split Logics (SLs), which are interpreted over structures where every interval can be 'chopped' in at most one way [13]. Another possibility is to constrain the relation between the truth value of a formula over an interval and its truth values over subintervals of that interval [14, 17]. As an example, one can constrain a propositional letter to be true over an interval if, and only if, it is true at its starting point (*locality*) or it is true over all its subintervals (*homogeneity*). All these solutions impose suitable syntactic and/or semantic restrictions that make it possible to reduce the logics to point-based ones [12].

A major challenge is thus to look for expressive enough interval temporal logics, which cannot be reduced to point-based ones, but are decidable. Some positive examples have been found in the class of propositional neighborhood logics (PNL for short) [8]. These logics feature modalities for right and left interval neighborhood, namely, the *meets* operator $\langle A \rangle$ and its transpose *met-by* $\langle \overline{A} \rangle$. In [6], Bresolin et al. prove the decidability of the future fragment of PNL (RPNL for short) over the natural numbers. They first show that an RPNL formula is satisfiable if, and only if, there exist a finite model or an ultimately periodic (infinite) one with a finite representation of *bounded size*, and then they develop an optimal tableau-based decision procedure (NEXPTIME). Such a result has been later extended to full PNL over the integers [5]. The NEXPTIME decidability of PNL over the class of all linear orderings, as well as over other more specific classes, has been proved in [3], by reducing it to the decision problem for the 2-variable fragment of first-order logic extended with a linear ordering [15]. Finally, a branching-time neighborhood logic that interleaves CTL operators with RPNL ones has been developed in [4].

In this paper, we address the decision problem for RPNL over trees and we positively solve it by providing a tableau-based decision procedure. The paper is organized as follows. In Section 2, we introduce syntax and semantics of

1530-1311/08 $25.00 © 2008 IEEE
DOI 10.1109/TIME.2008.17

RPNL interpreted over trees, pointing out the problems one must face when linear structures are replaced with trees. In Section 3, we present a tableau-based decision procedure and we prove its soundness, completeness, and optimality. Conclusions provide an assessment of the work and outline future research directions.

2 RPNL syntax and semantics

In this section we introduce RPNL and we show how to interpret it over branching structures (trees), where every time point may have many successor time points. We assume every path to be either finite or isomorphic to $\langle \mathbb{N}, < \rangle$ and we allow any node to have infinitely many (possibly, uncountably many) successors[1].

A directed graph is a pair $\mathbb{G} = \langle G, S \rangle$, where G is a set of nodes and $S \subseteq G \times G$ is a binary relation over them, called successor relation. A *finite S-sequence* over $\mathbb{G}$ is a sequence of nodes $g_1 g_2 \dots g_n$, with $n \geq 2$, such that $S(g_i, g_{i+1})$ for $i = 1, \dots, n-1$. *Infinite S-sequences* can be defined analogously. A *path* ρ in $\mathbb{G}$ is a finite or infinite S-sequence. In the following, we shall take advantage of a relation $S^+ \subseteq G \times G$ such that $S^+(g_i, g_j)$ if, and only if, g_i and g_j are respectively the first and the last element of a finite S-sequence. In such a case, we say that g_j is S-reachable from g_i or, equivalently, that g_i is an ancestor of g_j. Trees can be either finite or infinite graphs. They are formally defined as follows.

Definition 1. *A* tree *is a directed graph $\mathbb{T} = \langle T, S \rangle$. The elements of T are called* time points*. T contains a distinguished time point t_0, called the* root *of the tree. The relation S is such that:*

- *there exists no t' such that $S(t', t_0)$, that is, the root has no S-predecessors;*
- *for every $t \in T$, if $t \neq t_0$, then $S^+(t_0, t)$, that is, every time point $t \neq t_0$ is S-reachable from the root;*
- *for every $t \in T$, if $t \neq t_0$, then there exists exactly one $t' \in T$ such that $S(t', t)$, that is, every time point $t \neq t_0$ has exactly one S-predecessor.*

Given a tree $\mathbb{T} = \langle T, S \rangle$, we can define a strict partial ordering $<$ over T such that, for every $t, t' \in T$, $t < t'$ if, and only if, $S^+(t, t')$. It can be easily shown that, for every infinite path ρ in $\mathbb{T}$, $\langle \rho, < \rangle$ is isomorphic to $\langle \mathbb{N}, < \rangle$. Given a tree $\mathbb{T} = \langle T, S \rangle$ and the corresponding strict partial ordering $\langle T, < \rangle$, an *interval* is an ordered pair $[t_i, t_j]$ such that $t_i, t_j \in T$ and $t_i < t_j$ (point-intervals $[t, t]$ are thus excluded). We denote the set of all intervals by $\mathbb{I}(\mathbb{T})$. The pair $\langle \mathbb{T}, \mathbb{I}(\mathbb{T}) \rangle$ is called an *interval structure*. For every pair of intervals $[t_i, t_j], [t'_i, t'_j] \in \mathbb{I}(\mathbb{T})$, we say that $[t'_i, t'_j]$ is a *right* (resp., *left*) *neighbor* of $[t_i, t_j]$ if, and only if, $t_j = t'_i$ (resp., $t'_j = t_i$). The vocabulary of *Right Propositional Neighborhood Logic* [6] (RPNL for short) consists of a set AP of propositional letters, the Boolean connectives $\neg$ and $\vee$, and the modal operator $\langle A \rangle$. *Formulae* of RPNL, denoted by $\varphi, \psi, \dots$, are recursively defined by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \langle A \rangle \varphi.$$

The other Boolean connectives, the logical constants $\top$ (true) and $\bot$ (false), and the dual modal operator $[A]$ are defined as usual. We denote by $|\varphi|$ the length of φ, that is, the number of symbols in φ (as a matter of fact, we shall use $|\ |$ to denote the cardinality of a set as well). Whenever there are no ambiguities, we call an RPNL formula just a formula. Formulae of the forms $\langle A \rangle \psi$ or $[A]\psi$ are called *temporal formulae* (from now on, we identify $\neg\langle A \rangle \psi$ with $[A]\neg\psi$ and $\neg[A]\psi$ with $\langle A \rangle \neg\psi$); formulae of the form $\langle A \rangle \psi$ are called *temporal requests*.

A *model* for an RPNL formula is a pair $\mathbf{M} = \langle \langle \mathbb{T}, \mathbb{I}(\mathbb{T}) \rangle, \mathcal{V} \rangle$, where $\langle \mathbb{T}, \mathbb{I}(\mathbb{T}) \rangle$ is an interval structure and $\mathcal{V} : \mathbb{I}(\mathbb{T}) \rightarrow 2^{AP}$ is a *valuation function* assigning to every interval the set of propositional letters true on it. Given a model $\mathbf{M} = \langle \langle \mathbb{T}, \mathbb{I}(\mathbb{T}) \rangle, \mathcal{V} \rangle$ and an interval $[d_i, d_j] \in \mathbb{I}(\mathbb{T})$, the semantics of RPNL is recursively defined by the *satisfaction relation* $\Vdash$ as follows:

- for every $p \in AP$, $\mathbf{M}, [t_i, t_j] \Vdash p$ iff $p \in \mathcal{V}([t_i, t_j])$;
- $\mathbf{M}, [t_i, t_j] \Vdash \neg\psi$ iff $\mathbf{M}, [t_i, t_j] \nVdash \psi$;
- $\mathbf{M}, [t_i, t_j] \Vdash \psi_1 \vee \psi_2$ iff $\mathbf{M}, [t_i, t_j] \Vdash \psi_1$ or $\mathbf{M}, [t_i, t_j] \Vdash \psi_2$;
- $\mathbf{M}, [t_i, t_j] \Vdash \langle A \rangle \psi$ iff $\exists t_k \in T$, with $t_k > t_j$, such that $\mathbf{M}, [t_j, t_k] \Vdash \psi$.

We place ourselves in the most general setting and we do not impose any constraint on the valuation function. In particular, given an interval $[d_i, d_j]$, it may happen that $p \in \mathcal{V}([d_i, d_j])$ and $p \notin \mathcal{V}([d'_i, d'_j])$ for all intervals $[d'_i, d'_j]$ (properly) included in $[d_i, d_j]$.

2.1 RPNL over linear and branching structures

We conclude the section by pointing out the differences between interpreting RPNL over linear structures and over branching ones. ¿From the satisfiability of an RPNL formula over a linear structure, it immediately follows its satisfiability over a branching one. However, the opposite does not hold in general. Consider the following example. Consider the formula $\varphi_1 \equiv \langle A \rangle [A][A] p \wedge \langle A \rangle [A][A] \neg p \wedge [A]\langle A \rangle \top$ which states that (i) there exists an interval in the future of the current one such that p holds over every interval in its future (the double $[A]$ allows us to refer to all intervals strictly to the right of the current one), (ii) there

[1] It is easy to see that, as far as RPNL is concerned, such trees are indistinguishable from finitely branching trees.

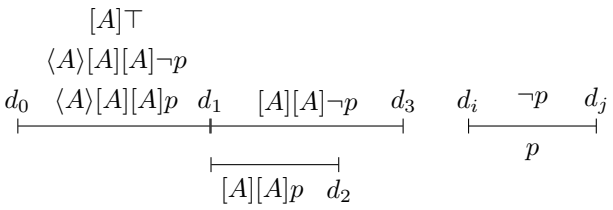

Figure 1. Unsatisfiability of φ_1 over a linear structure.

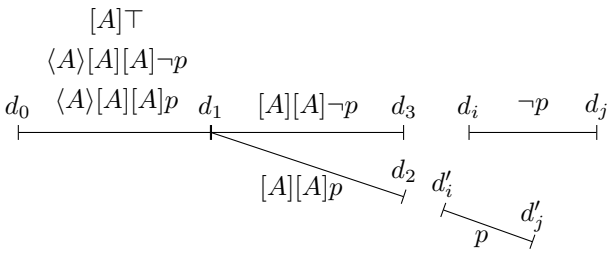

Figure 2. A branching structure satisfying φ_1.

exists an interval in the future of the current one such that $\neg p$ holds over every interval in its future, and (iii) the model is infinite. On a linear ordering, φ_1 forces the existence of an interval (in fact, infinitely many ones) over which both p and $\neg p$ hold, and thus it is clearly unsatisfiable, as shown in Figure 1. On the contrary, it can be easily satisfied over a branching structure forcing condition (i) to hold on a given branch and condition (ii) to hold on another one, as shown in Figure 2. In general, interpretations of RPNL formulae over linear and branching structures present some similarities. In particular, in both of them intervals sharing their right endpoints must satisfy the same temporal formulae, that is, the same $\langle A\rangle\psi$ and $[A]\psi$ formulae. This allows us to associate with any time point the set of *its* temporal formulae. However, interpretations over linear and branching structures differ in two fundamental respects. On the one hand, linear structures feature a single time line over which all existential requests associated with a time point must be fulfilled. As a consequence, the order according to which requests are fulfilled plays often a crucial role. In branching structures, we can introduce as many branches as the existential requests associated with a given time point are and satisfy distinct requests over distinct branches. Hence, every existential request associate with a given time point can be immediately satisfied (in a distinct branch). On the other hand, for any pair of time points of a linear structure, either the past of the first one includes that of the second one or vice versa, while the pasts of a pair of time points of a branching structure can be only partially overlapped. Formally, the past of a time point can be described as a set of sets of temporal requests associated with different time points in its past. In the linear case, we may need to consider an exponential number of such sets (exponential in the number of temporal requests), while in the branching case we may need to take into account a doubly exponential number of sets (the doubly exponential number of subsets of the set of sets of temporal requests).

3 A tableau system for RPNL over trees

In this section, we define a tableau-based decision procedure for RPNL, prove its soundness and completeness, and analyze its computational complexity. As a preliminary step, we introduce some basic notions. Let φ be an RPNL formula to check for satisfiability. For the sake of brevity, we use $(A)\psi$ as a shorthand for both $\langle A\rangle\psi$ and $[A]\psi$.

Definition 2. *The* closure $\mathrm{CL}(\varphi)$ *of φ is the set of all its subformulae and of their negations (we identify $\neg\neg\psi$ with ψ).*

Definition 3. *The set* $\mathrm{TF}(\varphi)$ *is the set of all temporal formulae in* $\mathrm{CL}(\varphi)$*, that is,* $\mathrm{TF}(\varphi) = \{(A)\psi \in \mathrm{CL}(\varphi)\}$.

By induction on the structure of φ, we can easily prove the following proposition.

Proposition 1. *For every formula φ, $|\mathrm{CL}(\varphi)|$ is less than or equal to $2 \cdot |\varphi|$, while $|\mathrm{TF}(\varphi)|$ is less than or equal to $2 \cdot (|\varphi| - 1)$.*

The notion of φ-atom is defined in the standard way.

Definition 4. *A* φ-atom *is a set $A \subseteq \mathrm{CL}(\varphi)$ such that:*

- *for every $\psi \in \mathrm{CL}(\varphi)$, $\psi \in A$ iff $\neg\psi \notin A$;*
- *for every $\psi_1 \vee \psi_2 \in \mathrm{CL}(\varphi)$, $\psi_1 \vee \psi_2 \in A$ iff $\psi_1 \in A$ or $\psi_2 \in A$.*

We denote the set of all φ-atoms by A_φ. We have that $|A_\varphi| \leq 2^{|\varphi|}$. For any φ-atom A, let $\mathrm{REQ}(A)$ be the *set of temporal formule* in A, that is, $\mathrm{REQ}(A) = A \cap \mathrm{TF}(\varphi)$. Given a model $\mathbf{M} = \langle\langle\mathbb{T}, \mathbb{I}(\mathbb{T})\rangle, \mathcal{V}\rangle$ for φ, we define a function $\mathcal{V}_A : \mathbb{I}(\mathbb{T}) \to A_\varphi$ that associates an atom A with every $[d_i, d_j] \in \mathbb{I}(\mathbb{T})$ in such a way that for every $\psi \in CL(\varphi)$, $\psi \in A$ if, and only if, $\mathbf{M}, [d_i, d_j] \models \psi$.
The following theorem proves that atoms associated with intervals whose right endpoints coincide feature the same temporal formulae.

Proposition 2. *Let $\mathbf{M} = \langle\langle\mathbb{T}, \mathbb{I}(\mathbb{T})\rangle, \mathcal{V}\rangle$ be a model for φ and $d_i, d_j, d_e \in T$ be such that both $d_i < d_e$ and $d_j < d_e$. If $\mathcal{V}_A([d_i, d_e]) = A$ and $\mathcal{V}_A([d_j, d_e]) = A'$, then $REQ(A) = REQ(A')$.*

Proof. Suppose, by contradiction, that $\mathcal{V}_A([d_i, d_e]) = A$, $\mathcal{V}_A([d_j, d_e]) = A'$, and $REQ(A) \neq REQ(A')$. Without loss of generality, we can assume that there exists $\langle A\rangle\psi$ such that $\langle A\rangle\psi \in REQ(A)$ and $\langle A\rangle\psi \notin REQ(A')$. By

definition of atom, it follows that $[A]\neg\psi \in REQ(A')$. Since $\mathbf{M}$ is a model for φ, there exists $d_h > d_e$ such that $\mathbf{M}, [d_e, d_h] \models \psi$ (for $\langle A\rangle\psi \in REQ(A)$) and for every $d_k > d_e$ $\mathbf{M}, [d_e, d_k] \models \neg\psi$ (for $[A]\neg\psi \in REQ(A')$). For $k = h$, we have $\mathbf{M}, [d_e, d_h] \models \neg\psi$ (contradiction). □

Atoms are connected by the following binary relation.

Definition 5. *Let R_φ be a binary relation over A_φ such that, for every pair of atoms $A, A' \in A_\varphi$, $A\ R_\varphi\ A'$ if, and only if, for every $[A]\psi \in \mathrm{CL}(\varphi)$, if $[A]\psi \in A$, then $\psi \in A'$.*

3.1 The tableau system

A tableau for an RPNL formula φ is a suitable *decorated tree* $\mathcal{T}$. A finite prefix of the natural numbers $\mathbb{D}_B = \langle D_B, <\rangle$ is associated with every branch B of $\mathcal{T}$. The *decoration* of a node n in $\mathcal{T}$, denoted by $\nu(n)$, is a pair $\langle [d_i, d_j], A\rangle$, where A is an atom and $d_i, d_j \in D_B$ for every branch B containing n. Given a node n, we denote by $A(n)$ the atom in $\nu(n)$.

Expansion rules. Tableau construction is based on the following expansion rules. Let n be a leaf node of the current tableau $\mathcal{T}$ with decoration $\langle [d_i, d_j], A_n\rangle$. Since n is a leaf, there is a unique branch B containing n in $\mathcal{T}$. Let $\mathbb{D}_B$ be the finite strictly ordered set associated with B. The following *expansion rules* can be possibly applied to n:

1. *Fill-in rule*. Let $d \in D_B$, with $d_0 < d < d_j$, be such that there are no ancestors n' of n with decoration $\langle [d, d_j],\ A'\rangle$, for a suitable A'. If there exists an atom A'' such that $\mathrm{REQ}(A'') = \mathrm{REQ}(A_n)$ and for all ancestors $\overline{n}$ of n with decoration $\langle [\overline{d}, d], \overline{A}\rangle$, for suitable $\overline{d}$ and $\overline{A}$, $\overline{A}\ R_\varphi\ A''$, we add an immediate successor n'' to n with decoration $\langle [d, d_j], A''\rangle$.
2. *Step rule*. Let $\{\langle A\rangle\psi_1, \ldots, \langle A\rangle\psi_k\}$, with $k \geq 1$, be the set of $\langle A\rangle$-formulae in A_n. If there exist k atoms $A'_1, .., A'_k$ such that, for $1 \leq h \leq k$, $A_n\ R_\varphi\ A'_h$ and $\psi_h \in A'_h$, we add k immediate successors n'_h, with decoration $\langle [d_j, d_{j+1}], A'_h\rangle$, to n. Let $B_1, ..., B_k$ be the new added branches. For $1 \leq h \leq k$, $\mathbb{D}_{B_h}$ is obtained from $\mathbb{D}_B$ by adding a new point d_{j+1} greater than all points in D_B.

The fill-in rule adds one successor to n, the step rule one or more. However, while the step rule decorates every successor with a new interval ending at a new point d_{j+1}, the fill-in rule decorates the successor with a new interval whose endpoints are already in $\mathbb{D}_B$. Moreover, the fill-in rule forces atoms associated with intervals with the same right endpoint d to agree on their temporal requests.

Definition 6. *For all branches B in $\mathcal{T}$ and for all $d > d_0$ in $\mathbb{D}_B$, we define the set of temporal formulae associated with d, denoted $REQ(d)$, as the set $REQ(A(n))$ for all nodes n in B with decoration $\langle [\overline{d}, d], A\rangle$.*

Blocking condition. To guarantee the termination of the tableau construction, we impose a *blocking condition* that prevents one from applying infinitely many times the expansion rules in the case of infinite models. We say that a leaf node n with decoration $\langle [d_i, d_j], A_n\rangle$ belonging to a branch B is *blocked* if there exists an ancestor n' of n with decoration $\langle [d_k, d_l], A_{n'}\rangle$, with $d_l < d_j$ in $\mathbb{D}_B$, such that:

$\mathrm{REQ}(A_n) = \mathrm{REQ}(A_{n'})$, and
$\forall d_h(d_h < d_j \rightarrow \exists d_g(d_g < d_l \wedge \mathrm{REQ}(d_h) = \mathrm{REQ}(d_g)))$.

Roughly speaking, we block a leaf node n if it has an ancestor n' with the same set of temporal formulae and every set of temporal formulae that occurs in the path from n' to n also occurs in the path from the root to n'.

Expansion strategy. Given a decorated tree $\mathcal{T}$ and a branch B of $\mathcal{T}$ ending in a leaf node n, we say that an expansion rule is *applicable* (to n) if n is (non-closed,) non-blocked and its application generates at least one new node. To any branch B ending in a leaf node n, with decoration $\langle [d_i, d_j], A_n\rangle$, we apply the following *branch-expansion strategy*:

1. if the fill-in rule is applicable, *apply* it to n;
2. if the fill-in rule is not applicable and there exists a point d_k in $\mathbb{D}_B$, with $d_k < d_j$, such that there are no ancestors of n with decoration $\langle [d_k, d_j], A'\rangle$, for a suitable A', *close* the node n;
3. if the fill-in rule is not applicable and n is not closed, *apply* the step rule to n, if it is applicable.

Notice that, at step 1, the fill-in rule is applied exhaustively to add a node for each missing interval.

Tableau. Let φ be the formula to check for satisfiability. An *initial tableau* for φ is a decorated tree with one single node $\langle [d_0, d_1], A\rangle$, with $\varphi \in A$ and $\mathbb{D}_B = \{d_0 < d_1\}$. A *tableau* for φ is any decorated tree $\mathcal{T}$ obtained by expanding an initial tableau for φ through successive applications of the branch-expansion strategy to the leaves, until it cannot be applied anymore.

Pruning the tableau. Given a tableau $\mathcal{T}$ for φ, we apply the following pruning procedure until no further nodes can be removed:

1. remove any closed node from the tableau;
2. remove any node n devoid of successors such that the fill-in rule has been applied to it during the tableau construction;
3. remove any node n such that the step rule has been applied to it during the tableau construction and there exists $\langle A\rangle\psi \in A_n$ such that there is no successor n' of n with $\psi \in A(n')$;
4. remove every node which is not reachable from the root.

We shall prove that an RPNL formula φ is satisfiabile if, and only if, there exists a non-empty pruned tableau for it.

3.2 An example

We conclude the section by applying the proposed tableau method to the formula $\varphi_2 = \langle A\rangle(\langle A\rangle\top \wedge [A]\langle A\rangle\top) \wedge \langle A\rangle[A]\bot$. A non-empty pruned tableau for φ_2 is depicted in Figure 3. We associate a linear ordering $\mathbb{D}_i = \{d_0 < \ldots < d_i\}$ with every node. It represents the ordering associated with the branch ending at that node. Three atoms come into play: $A_0 = \{\top, \langle A\rangle\top, \langle A\rangle[A]\bot, \langle A\rangle\psi, \neg\psi, \varphi_2\}$, $A_1 = \{\top, \langle A\rangle\top, [A]\langle A\rangle\top, \langle A\rangle\psi, \psi, \neg\varphi_2\}$, and $A_2 = \{\top, [A]\bot, [A]\langle A\rangle\top, [A]\neg\psi, \neg\psi, \neg\varphi_2\}$, where ψ is a shorthand for $\langle A\rangle\top \wedge [A]\langle A\rangle\top$. The relation R_φ over them is defined as follows: $R_\varphi = \{(A_0, A_0), (A_0, A_1), (A_0, A_2), (A_1, A_0), (A_1, A_1)\}$. The root of the tableau is the node n_0. We apply the step rule to it. Since atom A_0 contains three $\langle A\rangle$-formulae, we add three successors to n_0 whose decorations include atoms A_0, A_1, and A_2, respectively. $\langle A\rangle\top$ is dealt with by node n_1, which turns out to be blocked. $\langle A\rangle[A]\bot$ is dealt with by node n_3. Since A_2 does not contain $\langle A\rangle$-formulae, n_3 is not expanded further. Finally, $\langle A\rangle\psi$ is dealt with by node n_2. Since the atom A_1 contains the $\langle A\rangle$-formulae $\langle A\rangle\top$ and $\langle A\rangle\psi$, we apply the step rule to n_2, which produces two successors nodes n_4 and n_5 whose decorations include the atoms A_0 and A_1, respectively. As for node n_4, it does not satisfy the blocking condition and thus we apply the fill-in rule to it, which produces

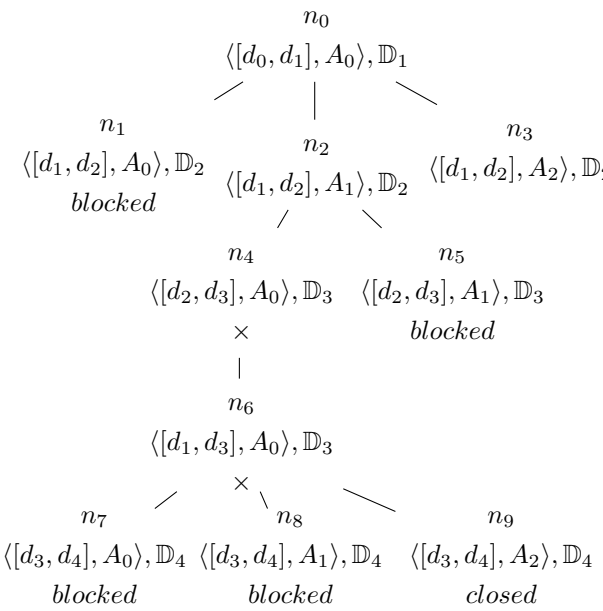

Figure 3. A tableau for the formula $\varphi_2 = \langle A\rangle(\langle A\rangle\top \wedge [A]\langle A\rangle\top) \wedge \langle A\rangle[A]\bot$

a new node n_6 associating an atom with the interval $[d_1, d_3]$. Then, since the fill-in rule is no more applicable, we apply the step rule to node n_6, that adds three successor nodes n_7, n_8, and n_9. Nodes n_7 and n_8 satisfy the blocking condition, while node n_9 does not satisfy it. At this point, the construction should proceed with a double application of the fill-in rule to node n_9 to associate suitable atoms with the intervals $[d_1, d_4]$ and $[d_2, d_4]$. However, the fill-in rule turns out to be unapplicable, since no suitable atom can be found for the interval $[d_2, d_4]$. Thus, node n_9 is closed. As for node n_5, it immediately turns out to be blocked. This concludes the tableau construction. Node n_9 is closed and thus removed by condition 1 of the pruning procedure. Since neither A_0 nor A_1 contains $[A]\bot$, there exists a formula in A_0, namely, $\langle A\rangle[A]\bot$, such that there are no successors n' of the node n_6 with $[A]\bot \in A(n')$. Hence, node n_6 is removed by condition 3 of the pruning procedure. Such a removal makes n_7 and n_8 no more reachable from the root and thus, by condition 4 of the pruning procedure, they are removed from the tableau. Finally, condition 2 of the pruning procedure forces the removal of node n_4. Since no further removal steps can be applied, the resulting pruned tableau is not empty.

3.3 Soundness and completeness

In this section, we prove the soundness and completeness of the method. Soundness is proved by showing how to construct a model satisfying φ from a non-empty pruned tableau $\mathcal{T}$ for it. Conversely, completeness is proved by showing that, for any satisfiable formula φ, there exists a non-empty pruned tableau for it.

Theorem 1 (Soundness)**.** *Given a formula φ and a pruned tableau $\mathcal{T}$ for it, if $\mathcal{T}$ is non-empty, then φ is satisfiable.*

Proof. Let $\mathcal{T}$ be a non-empty pruned tableau for φ. We show that we can build a model for φ based on $\mathcal{T}$. Since $\mathcal{T}$ is not empty, it has a root with decoration $\langle[d_0, d_1], A_0\rangle$ and $\mathbb{D}_B = \{d_0 < d_1\}$. We start the construction with a partial model, which consists of a two-node tree $\mathbb{T}_0 = \{d_0, d_1\}$ and a valuation $\mathcal{V}[d_0, d_1] = \{p : p \in A_0\}$. Then, we progressively turn it into a model for φ by a depth-first visit of $\mathcal{T}$. Let $\mathbf{M}$ be the current partial model, n be the current node of $\mathcal{T}$, with decoration $\langle[d_i, d_j], A_n\rangle$, and $\mathbb{D}_B$ be the ordering associated with the branch B ending in n. We proceed by induction on the expansion rule that has been applied to n.

- *No expansion rule has been applied to n and n is not blocked.* In such a case, there are no $\langle A\rangle$-formulae in A_n and thus we do not need to expand the model.
- *The step rule has been applied to n.* For every formula $\langle A\rangle\psi \in A_n$, there exists a successor n_ψ of n such that $\psi \in A(n_\psi)$. We expand the partial model by

adding an immediate successor d_ψ to d_j and by putting $\mathcal{V}[d_j, d_\psi] = \{p : p \in A(n_\psi)\}$.

- *The fill-in rule has been applied to* n. The decoration of the successor of n includes an interval $[d_k, d_j]$ and an atom A'. We expand the model by putting $\mathcal{V}[d_k, d_j] = \{p : p \in A'\}$.

- *The node* n *is blocked*. In such a case, there exists an ancestor n' of n, with decoration $\langle [d_k, d_l], A_{n'} \rangle$, such that $\mathrm{REQ}(A_n) = \mathrm{REQ}(A_{n'})$ and for all $d_h < d_j$ in $\mathbb{D}_B$ there exists $d_m < d_l$ with $\mathrm{REQ}(d_h) = \mathrm{REQ}(d_m)$. Since no new atoms occur between n' and n, we can proceed with the model construction from n as we did from n'.

It is easy to see that such a (possibly infinite) construction produces a model for φ. □

Theorem 2 (Completeness)**.** *Given a satisfiable formula* φ*, there exists a non-empty pruned tableau* $\mathcal{T}$ *for* φ*.*

Proof. Let φ be a satisfiable formula and let $\mathbf{M} = \langle \langle \mathbb{T}, \mathbb{I}(\mathbb{T}) \rangle, \mathcal{V} \rangle$ be a model for it. We prove that there exists a non-empty pruned tableau $\mathcal{T}$ corresponding to $\mathbf{M}$. Since $\mathbf{M}$ is a model for φ, there exists an interval $[d_0, d_1]$ such that $\mathbf{M}, [d_0, d_1] \Vdash \varphi$. Let $A_0 = \{\psi \in \mathrm{CL}(\varphi) : \mathbf{M}, [d_0, d_1] \Vdash \psi\}$. We start the construction of $\mathcal{T}$ with a partial tableau which includes a single node with decoration $\langle [d_0, d_1], A_0 \rangle$ and $\mathbb{D}_B = \{d_0 < d_1\}$. Then, we proceed by induction on expansion rules. Let $\mathcal{T}$ be the current partial tableau, n be a leaf node with decoration $\langle [d_i, d_j], A_n \rangle$, and $\mathbb{D}_B$ be the ordering associated with the branch ending in n. Three cases may arise.

- *The fill-in rule is applicable to* n. Let d_k be a point such that there are no nodes associated with $[d_k, d_j]$. We add a successor n' to n with decoration $\langle [d_k, d_j], A' \rangle$, where $A' = \{\psi \in \mathrm{CL}(\varphi) : \mathbf{M}, [d_k, d_j] \Vdash \psi\}$.

- *The step rule is applicable to* n. For every $\langle A \rangle \psi \in A_n$, there exists an interval $[d_j, d_\psi]$ such that $\mathbf{M}, [d_j, d_\psi] \Vdash \psi$. Let $A_\psi = \{\theta \in \mathrm{CL}(\varphi) : \mathbf{M}, [d_j, d_\psi] \Vdash \theta\}$. For every $\langle A \rangle \psi \in A_n$, we add a successor n_ψ to n with decoration $\langle [d_j, d_\psi], A_\psi \rangle$ and, for every new branch B', we define $\mathbb{D}_{B'} = \mathbb{D}_B \cup \{d_\psi\}$.

- *No rule is applicable to* n. Since $\mathbf{M}$ is a model for φ, n cannot be a closed node. Hence, either there are no $\langle A \rangle$-formulae in A_n or n is blocked.

It is easy to prove that the tableau $\mathcal{T}$ generated by such a procedure is a non-empty tableau for φ to which no removal step can be applied. □

3.4 Computational complexity

As a preliminary step, we show that the proposed tableau method terminates by providing a bound on the length of any branch B of any tableau for φ.

Let $n = |\varphi|$. We have that $REQ(d)$ can take $2^{|TF(\varphi)|}$, where $|TF(\varphi)| \leq 2 \cdot (n-1)$, different values and there can be at most $2^{|TF(\varphi)|}$ different sets of requests associated with time points $d' < d$ Hence, by the blocking condition, after at most $O(2^n)$, applications of the step rule, the expansion strategy cannot be applied anymore to a branch.

Moreover, given a branch B, between two consecutive applications of the step rule, the fill-in rule can be applied at most $m-3$ times, where $m = |D_i|$ and D_i is the underlying set of the linear ordering associated with the last node of B (in fact, $m-2$ is exactly the number of applications of the step rule up to that point).

This allows us to conclude that the length of a branch is (at most) exponential in n. Since the outgoing degree of every node is bounded by the number of $\langle A \rangle$-formulae in $\mathrm{CL}(\varphi)$, the size of a tableau is thus $O(2^{2^n})$. The following theorem immediately follows.

Theorem 3. *The decision problem for* RPNL *over trees is in EXPSPACE.*

Proof. The proposed decision procedure does not need to explicitly generate the whole tableau, but it can keep track of a branch at a time and expand it in a non-deterministic way. Since the length of any branch is at most $O(2^n)$, the procedure is in EXPSPACE. □

To prove EXPSPACE-hardness of the decision problem for RPNL interpreted over trees, we exploit *Alternating Turing Machines* (ATM for short) [7]. An ATM is a tuple $M = (Q, \Gamma, \delta, q_0, g)$, where (Q, Γ, δ, q_0) is a one-tape non-deterministic Turing Machine and g is a function $g : Q \rightarrow \{\vee, \wedge, accept, reject\}$ that classifies the states of M. Given an input word w and the computation tree of the one-tape non-deterministic Turing Machine (Q, Γ, δ, q_0) on w, we say that a configuration $C = (q, v, i)$, that is, a node of the computation tree, is *accepting* if either $g(q) = accept$, or $g(q) = \vee$ and at least one successor of C is accepting, or $g(q) = \wedge$ and all successors of C are accepting. We say that an ATM M *accepts* w if the root $(q_0, w, 1)$ of the computation tree is an accepting configuration. It is possible to prove that the complexity class AEXPTIME, that is, the class of the problems that can be decided in exponential time by an ATM, corresponds to the complexity class EXPSPACE [7].

Theorem 4. *The decision problem for* RPNL *over trees is EXPSPACE-hard.*

Proof. Without loss of generality, we can assume that every non final configuration of M has exactly two successor configurations and that once the machine reaches an accepting

or a rejecting state, it remains in that state forever, without changing the contents of the tape.

Let w be the input word and M be an ATM that runs in time 2^n, where $n \in O(|w|)$. We build a formula φ whose models encode accepting computation trees for M on input w. Every branch in the computation tree includes 2^n configurations. Every configuration, which consists of the current state q, the current position of the head i, and the contents of 2^n tape cells, is represented by 2^n elements of the model. We encode every level c of the computation tree (from level 0 to level $2^n - 1$) by means of n propositional letters C_1, ..., C_n. Moreover, we encode every position p in the tape by means of n propositional letters P_1, ..., P_n. Given a level c (resp., a tape position p), we denote by $c + 1$ (resp., $p + 1$) the next level (resp., the next tape position). Moreover, we introduce h propositional letters $A_1, \ldots, A_h$, with $h = |\Gamma|$, for the symbols in the alphabet Γ, m propositional letters $Q_1, \ldots, Q_m$, with $m = |Q|$, for the states in Q, and a propositional letter H for the head of M.

As a first step, we impose a sort of *locality principle* on all the above-mentioned propositional letters [6], according to which each of them assumes the same truth value over intervals starting at the same time point, that is, for every propositional letter R, *R holds over an interval $[d_i, d_j]$ if, and only if, R holds over $[d_i, d_k]$ for every $d_k > d_i$*. It allows us to interpret every point d of the model as pair $(c(d), p(d))$, where $c(d)$ is a level and $p(d)$ is a position. Such a condition is imposed by means of the formula $(\langle A\rangle R \rightarrow [A]R) \wedge [A](\langle A\rangle R \rightarrow [A]R)$. Let ψ_{loc} be the conjunction of these formulae.

Next, we provide some auxiliary formulae that will be used to encode the behavior of the ATM. First, we introduce the formulae $\psi^C_= = \wedge^n_{i=1}(C_i \leftrightarrow [A]C_i)$ and $\psi^P_= = \wedge^n_{i=1}(P_i \leftrightarrow [A]P_i)$ such that, for any interval $[d_i, d_j]$, $\psi^C_=$ (resp., $\psi^P_=$) holds over $[d_i, d_j]$ if, and only if, $c(d_i) = c(d_j)$ (resp., $p(d_i) = p(d_j)$). Next, we introduce the auxiliary formulae ψ^P_{+1} and ψ^P_{-1} such that, for any interval $[d_i, d_j]$, ψ^P_{+1} (resp., ψ^P_{-1}) holds over $[d_i, d_j]$ if, and only if, $c(d_j) = c(d_i) + 1$ (resp., $c(d_j) = c(d_i) - 1$). Such formulae are defined as follows:

$$
\begin{aligned}
\psi^P_{+1}(n) &= \neg P_n \wedge [A]P_n \\
\psi^P_{+1}(k) &= (\neg P_k \wedge [A]P_k \wedge \wedge^n_{i=k+1}(P_i \leftrightarrow [A]P_i)) \vee \\
&\quad ((P_k \wedge [A]\neg P_k) \wedge \psi^P_{+1}(k+1)) \\
\psi^P_{+1} &= \psi^P_{+1}(1) \\
\psi^P_{-1}(n) &= P_n \wedge [A]\neg P_n \\
\psi^P_{-1}(k) &= (P_k \wedge [A]\neg P_k \wedge \wedge^n_{i=k+1}(P_i \leftrightarrow [A]P_i)) \vee \\
&\quad ((\neg P_k \wedge [A]P_k) \wedge \psi^P_{+1}(k+1)) \\
\psi^P_{-1} &= \psi^P_{-1}(1)
\end{aligned}
$$

Analogously, we introduce a formula ψ^C_{+1} that holds over an interval $[d_i, d_j]$ if, and only if, $c(d_j) = c(d_i) + 1$, which can be obtained by substituting C for P everywhere in the previous formulae (notice that a formula ψ^C_{-1} is not needed).

The behavior of the ATM can be encoded as follows. For the sake of brevity, we use the shorthand $[U]\psi$ for $\psi \wedge [A]\psi \wedge [A][A]\psi$. First, we impose that every element of the model d_i where at least one among $C_1, \ldots, C_n, P_1, \ldots, P_n$ evaluates to false has a successor d_j. Moreover, if every P_i evaluates to true in d_i, that is, if d_i represents the last tape cell, then every P_i evaluates to false in d_j and $c(d_j) = c(d_i) + 1$; otherwise, $c(d_j) = c(d_i)$ and $p(d_j) = p(d_i) + 1$. Such a condition is imposed by the formula $[U]\psi_{succ}$, where ψ_{succ} is defined as follows:

$$
\begin{aligned}
\psi_{succ} &= (\psi_{next} \wedge \neg \wedge^n_{i=1}(P_i \wedge C_i)) \rightarrow \langle A\rangle\psi_{next} \\
\psi_{next} &= (\wedge^n_{i=1}(P_i \wedge [A]\neg P_i) \wedge \psi^C_{+1}) \vee (\psi^C_= \wedge \psi^P_{+1})
\end{aligned}
$$

Next, we impose that every tape cell contains only one symbol of Γ and that, in a given configuration, the head is associated only with one tape position by means of the formulae $[U]\psi_a$, where $\psi_a = \vee^h_{i=1}A_i \wedge \wedge^h_{i=1}(A_i \rightarrow \wedge_{j\neq i}\neg A_j)$, and $[U]\psi_{head}$, where $\psi_{head} = (H \wedge \psi^C_=) \rightarrow [A]\neg H$. Furthermore, we associate the state of the machine M with the head position by means of the formula $[U]\psi_{state}$, where $\psi_{state} = (H \leftrightarrow \vee^m_{i=1}Q_i) \wedge \wedge^m_{i=1}(Q_i \rightarrow \wedge_{j\neq i}\neg Q_j)$.

Now, we have to ensure that the sequence of configurations respects the transitions of M. First of all, we impose that if a position in a configuration does not contain the head, its symbol remains unchanged in the next configuration by means of the formula $[U]\psi_{apos}$, where $\psi_{apos} = (\psi^P_= \wedge \psi^C_{+1} \wedge \neg H) \rightarrow \wedge^h_{i=1}(A_i \rightarrow [A]A_i)$. By definition of ATM, two consecutive configurations may differ only in the state, in the symbol associated with the current cell, and in the head position, that can move left or right. Let $\delta(Q, A) = (Q', A', \sim_1), (Q'', A'', \sim_2)$, where $\sim_1, \sim_2 \in \{\rightarrow, \leftarrow\}$, be a transition of M. Suppose that $\sim_1 = \rightarrow$ and $\sim_2 = \leftarrow$ (the other cases are similar). Moreover, let:

$$
\begin{aligned}
\psi^\delta_1 &= (Q \wedge A \wedge H \wedge \psi^P_= \wedge \psi^C_{+1}) \rightarrow \\
&\quad \langle A\rangle(A' \wedge \psi_{next} \wedge \langle A\rangle(Q' \wedge H)) \\
\psi^\delta_2 &= (Q \wedge A \wedge H \wedge \psi^P_{-1} \wedge \psi^C_{+1}) \rightarrow \\
&\quad \langle A\rangle(Q'' \wedge H \wedge \psi_{next} \wedge \langle A\rangle(A''))
\end{aligned}
$$

If Q is an $\vee$-state, we encode the transition $\delta(Q, A)$ with the formula $\psi^\delta_\vee = [U](\psi^\delta_1 \vee \psi^\delta_2)$, while if Q is an $\wedge$-state, we encode the transition with the formula $\psi^\delta_\wedge = [U](\psi^\delta_1 \wedge \psi^\delta_2)$.

Let w the input word. We denote by $w(i)$ the i^{th} symbol of w and by $A_{w(i)}$ the corresponding propositional letter. Let Q_0 be the initial state of M. We encode the initial configuration of the machine with the formula ψ_{init} defined as

follows:

$$\begin{aligned}
\psi_{init} &= \psi_{in}(1) \wedge H \wedge Q_0 \wedge \wedge_{i=1}^{n}(\neg C_i \wedge \neg P_i);\\
\psi_{in}(|w|) &= \psi_{next} \wedge A_{w(|w|)} \wedge [A](\wedge_{i=1}^{n}\neg C_i \rightarrow \\
&\quad A_{blank}) \wedge [A][A](\wedge_{i=1}^{n}\neg C_i \rightarrow A_{blank});\\
\psi_{in}(k) &= \psi_{next} \wedge A_{w(k)} \wedge \langle A\rangle \psi_{in}(k+1),\\
&\quad \text{with } k < |w|,
\end{aligned}$$

where A_{blank} is a propositional letter associated with the blank symbol in Γ. Finally, w.l.o.g., we assume that Q_{reject} is the unique rejecting state of M and we encode the accepting condition with the formula $\psi_{acc} = [U]\neg Q_{reject}$. □

4 Conclusions

In this paper, we solved the satisfiability problem for the future fragment of PNL, interpreted over trees, by providing an EXPSPACE tableau-based decision procedure. Moreover, we proved the EXPSPACE-hardness of the problem. We are currently looking for a possible generalization of the method to full PNL interpreted over trees and over partial orderings with the linear interval property.

References

[1] D. Bresolin, V. Goranko, A. Montanari, and P. Sala. Tableau-based decision procedure for the logic of proper subinterval structures over dense orderings. In *Proc. of the 5th Int. Workshop on Methods for Modalities (M4M)*, pages 335–351, 2007.

[2] D. Bresolin, V. Goranko, A. Montanari, and P. Sala. Tableau systems for logics of subinterval structures over dense orderings. In *Proc. of TABLEAUX 2007*, volume 4548 of *LNAI*, pages 73–89. Springer, 2007.

[3] D. Bresolin, V. Goranko, A. Montanari, and G. Sciavicco. On Decidability and Expressiveness of Propositional Interval Neighborhood Logics. In *Proc. of the International Symposium on Logical Foundations of Computer Science (LFCS)*, volume 4514 of *LNCS*, pages 84–99. Springer, 2007.

[4] D. Bresolin and A. Montanari. A tableau-based decision procedure for a branching-time interval temporal logic. In *Proc. of the 4th Int. Workshop on Methods for Modalities (M4M)*, pages 38–53, 2005.

[5] D. Bresolin, A. Montanari, and P. Sala. An optimal tableau-based decision algorithm for Propositional Neighborhood Logic. In *Proc. of the 24th International Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 4393 of *LNCS*, pages 549–560. Springer, 2007.

[6] D. Bresolin, A. Montanari, and G. Sciavicco. An optimal decision procedure for Right Propositional Neighborhood Logic. *Journal of Automated Reasoning*, 38(1-3):173–199, 2007.

[7] A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.

[8] V. Goranko, A. Montanari, and G. Sciavicco. Propositional interval neighborhood temporal logics. *Journal of Universal Computer Science*, 9(9):1137–1167, 2003.

[9] V. Goranko, A. Montanari, and G. Sciavicco. A road map of interval temporal logics and duration calculi. *Journal of Applied Non-Classical Logics*, 14(1–2):9–54, 2004.

[10] V. Goranko, A. Montanari, G. Sciavicco, and P. Sala. A general tableau method for propositional interval temporal logics: Theory and implementation. *Journal of Applied Logic*, 4(3):305–330, 2006.

[11] Joseph Y. Halpern and Yoav Shoham. A propositional modal logic of time intervals. *Journal of the ACM*, 38(4):935–962, October 1991.

[12] A. Montanari. Propositional interval temporal logics: some promising paths. In *Proc. of the 12th International Symposium on Temporal Representation and Reasoning (TIME)*, pages 201–203. IEEE Computer Society Press, 2005.

[13] 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 AI*, volume 2424 of *LNAI*, pages 259–270. Springer, 2002.

[14] B. Moszkowski. *Reasoning about digital circuits*. Tech. rep. stan-cs-83-970, Dept. of Computer Science, Stanford University, Stanford, CA, 1983.

[15] M. Otto. Two variable first-order logic over ordered domains. *Journal of Symbolic Logic*, 66(2):685–702, 2001.

[16] I. Shapirovsky and V. Shehtman. Chronological future modality in Minkowski spacetime. In P. Balbiani, N. Y. Suzuki, F. Wolter, and M. Zakharyaschev, editors, *Advances in Modal Logic*, volume 4, pages 437–459. King's College Publications, London, 2003.

[17] Y. Venema. A modal logic for chopping intervals. *Journal of Logic and Computation*, 1(4):453–476, 1991.