Decidability of Interval Temporal Logics

CANDIDATE:
Dott. Pietro Sala

SUPERVISOR:
Prof. Angelo Montanari
Author’s e-mail: pietro.sala@dimi.uniud.it

Author’s address:
Dipartimento di Matematica e Informatica
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine
Italia
...to Victor since his eyes will never stop to look proudly at me.
First of all I want to spend a few words about the two guys that gave the main contributions to my formative path during the last three years, prof. Angelo Montanari and dott. Gabriele Puppis. Prof. Angelo Montanari, as you may notice from the cover of this work, is my supervisor and during these years he gave full support to all my ideas. I learned many things from this fruitful collaboration, his helpfulness together with his polite behaviour creates the perfect work environment. Finally his contribution to the presentation of the results is consistent in every chapter of the present work. Dott. Gabriele Puppis is, in my opinion, one of the best co-author that one can have. From my point of view his outstanding professional qualities are overcame only by his human ones. He is a very positive person and his passion for research and life is highly contagious. I want to thank dott. Davide Bresolin, prof. Valentin Goranko and dott. Guido Sciavicco for their collaboration into many of the results presented here. The direct interaction with colleagues from the other countries during conferences and summer schools has contributed significantly to my scientific growth. For this reason I would like to thank, for their financial support, prof. Fabio Alessi, prof. Maria Stanizskis and prof. Agostino Dovier. In particular prof. Agostino Dovier who covered the greatest part of these expenses behaved more like friend than professor. One of my favourite blues song says "... nobody knows you when you are down and out... " but, since I am in the logic business, I found some counter-examples to this statement. I will never stop to thank Elena Benedetti, Barbara Picchio, Gabriele Puppis and Luca Vassena for their true friendship. I would like to thank Clarissa who had patiently removed every thorn from my heart. Maybe the fate put to much distance between I and my "brothers" Daniele Amidani and Lino Baker but I will never forget their sincere help in too many hard situations. Another person that I will never forget is Luca Bellotto who had tragically died last November, he was a pure at heart person and kept me and my old friends together for a long time. I want to thank Giacomina de Nard, my uncle Valerio and my aunt Teresa for their lovable support. Finally I would like to thank my mother who constantly acts looking at my happiness.
Contents

1 Introduction 1

2 Propositional Neighbourhood Logic over linear orders 5
   2.1 Propositional Neighborhood Logic .................................................. 5
   2.2 Labeled Interval Structures and PNL Satisfiability .......................... 7
   2.3 Decidability of PNL over different classes of linear orderings ............ 10
      2.3.1 Decidability of PNL over the class of all linear orders .............. 11
      2.3.2 Decidability of PNL over the class of dense linear orders ........... 14
      2.3.3 Decidability of PNL over the class of discrete linear orders ......... 15
      2.3.4 Decidability of PNL over the integers .................................. 21
   2.4 Tableau-based decision procedures for PNL ..................................... 29
      2.4.1 A tableau-based decision procedure for PNL over all linear orders .. 29
      2.4.2 A tableau-based decision procedure for PNL over dense linear orders 31
      2.4.3 A tableau-based decision procedure for PNL over discrete linear orders 32
      2.4.4 A tableau-based decision procedure for PNL over the integers .......... 34
   2.5 Optimality of the proposed methods .............................................. 37

3 The future fragment of PNL interpreted over trees 41
   3.1 RPNL syntax and semantics .......................................................... 41
      3.1.1 RPNL over linear and branching structures ................................ 42
   3.2 A tableau system for RPNL over trees ............................................ 43
      3.2.1 The tableau system ............................................................... 44
      3.2.2 An example ............................................................................ 45
      3.2.3 Soundness and completeness .................................................... 46
      3.2.4 Computational complexity ....................................................... 47

4 WSpPNL: a decidable spatial extension of PNL 51
   4.1 A brief introduction on spatial logics .............................................. 51
   4.2 SpPNL and WSpPNL ........................................................................ 52
   4.3 WSpPNL Expressiveness ................................................................. 54
   4.4 WSpPNL Decidability and Complexity ............................................. 55
   4.5 The Tableau Method ....................................................................... 63

5 D-logic over dense linear orderings 69
   5.1 Preliminaries .................................................................................... 70
   5.2 The logic of strict subinterval structures ......................................... 71
      5.2.1 Structures for formulas ............................................................. 71
      5.2.2 A small-model theorem for structures ...................................... 74
      5.2.3 A tableau method for .............................................................. 75
      5.2.4 Termination, complexity, soundness, and completeness ........... 77
   5.3 The logic of proper subinterval structures ....................................... 80
      5.3.1 Structures for formulas ............................................................. 80
      5.3.2 A small-model theorem for structures ...................................... 84
      5.3.3 The tableau method for ........................................................... 85
      5.3.4 Soundness and completeness .................................................... 91
   5.4 Implementation ................................................................................. 94
      5.4.1 A short overview of Lotree ...................................................... 94
5.4.2 Lotrec implementation of the tableau for $\mathbb{D}$ ................................. 96
5.4.3 An application example ................................................................. 96

6 Cone Logic ...................................................................................... 99
  6.1 Introduction .................................................................................. 99
  6.2 Syntax and semantics of Cone Logic ............................................. 99
  6.3 Basic machinery: types, dependencies and shadings ...................... 101
  6.4 From the rational plane to the infinite binary tree ......................... 102
      6.4.1 Shading sequences and stripe expressions ............................. 102
      6.4.2 Recursive decompositions of stripes ................................. 104
  6.5 Reducing Cone Logic to a proper fragment of CTL ....................... 108
  6.6 Cone Logic and interval temporal logics ....................................... 110
  6.7 Conclusions .............................................................................. 112

7 $\mathbb{A}\mathbb{B}$-logic ............................................................................. 115
  7.1 The interval temporal logic $\mathbb{A}\mathbb{B}$ ........................................... 115
      7.1.1 Syntax and semantics .......................................................... 116
      7.1.2 Atoms, types, and dependencies ....................................... 116
      7.1.3 Compass structures ............................................................ 117
  7.2 Deciding the satisfiability problem for $\mathbb{A}\mathbb{B}$ .......................... 118
      7.2.1 A small-model theorem for finite structures ....................... 118
      7.2.2 A small-model theorem for infinite structures ................... 121
  7.3 Tight complexity bounds to the satisfiability problem for $\mathbb{A}\mathbb{B}$ .... 123

8 $\mathbb{A}\overline{\mathbb{A}\mathbb{B}}$-logic ................................................................. 131
  8.1 The interval temporal logic $\mathbb{A}\overline{\mathbb{A}\mathbb{B}}$ ............................... 131
  8.2 Decidability and complexity of the satisfiability problem for $\mathbb{A}\overline{\mathbb{A}\mathbb{B}}$ over finite linear orders .................................. 133
  8.3 Undecidability is the rule, decidability the exception ...................... 141

9 Conclusions ..................................................................................... 145

Bibliography ..................................................................................... 149
1.1 The thirteen Allen’s relations between intervals. ........................... 1
2.1 Unsatisfiability of AccPoints over \( \mathbb{N} \): case (1). .......................... 7
2.2 Unsatisfiability of AccPoints over \( \mathbb{N} \): case (2). .......................... 7
2.3 A model for AccPoints over the class of linearly ordered domains. .......... 7
2.4 Relabeling of the interval \([d, d]_i\) in Case 1a. ............................... 11
2.5 Labeling of the intervals \([d, d']\) and \([d', d]\) in Case 1b. .................... 12
2.6 Labeling of intervals starting/ending in \( \hat{d} \) in Case 2c. ....................... 13
2.7 Labeling of intervals starting/ending in \( \hat{d} \) in case 1b (when \( d_{j} = \max(\text{interval}(d_{j})) \)). ............................ 17
2.8 Possible relationships between \( d' \) and \( d_{j} \) in case \( d_{j} = \min(\text{interval}(d_{j})) \). ............................... 18
2.9 Labeling of intervals starting/ending in \( \hat{d} \) in case 1b (when \( d_{j} = \min(\text{interval}(d_{j})) \)). ............................... 19
2.10 The relabeling of intervals beginning at a point in \( P_{d_{e}} \) and ending at \( \hat{d} \). ............................... 22
2.11 Necessary past points for \( d_{e} \) and necessary future points for \( d_{f} \), for every \( d_{j} \in P_{d_{e}} \). ............................... 23
2.12 The relabeling that allows one to fix the \( d \)-defect caused by the removal of \( d_{e} \). ............................... 23
2.13 The fulfilling step: addition of a new point and fulfillment of its past requests... 28
3.1 Unsatisfiability of \( \varphi_1 \) over a linear structure. ............................. 42
3.2 A branching structure satisfying \( \varphi_1 \). ........................................ 43
3.3 A tableau for the formula \( \varphi_2 = \langle A \rangle((A)\top \land (A)\top) \land (A)\bot \). ............................... 45
4.1 The semantics of \( (E) \) and \( (N) \) in SpPNL (left) and WSpPNL (right). ............ 52
4.2 Fixing defects of type 3. ............................................................... 57
5.1 An example of \( D_{\bot} \)-graph. ...................................................... 71
5.2 A fulfilling \( D_{\bot} \)-structure for the formula \( \varphi = (D)p \land (D)\neg p \land (D)q \). ............................................... 72
5.3 Turning a fulfilling \( D_{\bot} \)-structure into a model. ............................... 74
5.4 The tableau for \( (D)(D)p \land (D)\bot q \land (D)(\neg p \lor (D)\neg q) \). .................. 77
5.5 An example of \( D_{\bot} \)-graph. ...................................................... 80
5.6 (Part of) the tableau for \( \varphi = (D)p \land (D)q \land (D)(\neg p \land (D)q) \). .................. 90
5.7 An example of Lotrec tableau. ..................................................... 97
6.1 The four quadrants and the cone-shaped cardinal directions. ................... 100
6.2 A labeled rational plane satisfying \( \varphi_\constr < \). ................................. 100
6.3 Shading sequences (a), stripe expressions (b), and decompositions (c). ............................... 103
6.4 A \( \pi \)-labeled region delimiting (pseudo)interval-points. .......................... 111
6.5 The cone logic encoding of the \( (L)\psi \). ........................................... 112
6.6 The cone logic encoding of the \( (L)\neg \psi \). ........................................... 112
7.1 Correspondence between intervals and points of a discrete grid. ............... 118
7.2 Contraction \( \mathcal{G}' \) of a compass structure \( \mathcal{G} \). ............................ 119
7.3 A periodic compass structure with threshold \( \hat{y}_0 \), period \( \hat{y} \), and binding \( \hat{g} \). ............................... 122
7.4 A set \( 3m_{g_y}(f) \) of rows that fulfill all requests at row \( \hat{y}_0 \). .................... 123
7.5 Encoding of a tiling function. ....................................................... 124
7.6 Algorithm for the satisfiability problem over finite structures. ................. 127
7.7 Algorithm for the satisfiability problem over infinite structures. ............... 129
8.1 A compass structure. ............................................................... 132
8.2 Contraction of a compass structure. ............................................... 133
8.3 Decision algorithm for the satisfiability problem over finite linear orders. . . . . . 136
8.4 Encoding of part a computation of a lossy counter machine. . . . . . . . . . . 139

9.1 The current classification of HS fragments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 147
Introduction

Interval temporal logics are modal logics that allow one to represent and to reason about time intervals. It is well known that, on a linear order, one among thirteen different binary relations may hold between any pair of intervals, namely, “ends”, “during”, “begins”, “overlaps”, “meets”, “before”, together with their inverses, and the relation “equals” (the so-called Allen’s relations [3]). The whole set of Allen’s relations is depicted in Figure 1.1. Amongst the multitude of ternary relations over intervals there is one of particular importance, which corresponds to the binary operation of concatenation of meeting intervals. The logic CDT of such a ternary relation has been investigated by Venema in [61]. A systematic analysis of its fragments has been recently given by Hodkinson et al. [37]. Both CDT and its fragments have been shown to be highly undecidable. Since this dissertation focuses on decidable interval temporal logics, we do not consider these logics (as a matter of fact, in [32] we developed a sound and complete, but in general non-terminating, tableau system for CDT interpreted over branching interval structures).

![Diagram of Allen's relations between intervals.](image)

Figure 1.1: The thirteen Allen’s relations between intervals.

Allen’s relations give rise to respective unary modal operators, thus defining the modal logic of time intervals HS introduced by Halpern and Shoham in [36]. Some of these modal operators are actually definable in terms of others; in particular, if singleton intervals are included in the structure, it suffices to choose as basic modalities those corresponding to the relations “begins” B and “ends” E, and their transposes \( \bar{B} \), \( \bar{E} \). As CDT, HS turns out to be highly undecidable under very weak assumptions on the class of interval structures over which its formulas are interpreted.
In particular, undecidability holds for any class of interval structures over linear orders that contains at least one linear order with an infinite ascending or descending chain, thus including the natural time flows \( \mathbb{N}, \mathbb{Z}, \mathbb{Q}, \) and \( \mathbb{R} \). Undecidability of \( \text{HS} \) over finite structures directly follows from results in [40]. In [39], Lodray sharpens the undecidability of \( \text{HS} \) showing that the two modalities \( B, E \) suffice for undecidability over dense linear orders (in fact, the result applies to the class of all linear orders [34]). Even though \( \text{HS} \) is very natural and the meaning of its operators is quite intuitive, for a long time such sweeping undecidability results have discouraged the search for practical applications and further investigations in the field.

A renewed interest in interval temporal logics has been recently stimulated by the identification of some decidable fragments of \( \text{HS} \), whose decidability does not depend on simplifying semantic assumptions such as locality and homogeneity [34]. This is the case with the logic of the “begins/begun by’ (resp., “ends/ended by”) relations \( BB \) (resp., \( EE \)) [34] and the logic of temporal neighborhood \( \text{A}\text{A} \), whose modalities capture the “meets/met by” relations [14, 16, 33]. In particular, the decidability of \( \text{A}\text{A} \), as well as that of its future fragment \( A \), over various classes of linear orders, has been proved by Bresolin et al. by reducing its satisfiability problem to that of the two-variable fragment of first-order logic over the same classes of structures [13], whose decidability has been proved by Otto in [52]. Optimal tableau methods for \( A \), with respect to various classes of interval structures, can be found in [18, 20]. A decidable metric extension of \( A \) over the natural numbers has been proposed in [22]. A number of undecidable extensions of \( A \), and \( \text{A}\text{A} \), have been given in [10, 14].

In this dissertation, we show that the set of decidable fragments of \( \text{HS} \) contains various other meaningful elements. We first prove the decidability of \( \text{A}\text{A} \) with respect to other classes of interval structures. In particular, we consider the cases of discrete linear orders, of dense linear orders, and of branching interval structures over natural numbers. In addition, we study a 2-dimensional extension of \( A \) that can naturally interpreted as spatial generalization of \( A \). Next, we systematically investigate the interval logics of the sub-interval and super-interval relations over dense linear orders. Finally, we consider some combinations of the logics \( B, BB \) and \( A, \text{A}\text{A} \).

The dissertation is organized as follows.

In Chapter 2, we give a uniform model-theoretic proof of the decidability of the logic \( \text{A}\text{A} \), interpreted over different classes of linear orders, namely, all linear orders, dense linear orders, discrete linear orders, and integers (preliminary results in this direction can be found in [16, 18]. We prove a small model theorem for the case of all linear orders and then we show how to adapt it to the other cases. Sometimes such an adaptation is easy (dense case), sometimes it turns out to be rather involved (discrete case). We also prove that, in all the considered cases, the satisfiability problem is \text{NEXPTIME}-complete. Finally, we develop a sound and complete, optimal tableau system for all of them.

In Chapter 3 and Chapter 4, we take into consideration two particularly interesting special cases. In Chapter 3, we focus our attention on the satisfiability problem for the future fragment \( A \) interpreted in branching interval structures over natural numbers [17]. We prove that the problem is still decidable, but there is an increase in complexity (from \text{NEXPTIME} to \text{EXPSPACE}). We prove \text{EXPSPACE}-completeness of the problem and we develop a sound and complete, optimal tableau system for the logic. In Chapter 4, after a short account the existing spatial logics, we propose a spatial generalization of \( A \), called \( \text{WSPNL} \), and we prove that its satisfiability problem is decidable [19] (in fact, it remains \text{NEXPTIME}-complete). The proof is a nontrivial adaptation of the one in [16]. First, we prove a small model theorem for finite models and then, by exploiting very similar techniques, we determine a bound for the size of finite representations of a special class of infinite models (periodic models) that can always be extracted from (arbitrary) infinite ones.

Chapter 5 deals with interval logics of sub-interval structures over dense linear orders. There are three natural definitions of the sub-interval relation: reflexive \( \sqsubseteq \) (the current interval is a sub-interval of itself), proper \( \subset \) (sub-intervals can share one endpoint with the current interval), and strict \( \subset \) (both endpoints of the sub-intervals are strictly inside the current interval). The logic \( \mathcal{D}_\subset \) of reflexive sub-intervals has been studied first by van Benthem in [58], where it is proved that this
logic, if interpreted over dense linear orders, is equivalent to the standard modal logic S4. The connections between the logic of strict sub-intervals $D_{\subseteq}$ and the logic of Minkowski space-time have been explored by Shapirovsky and Shehtman in [56]. The authors proved that the following axiomatic system is sound and complete for $D_{\subseteq}$ over the class of dense orders:

- the K axiom:
- transitivity: $(\Diamond)p \rightarrow (\Diamond)(\Diamond)p$,
- seriality: $(\Diamond)\top$,
- 2-density: $(\Diamond)p_1 \land (\Diamond)p_2 \rightarrow (\Diamond)((\Diamond)p_1 \land (\Diamond)p_2)$.

By means of a suitable filtration technique, they also proved $D_{\subseteq}$ decidability and PSPACE completeness [55, 56].

We develop a sound, complete, and terminating tableau system for $D_{\subseteq}$. In order to prove soundness and completeness, we introduce a kind of finite pseudo-models for $D_{\subseteq}$, called $D_{\subseteq}$-structures, and we show that every formula satisfiable in $D_{\subseteq}$ is satisfiable in such pseudo-models with a bound on their dimension which depends on the size of the formula to be checked for satisfiability, thereby proving small-model property and decidability in PSPACE of $D_{\subseteq}$ (the result established earlier by Shapirovsky and Shehtman by means of filtration). Inter alia, we show that $D_{\subseteq}$ is also the logic of sub-interval structures over the interval $[0, 1]$ of the rational line.

Then, we extend our results to the case of the interval logic $D_{\subseteq}$ interpreted in dense interval structures with proper (irreflexive) sub-interval relation. $D_{\subseteq}$ differs substantially from $D_{\subseteq}$ and is much more difficult to analyze. The presence of the special families of beginning sub-intervals and ending sub-intervals of a given interval in a structure with proper sub-interval relation leads to considerable complications in the constructions of both pseudo-models and tableaux with respect to the case of $D_{\subseteq}$. For instance, the formula $((\Diamond)p \land (\Diamond)q) \rightarrow (\Diamond)((\Diamond)p \land (\Diamond)q)$ is valid in $D_{\subseteq}$, but not in $D_{\subseteq}$ (in $D_{\subseteq}$, $p$ and $q$ could be satisfied in respectively beginning and ending sub-intervals only). Furthermore, the formula

$$(\Diamond)(p \land (\Diamond)q) \lor (\Diamond)(p \land (\Diamond)\neg q) \land (\Diamond)\neg((\Diamond)(p \land (\Diamond)q) \land (\Diamond)(p \land (\Diamond)\neg q))$$

can only be satisfied in a $D_{\subseteq}$-structure, as it forces $p$ to be true at some beginning and at some ending sub-intervals, a requirement which cannot be imposed in $D_{\subseteq}$. Note, however, that $D_{\subseteq}$ can refer to beginning or ending sub-intervals, but it can differentiate between them only a finite number of times (using ad-hoc propositional variables). This is a subtle, but crucial, point: as shown by Lodaya [39], the interval logic BE with modalities respectively for beginning and ending sub-intervals is undecidable over the class of dense orders. However, the decidability problem for the logic $D$ over finite linear orders remains open and it probably represents the main obstacle to the complete classification of HS-fragment.

In Chapter 6, we introduce a novel spatial modal logic, called Cone Logic, which allows one to reason about cone-shaped directional relations between points in the rational plane. While the satisfiability problem for spatial modal logics with projection modalities turns out to be highly undecidable [42, 48], we prove that Cone Logic enjoys a decidable satisfiability problem (in fact, PSPACE-complete) by taking advantage of a suitable filtration technique. We also show that Cone Logic subsumes interesting interval temporal logics, such as the temporal logic of sub-interval/super-interval/begin/begun by/future/past ($BBDDLL$), thus generalizing previous results in the literature [12] and basically disproving a conjecture by Lodaya [39]. We also prove that the logic $BBDDLL$ is maximal with respect to decidability.

In Chapter 7, we introduce the product logic $ABB$ [47], obtained from the join of $BB$ and $A$ (the case of $AEE$ is fully symmetric), interpreted over the linear order $N$ of the natural numbers (or a finite prefix of it). The decidability of $BB$ can be proved by translating it into the point-based propositional temporal logic of linear time $LTL$ with temporal modalities $F$ (sometime in the future) and $P$ (sometime in the past), which has the finite (pseudo-)model property and is decidable, e.g., [30]. In general, such a reduction to point-based temporal logics does not work: formulas of interval temporal logics are evaluated over pairs of points and translate into binary relations. For instance, this is the case with $A$. Unlike the case of $BB$, when dealing with $A$ one
cannot abstract way from the left endpoint of intervals, as contradictory formulas may hold over intervals with the same right endpoint and a different left endpoint. $\mathbb{ABB}$ retains the simplicity of its constituents $\mathbb{BB}$ and $\mathbb{A}$, but it improves a lot on their expressive power (as we shall show, such an increase in expressiveness is achieved at the cost of an increase in complexity). First, it allows one to express assertions that may be true at certain intervals, but at no sub-interval of them, such as the conditions of accomplishment. Moreover, it makes it possible to easily encode the until operator of point-based temporal logic (this is possible neither with $\mathbb{BB}$ nor with $\mathbb{A}$). Finally, meaningful metric constraints about the length of intervals can be expressed in $\mathbb{ABB}$, that is, one can constrain an interval to be at least (resp., at most, exactly) $k$ points long. We prove the decidability of $\mathbb{ABB}$ interpreted over $\mathbb{N}$ by providing a small model theorem based on an original contraction method. To prove it, we take advantage of a natural (equivalent) interpretation of $\mathbb{ABB}$ formulas over grid-like structures based on a bijection between the set of intervals over $\mathbb{N}$ and (a suitable subset of) the set of points of the $\mathbb{N} \times \mathbb{N}$ grid. In addition, we prove that the satisfiability problem for $\mathbb{ABB}$ is EXSPACE-complete (that for $\mathbb{A}$ is NEXPTIME-complete). In the proof of hardness, we use a reduction from the exponential-corridor tiling problem.

In Chapter 8, we focus our attention on a proper extension of $\mathbb{ABB}$ that features four modal operators corresponding to the relations “meets”, “met by”, “begun by”, and “begins” of Allen’s interval algebra ($\mathbb{ABB}$ logic). $\mathbb{ABB}$ can be viewed as the join of the logics $\mathbb{BB}$ and $\mathbb{A}$. We prove that the satisfiability problem for $\mathbb{ABB}$, interpreted over finite linear orders, is decidable, but not primitive recursive (as a matter of fact, $\mathbb{ABB}$ turns out to be maximal with respect to decidability). Then, we show that it becomes undecidable when $\mathbb{ABB}$ is interpreted over classes of linear orders that contains at least one linear order with an infinitely ascending sequence, thus including the natural time flows $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$.

Finally, Chapter 9 provides an updated picture of the status of HS fragments with respect to decidability/undecidability, taking into account the contributions of the present dissertation, and it outlines the few remaining open problems in the field.

Chapter 2 is based on the papers “An optimal tableau-based decision algorithm for propositional neighborhood logic” (with Davide Bresolin and Angelo Montanari), published in the Proceedings of the 24th Annual Symposium on Theoretical Aspects of Computer Science, and “Optimal tableau for right propositional neighborhood logic over linear orders” (with Davide Bresolin, Angelo Montanari and Guido Sciavicco), published in the Proceedings of the 11th European Conference on Logics in Artificial Intelligence. Chapter 3 is based on the paper “Optimal tableau for right propositional neighborhood logic over trees” (with Davide Bresolin and Angelo Montanari), published in the Proceedings of the 15th International Symposium on Temporal Representation and Reasoning. Chapter 4 is based on the paper “Tableau-based system for spatial reasoning about directional relations” (with Davide Bresolin, Angelo Montanari and Guido Sciavicco), published in the Proceedings of the 18th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods. Chapter 5 is based on the paper “Tableau-based decision procedures for the logics of sub-interval structures over dense orderings” (with Davide Bresolin, Valentín Goranko and Angelo Montanari), published in the Journal of Logic and Computation. Chapter 6 is based on the paper “A decidable spatial logic with cone-shaped cardinal directions” (with Angelo Montanari and Gabriele Puppis), published in the Proceedings of the 18th Annual Conference of the EACSL. Chapter 7 is based on the paper “Decidability of the interval temporal logic $\mathbb{ABB}$ over the natural numbers” (with Angelo Montanari, Gabriele Puppis and Guido Sciavicco), published in the Proceedings of the 27th Annual Symposium on Theoretical Aspects of Computer Science. Chapter 8 is based on the paper “Maximal decidable fragments of Halpern and Shohami’s modal logic of intervals” (with Angelo Montanari and Gabriele Puppis), to be published in the Proceedings of the 37th International Colloquium on Automata, Languages and Programming.
Propositional Neighbourhood Logic
over linear orders

In this chapter, we focus our attention on the propositional logics of temporal neighborhood (PNL for short). PNL is the propositional fragment of Neighborhood Logic originally proposed in [25]. It can be viewed as the fragment of HS that features two modal operators ($A$ and $A'$), that respectively correspond to Allen’s relations *meets* and *met-by*. Basic logical properties of PNL have been investigated by Goranko et al. in [33]. The authors first introduce interval neighborhood frames and provide representation theorems for them; then, they develop complete axiomatic systems for PNL with respect various classes of interval neighborhood frames. The satisfiability problem for PNL has been addressed by Bresolin et al. in [14]. NEXPTIME-completeness with respect to the classes of all linearly ordered domains, well-ordered domains, finite linearly ordered domains, and natural numbers has been proved via a reduction to the satisfiability problem for the two-variable fragment of first-order logic for binary relational structures over ordered domains [52]. Despite these significant achievements, the problem of devising decision procedures of practical interest for PNL has been only partially solved. In [20], a tableau system for its future fragment RPNL, interpreted over the natural numbers, has been developed; such a system has been later extended to full PNL over the integers [16]. In this chapter, we develop a NEXPTIME tableau-based decision procedure for PNL interpreted over the class of all linear orders and then we show how to tailor it to the subclasses of dense linear orders and of discrete linear orders. NEXPTIME-hardness can be proved exactly as in [20], and thus the proposed procedures turn out to be optimal. From a technical point of view, the proposed tableau systems are quite different from that for RPNL over the natural numbers [20]. While models for RPNL formulas over the natural numbers can be generated by simply adding future points (possibly infinitely many) to a given partial model, the construction of a model for an PNL formula over an arbitrary (resp., dense, discrete) linearly ordered domain may require the addition of points (possibly infinitely many) in between existing ones. In the last part of the paper, we provide an optimal NEXPTIME tableau-based decision procedure for PNL over the integers that refines the one outlined in [16]. The chapter is organized as follows. In Section 2.1 we introduce syntax and semantics of PNL. Then, in Section 2.2 we introduce the notion of labeled interval structure (LIS) and we show that PNL satisfiability can be reduced to the existence of a fulfilling LIS. In Section 2.3 we prove the decidability of PNL over different classes of linear orders by a model-theoretic argument. Next, in Section 2.4, by taking advantage of the results given in the previous section, we develop optimal tableau-based decision procedures for PNL over the considered classes of linear orders. Finally, in Section 2.5 we prove the optimality of the proposed methods.

2.1 Propositional Neighborhood Logic

In this section, we give syntax and semantics of PNL interpreted over different classes of linear orders. Let $D$ be a set of points and $\mathcal{D} = (D, \prec)$ be a linear order on it. We say that $\mathcal{D}$ is *discrete* if any point having a successor (resp., predecessor) has an immediate one and that $\mathcal{D}$ is *dense* if
for every pair of points \( d_1 < d_1 \) there exists a point \( d_k \) such that \( d_1 < d_k < d_1 \). In the following, we will focus our attention on the representative classes of all linear orders, dense linear orders, and discrete linear orders, as well as to integers. In fact, similar results can be obtained for other classes of linear orders \([14]\).

An interval on \( \mathbb{D} \) is an ordered pair \([d_1, d_1]\) such that \( d_1, d_1 \in \mathbb{D} \) and \( d_1 < d_1 \). The set of all intervals over \( \mathbb{D} \) will be denoted by \( I(\mathbb{D}) \). The pair \((\mathbb{D}, I(\mathbb{D}))\) is called an interval structure. For every pair of intervals \([d_1, d_1]_I, [d_1', d_1']_I \in I(\mathbb{D})\), we say that \([d_1', d_1']_I\) is a right (resp., left) neighbor of \([d_1, d_1]_I\) if and only if \( d_1 = d_1' \) (resp., \( d_1' = d_1 \)).

The language of Propositional Neighborhood Logic (PNL for short) consists of a set \( \mathcal{AP} \) of propositional letters, the connectives \( \neg \) and \( \lor \), and the modal operators \((\mathcal{A})\) and \((\overline{\mathcal{A}})\). The other connectives, as well as the logical constants \( \top \) (true) and \( \bot \) (false), can be defined as usual. The formulae of PNL, denoted by \( \varphi, \psi, \ldots \), are recursively defined by the following grammar:

\[
\varphi ::= p | \neg \varphi | \varphi \lor \varphi | (\mathcal{A})\varphi | (\overline{\mathcal{A}})\varphi.
\]

We denote by \(|\varphi|\) the length of \( \varphi \), that is, the number of symbols in \( \varphi \) (in the following, we shall use \(|\cdot|\) to denote the cardinality of a set as well). Whenever there are no ambiguities, we call a PNL formula just a formula. A formula of the forms \((\mathcal{A})\varphi\), \(\neg(\mathcal{A})\varphi\), \((\overline{\mathcal{A}})\varphi\), or \(\neg(\overline{\mathcal{A}})\varphi\) is called a temporal formula (from now on, we identify \(\neg(\mathcal{A})\varphi\) with \(\neg I\varphi\) and \(\neg(\overline{\mathcal{A}})\varphi\) with \(\neg I\varphi\)).

A model for a PNL formula is a pair \( M = \langle (\mathbb{D}, I(\mathbb{D})), V \rangle \), where \( (\mathbb{D}, I(\mathbb{D})) \) is a strict interval structure and \( V : I(\mathbb{D}) \rightarrow 2^{\mathcal{AP}} \) is a valuation function assigning to every interval the set of propositional letters true over it. Given a model \( M = \langle (\mathbb{D}, I(\mathbb{D})), V \rangle \) and an interval \([d_1, d_1]_I \in I(\mathbb{D})\), the semantics of PNL is defined recursively by the satisfaction relation \(\models\) as follows:

- for every propositional letter \( p \in \mathcal{AP}, M, [d_1, d_1] \models p \) iff \( p \in V([d_1, d_1]) \);
- \( M, [d_1, d_1] \models \neg \psi \) iff \( M, [d_1, d_1] \not\models \psi \);
- \( M, [d_1, d_1] \models \psi_1 \lor \psi_2 \) iff \( M, [d_1, d_1] \models \psi_1 \) or \( M, [d_1, d_1] \models \psi_2 \);
- \( M, [d_1, d_1] \models (\overline{\mathcal{A}})\psi \) iff \( \exists d_k \in \mathbb{D} \) such that \( d_k > d_1 \) and \( M, [d_1, d_k] \not\models \psi \);
- \( M, [d_1, d_1] \models (\overline{\mathcal{A}})\psi \) iff \( \exists d_k \in \mathbb{D} \) such that \( d_k < d_1 \) and \( M, [d_k, d_1] \not\models \psi \).

We place ourselves in the most general (and difficult) setting where there are not constraints on the valuation function. As an example, given an interval \([d_1, d_1]_I\), it may happen that \( p \in V([d_1, d_1]) \) and \( p \not\in V([d_1', d_1']_I) \) for all intervals \([d_1', d_1']_I\) (strictly) contained in \([d_1, d_1]_I\).

It can be easily shown that PNL is expressive enough to distinguish between satisfiability over the class of all linear orders and the class of discrete (resp., dense) linear orders. To this end, it suffices to exhibit a formula that is satisfiable over the former and unsatisfiable over the latter.

Let \( \mathcal{G} \) be the universally-in-the-future operator defined as follows: \( \mathcal{G}\psi = \psi \land (\mathcal{A})\mathcal{G}\psi \) and let \( \text{seq}_p \) be a shorthand for \( p \rightarrow (\mathcal{A})p \). Consider the formula \( \text{AccPoints} = (\mathcal{A})p \land \mathcal{G}\text{seq}_p \land (\mathcal{A})\mathcal{G}\neg p \). We will show that \( \text{AccPoints} \) is unsatisfiable over \( \mathbb{N} \), while it is satisfiable whenever the temporal structure in which it is interpreted has at least one accumulation point, that is, a point which is right bound of an infinite (ascending) chain of points.

**Proposition 2.1.1.** The PNL formula \( \text{AccPoints} \) is satisfiable over the class of all linear orders, while it is not satisfiable over \( \mathbb{N} \).

**Proof.** We first show that \( \text{AccPoints} \) is not satisfiable over \( \mathbb{N} \). Suppose, by contradiction, that there exists an interpretation \( M \), based on \( \mathbb{N} \), such that \( M, [d_0, d_1] \models \text{AccPoints} \). From \( M, [d_0, d_1] \models (\mathcal{A})p \land \mathcal{G}\text{seq}_p \), it follows that there exists a sequence of points \( d_1 < d_1 < d_1 \ldots \) such that \( M, [d_1, d_1] \models p \) and \( M, [d_1, d_{i+1}] \models p \), for all \( i \geq 1 \). Moreover, from \( M, [d_0, d_1] \models (\mathcal{A})\mathcal{G}\neg p \), it follows that there exists a point \( d_1 \) such that \( M, [d_1, d_1] \models \neg \mathcal{G}\neg p \). Two cases may arise.

Case (1). Suppose \( d_1 < d_1 \). From \( M, [d_1, d_1] \models (\mathcal{A})\neg \mathcal{A}\neg p \), it follows that \( M, [d_1, d_1] \models \mathcal{A}\neg p \) and thus \( M, [d_1, d_1] \models \neg \mathcal{A}\neg p \). This allows us to conclude that both \( p \) and \( \neg p \) hold over \([d_1, d_2]_I \) as shown in Figure 2.1.

Case (2). Suppose \( d_1 < d_1 \). From \( M, [d_1, d_1] \models (\mathcal{A})\neg \mathcal{A}\neg p \), it follows that, for any point \( d_k > d_1 \), \( M, [d_1, d_k] \models \mathcal{A}\neg p \) and, for any point \( d_m > d_k \), \( M, [d_k, d_m] \models \neg p \). Since \( \text{AccPoints} \) is interpreted
2.2. Labeled Interval Structures and PNL Satisfiability

Let us consider now the class of all linearly ordered domains. A model satisfying AccPoints can be built as follows: we take an infinite sequence of points $d_1, d_2, d_3, \ldots$ such that $M, [d_i, d_{i+1}] \models p$, for every $i \geq 1$, and then we add an accumulation point $d_\omega$ greater than $d_i$, for every $i \geq 1$, such that $M, [d_1, d_\omega] \models \neg \neg p$. The definition of the valuation function can be easily completed without introducing any contradiction, thus showing that AccPoints is satisfiable (see Figure 2.3).

Figure 2.3: A model for AccPoints over the class of linearly ordered domains.

2.2 Labeled Interval Structures and PNL Satisfiability

In this section we introduce some preliminary notions and we establish some basic results on which our tableau method for PNL relies. Let $\varphi$ be a PNL formula to be checked for satisfiability and let $AP$ be the set of its propositional letters.

Definition 2.2.1. The closure $CL(\varphi)$ of $\varphi$ is the set of all subformulæ of $\varphi$ and of their negations, plus the formulæ $(A)\varphi$, $(\neg A)\varphi$, $(A)\neg \varphi$, and $(\neg A)\neg \varphi$ (we identify $\neg \neg \varphi$ with $\varphi$).

As it will become clear when we will consider the satisfiability problem for PNL over the integers, the presence of the formulæ $(A)\varphi, (A)\neg \varphi$ and of their negations in $CL(\varphi)$ allows us to easily guarantee that the removal process never deletes all intervals over which $\varphi$ holds.
Definition 2.2.2. The set of temporal formulae of $\varphi$ is the set $TF(\varphi) = \{ \langle A \rangle \psi, [A] \psi, (\overline{A}) \psi, [\overline{A}] \psi \in CL(\varphi) \}$. 

By induction on the structure of $\varphi$, we can easily prove that, for every formula $\varphi$, $|CL(\varphi)|$ is less than or equal to $2 \cdot |\varphi| + 1$, while $|TF(\varphi)|$ is less than or equal to $2 \cdot |\varphi|$. We are now ready to introduce the notion of $\varphi$-atom.

Definition 2.2.3. A $\varphi$-atom is a set $A \subseteq CL(\varphi)$ such that:

- for every $\psi \in CL(\varphi)$, $\psi \in A$ if and only if $\psi \not\in A$;
- for every $\psi_1 \lor \psi_2 \in CL(\varphi)$, $\psi_1 \lor \psi_2 \in A$ if and only if $\psi_1 \in A$ or $\psi_2 \in A$.

We denote the set of all $\varphi$-atoms by $A_{\varphi}$. By Definition 2.2.1 and Definition 2.2.3, it immediately follows that $|A_{\varphi}| \leq 2^{|\varphi| + 1}$.

Atoms are connected by the following binary relation.

Definition 2.2.4. Let $LR_\varphi$ be a relation such that for every pair of atoms $A_1, A_2 \in A_{\varphi}$, $A_1 LR_\varphi A_2$ if and only if (i) for every $[A] \psi \in CL(\varphi)$, if $[A] \psi \in A_1$ then $\psi \in A_2$ and (ii) for every $(\overline{A}) \psi \in CL(\varphi)$, if $(\overline{A}) \psi \in A_2$ then $\psi \in A_1$.

We now introduce a suitable labeling of interval structures based on $\varphi$-atoms.

Definition 2.2.5. A $\varphi$-labeled interval structure (LIS for short) is a pair $L = (\langle D, \llbracket D \rrbracket \rangle, \mathcal{L})$, where $\langle D, \llbracket D \rrbracket \rangle$ is an interval structure and $\mathcal{L} : \llbracket D \rrbracket \to A_\varphi$ is a labeling function such that, for every pair of neighboring intervals $[d_i, d_j], [d_i, d_k] \in \llbracket D \rrbracket$, $\mathcal{L}([d_i, d_j]) LR_\varphi \mathcal{L}([d_i, d_k])$.

In particular, we say that a LIS $L = (\langle D, \llbracket D \rrbracket \rangle, \mathcal{L})$ is discrete (resp., dense) if $D$ is discrete (resp., dense). If we interpret the labeling function as a valuation function, LISs represent candidate models for $\varphi$. The truth of formulae devoid of temporal operators indeed follows from the definition of $\varphi$-atom; moreover, universal temporal conditions, imposed by $[A]/[\overline{A}]$ operators, are forced by the relation $LR_\varphi$. However, to obtain a model for $\varphi$, we must also guarantee the satisfaction of existential temporal conditions, imposed by $\langle A \rangle/[\overline{A}]$ operators. To this end, we introduce the notion of fulfilling LIS.

Definition 2.2.6. A $\varphi$-labeled interval structure $L = (\langle D, \llbracket D \rrbracket \rangle, \mathcal{L})$ is fulfilling if and only if (i) for every temporal formula $\langle A \rangle \psi \in TF(\varphi)$ and every interval $[d_i, d_j] \in \llbracket D \rrbracket$, if $\langle A \rangle \psi \in \mathcal{L}([d_i, d_j])$, then there exists $d_k > d_i$ such that $\psi \in \mathcal{L}([d_k, d_j])$ and (ii) for every temporal formula $(\overline{A}) \psi \in TF(\varphi)$ and every interval $[d_i, d_j] \in \llbracket D \rrbracket$, if $(\overline{A}) \psi \in \mathcal{L}([d_i, d_j])$, then there exists $d_k < d_i$ such that $\psi \in \mathcal{L}([d_k, d_j])$.

The next theorem proves that for any given formula $\varphi$, the satisfiability of $\varphi$ is equivalent to the existence of a fulfilling LIS with an interval labeled by $\varphi$.

Theorem 2.2.7. A PNL formula $\varphi$ is satisfiable if and only if there exists a fulfilling LIS $L = (\langle D, \llbracket D \rrbracket \rangle, \mathcal{L})$ with $\varphi \in \mathcal{L}([d_i, d_j])$ for some $[d_i, d_j] \in \llbracket D \rrbracket$.

Proof. We first prove the left-to-right direction. Let $\varphi$ be a satisfiable PNL formula. Hence, there exist a model $M = (\langle D, \llbracket D \rrbracket \rangle, \mathcal{V})$ and an interval $[d_i, d_j] \in \llbracket D \rrbracket$ such that $M, [d_i, d_j] \models \varphi$. We show that $L = (\langle D, \llbracket D \rrbracket \rangle, \mathcal{L})$, where, for every $[d, d'] \in \llbracket D \rrbracket$, $\mathcal{L}([d, d']) = \{ \psi \in CL(\varphi) \mid M, [d, d'] \models \psi \}$, is a fulfilling LIS for $\varphi$. We first prove that for every $[d, d'] \in \llbracket D \rrbracket$, $\mathcal{L}([d, d'])$ is a $\varphi$-atom. For every $[d, d'] \in \llbracket D \rrbracket$ and $[d_i, d_j] \in \llbracket D \rrbracket$, we have that:

- by definition of $\models$, $M, [d, d'] \models \psi$ if and only if $M, [d_i, d_j] \not\models \neg \psi$ and thus, by definition of $\mathcal{L}$, $\psi \in \mathcal{L}([d, d'])$ if and only $\psi \not\in \mathcal{L}([d_i, d_j])$;

- by definition of $\models$, $M, [d, d'] \models \psi \lor \psi_2$ if and only if $M, [d_i, d_j] \models \psi_1$ or $M, [d, d'] \models \psi_2$ and thus, by definition of $\mathcal{L}$, $\psi \in \mathcal{L}([d, d'])$ if and only $\psi_1 \in \mathcal{L}([d_i, d_j])$ or $\psi_2 \in \mathcal{L}([d, d'])$.

Next, we prove that for every $d, d', d''$ in $D$, if $d < d' < d''$, then $\mathcal{L}([d, d']) LR_\varphi \mathcal{L}([d', d''])$. Suppose, by contradiction, that there exist $d, d', d''$ in $D$ such that $d < d' < d''$ and $\mathcal{L}([d, d']) LR_\varphi \mathcal{L}([d', d''])$ does not hold. By definition of $LR_\varphi$, this means that there exists $[A] \psi \in CL(\varphi)$ such
that \([A] \psi \in \mathcal{L}([d, d'])\) and \(\psi \not\in \mathcal{L}([d', d''])\) (and thus \(\neg \psi \in \mathcal{L}([d, d'])\)) or there exists \(\overline{A}\psi \in \mathcal{L}(\varphi)\) such that \(\overline{A}\psi \in \mathcal{L}([d', d''])\) and \(\psi \not\in \mathcal{L}([d, d'])\) (and thus \(\neg \psi \in \mathcal{L}([d', d''])\)). Let us consider the first case (the second one is completely symmetric, and thus omitted). By definition of \(\mathcal{L}\), we have that \(M, [d, d'] \models [A] \psi\) and \(M, [d', d''] \models \neg \psi\). By definition of \(\models\), \(M, [d, d'] \models [A] \psi\) implies that \(M, [d', d'] \models \psi\) for all \(d > d'\). Since, by hypothesis, \(d'' > d'\), we have that \(M, [d', d''] \models \psi\), which contradicts \(M, [d', d''] \models \neg \psi\). Finally, to prove that \(L\) is fulfilling, we must show that for every \([d, d'] \in \mathbb{I}(\mathbb{D})\) and every \(\langle A \rangle \psi \in \mathcal{L}([d, d'])\) (resp., \(\overline{A} \psi \in \mathcal{L}([d, d'])\)) there exists \(d'' > d'\) (resp., \(d'' < d\)) such that \(\psi \in \mathcal{L}([d', d''])\) (resp., \(\psi \not\in \mathcal{L}([d''', d])\)). Let \(\langle A \rangle \psi \in \mathcal{L}([d, d'])\) (the case in which \(\overline{A} \psi \in \mathcal{L}([d, d'])\) is completely symmetric, and thus omitted). By definition of \(\mathcal{L}\), we have that \(M, [d, d'] \models [A] \psi\). Since \(M\) is a model, we have that there exists \(d'' \in \mathbb{D} \) with \(d'' > d'\) for which \(M, [d', d''] \models \psi\) and, by definition of \(\mathcal{L}\), we have \(\psi \in \mathcal{L}([d', d''])\).

Let us consider now the right-to-left. Let \(L = \langle \mathbb{D}, \mathbb{I}(\mathbb{D}), \mathcal{L} \rangle\) a fulfilling LIS for a PNL formula \(\varphi\) we define \(M = M = \langle \mathbb{D}, \mathbb{I}(\mathbb{D}), \mathcal{V} \rangle\) with \(\forall [d, d'] \models \mathcal{L}([d, d']) \cap \mathcal{A}P\). For every \([d, d'] \in \mathbb{I}(\mathbb{D})\) we prove by induction for \(\psi \in \mathcal{CL}(\varphi)\) that \(M, [d, d'] \models \psi\) if and only if \(\psi \in \mathcal{L}([d, d'])\):

- \(\psi = p \in \mathcal{P} \), by construction \(p \in \mathcal{V}([d, d'])\) and only if \(p \in \mathcal{L}([d, d'])\);
- \(\psi = \neg \psi_1\), by induction \(M, [d, d'] \models \psi_1\) if and only if \(\psi_1 \in \mathcal{L}([d, d'])\). Since \(\mathcal{L}([d, d'])\) is an atom we have \(\psi_1 \in \mathcal{L}([d, d'])\) if and only if \(\neg \psi_1 \not\in \mathcal{L}([d, d'])\), finally we obtain \(M, [d, d'] \models \neg \psi_1\) if and only if \(\neg \psi_1 \not\in \mathcal{L}([d, d'])\);
- \(\psi = \psi_1 \lor \psi_2\) from inductive hypothesis we have that \(M, [d, d'] \models \psi_1\) if and only if \(\psi_1 \in \mathcal{L}([d, d'])\) and \(M, [d, d'] \models \psi_2\) if and only if \(\psi_2 \in \mathcal{L}([d, d'])\). By definition of \(\varphi\)-atom we have that \(\psi_1 \lor \psi_2 \in \mathcal{L}([d, d'])\) if and only if \(\psi_1 \in \mathcal{L}([d, d'])\) or \(\psi_2 \in \mathcal{L}([d, d'])\). Finally we obtain that \(M, [d, d'] \models \psi_1 \lor \psi_2\) if and only if \(M, [d, d'] \models \psi_1 \lor \psi_2\);
- \(\psi = \langle A \rangle \psi_1\) then since \(L\) is a LIS and is fulfilling we have that there exists \(d'' > d'\) with \(\psi_1 \in \mathcal{L}([d', d''])\) if and only if \(\langle A \rangle \psi_1 \in \mathcal{L}([d', d''])\) and only if \(\langle A \psi_1 \in \mathcal{L}([d', d''])\). From inductive hypothesis we have that \(\psi_1 \in \mathcal{L}([d', d''])\) if and only if \(M, [d', d''] \models \psi_1\). Summing up we obtain that \(\langle A \rangle \psi_1 \in \mathcal{L}([d', d''])\) if and only if \(M, [d, d'] \models \langle A \rangle \psi_1\).
- \(\psi = \overline{A} \psi_1\) this case is symmetric to the previous one.

To conclude the proof, it suffices to observe that since \(L\) is a fulfilling LIS there exists an interval \([d_i, d_j] \subseteq \mathbb{I}(\mathbb{D})\) for which \(\varphi \in \mathcal{L}([d_i, d_j])\) then we have \(M, [d_i, d_j] \models \varphi\).

The implication from left to right is straightforward; the opposite implication is proved by induction on the structure of the formula.

From now on, we say that a fulfilling LIS \(L = \langle \mathbb{D}, \mathbb{I}(\mathbb{D}), \mathcal{L} \rangle\) satisfies \(\varphi\) if and only if there exists an interval \([d_i, d_j] \subseteq \mathbb{I}(\mathbb{D})\) such that \(\varphi \in \mathcal{L}([d_i, d_j])\).

**Definition 2.2.8.** Given a LIS \(L = \langle \mathbb{D}, \mathbb{I}(\mathbb{D}), \mathcal{L} \rangle\) and a point \(d \in \mathbb{D}\), we define the set of future temporal requests of \(d\) as the set \(\text{REQ}^f(d) = \langle \mathbb{D} \rangle \mathcal{L} \models \mathcal{L}([d, d'])\) and the set of past temporal requests of \(d\) as the set \(\text{REQ}^p(d) = \langle \mathbb{D} \rangle \mathcal{L} \models \mathcal{L}([d', d])\). The set of temporal requests of \(d\) is defined as \(\text{REQ}(d) = \text{REQ}^f(d) \cup \text{REQ}^p(d)\).

**Definition 2.2.9.** Given a LIS \(L = \langle \mathbb{D}, \mathbb{I}(\mathbb{D}), \mathcal{L} \rangle\) for a PNL formula \(\varphi\), \(d \in \mathbb{D}\), and \(A \psi \in \text{REQ}^f(d)\) (resp., \(A \psi \in \text{REQ}^p(d)\)), we say that \(A \psi\) (resp., \(A \psi\)) is fulfilled for \(d\) in \(L\) if there exists \(d' \in \mathbb{D}\), with \(d' > d\) (resp., \(d' < d\)), such that \(\psi \in \mathcal{L}([d', d'])\) (resp., \(\psi \in \mathcal{L}([d', d'])\)). We say that \(d\) is fulfilled in \(L\) if for every \(A \psi \in \text{REQ}^f(d)\) (resp., \(A \psi \in \text{REQ}^p(d)\)) \(A \psi\) (resp., \(A \psi\)) is fulfilled for \(d\) in \(L\).

**Definition 2.2.10.** Given a LIS \(L = \langle \mathbb{D}, \mathbb{I}(\mathbb{D}), \mathcal{L} \rangle\) for a PNL formula \(\varphi\) and \(d \in \mathbb{D}\), we say that \(d\) (resp., \(\text{REQ}(d)\)) is unique in \(L\) if for every \(d \in \mathbb{D}\), with \(d \neq d\), \(\text{REQ}(d) \neq \text{REQ}(d)\).

Given a formula \(\varphi\), we denote by \(\text{REQ}_\varphi\) the set of all possible sets of requests. It is not difficult to show that \(|\text{REQ}_\varphi|\) is equal to \(2^{\text{TR}(\varphi)}\).

**Definition 2.2.11.** Given a LIS \(L = \langle \mathbb{D}, \mathbb{I}(\mathbb{D}), \mathcal{L} \rangle\), \(D' \subseteq \mathbb{D}\), and \(R \subseteq \text{REQ}(d)\), we say that \(R\) occurs \(n\) times in \(D'\) if and only if there exist exactly \(n\) distinct points \(d_{i_1}, \ldots, d_{i_n} \in D'\) such that \(\text{REQ}(d_{i_j}) = R\), for all \(1 \leq j \leq n\).
2.3 Decidability of PNL over different classes of linear orders

In this section, we first prove that the satisfiability problem for PNL over the classes of all, dense, and discrete linear orders is decidable; then, we prove its decidability over the integers.

**Definition 2.3.1.** Let \( \varphi \) be a PNL formula, \( A \) be a \( \varphi \)-atom, and \( S_1, S_2 \subseteq \text{TF}(\varphi) \). The triplet \( \langle S_1, A, S_2 \rangle \) is an interval-tuple if and only if for every \( [A] \psi \in S_1, \psi \in A, \) and for every \( [\overline{\mathcal{A}}] \psi \in S_2, \psi \in A. \)

**Proposition 2.3.2.** Let \( L = \langle \mathcal{D}, [\mathcal{D}], \mathcal{L} \rangle \) be a LIS for a PNL formula \( \varphi \). For every \( d, d' \in \mathcal{D} \), the triplet \( \langle \text{REQ}^d(d), \mathcal{L}([d, d']), \text{REQ}^d(d') \rangle \) is an interval-tuple.

*Proof.* It easily follows from Definition 2.2.4, Definition 2.2.5, and Definition 2.2.8. \( \square \)

**Definition 2.3.3.** Let \( \langle R, A, R' \rangle \) be an interval-tuple. We say that \( \langle R, A, R' \rangle \) occurs in \( L \) if there exists \( [d, d'] \in [\mathcal{D}] \) such that \( \mathcal{L}([d, d']) = \langle R, A, R' \rangle \). If \( \langle R, A, R' \rangle \) occurs in \( L \) and there exists \( [d, d'] \) such that \( \mathcal{L}([d, d']) = \langle R, A, R' \rangle \) and both \( d \) and \( d' \) are fulfilled in \( L \), then we say that \( \langle R, A, R' \rangle \) is fulfilled in \( L \) (via \( [d, d'] \)).

**Definition 2.3.4.** Given a finite LIS \( L = \langle \mathcal{D}, [\mathcal{D}], \mathcal{L} \rangle \) for a PNL formula \( \varphi \), we say that \( L \) is a pseudo-model for \( \varphi \) if every interval-tuple \( \langle R, A, R' \rangle \) that occurs in \( L \) is fulfilled.

From the fact that all interval-tuples are fulfilled in \( L \), that is, \( L \) is a pseudo-model for \( \varphi \), it does not follow that \( L \) is fulfilling, since in \( L \) there can be multiple occurrences of the same interval-tuple, associated with different intervals. Thus, to turn a pseudo-model into a fulfilling LIS (for \( \varphi \)) some additional effort is needed. The next definition introduces an important ingredient of such a process.

**Definition 2.3.5.** Let \( \varphi \) be a PNL formula and \( L = \langle \mathcal{D}, [\mathcal{D}], \mathcal{L} \rangle \) be a fulfilling LIS that satisfies it. For any \( d \in \mathcal{D} \), we say that:

- **(future)** a set \( \text{ES}^d_\varphi \subseteq \mathcal{D} \) is a future essential set for \( d \) if (i) for every \( \langle A \rangle \psi \in \text{REQ}^d(d) \), there exists \( d' \in \text{ES}^d_\varphi \) such that \( \psi \in \mathcal{L}([d, d']) \) (fulfilling condition) and (ii) for every \( d' \in \text{ES}^d_\varphi \) there exists a formula \( \langle A \rangle \psi \in \text{REQ}^d(d) \) such that, for every \( d'' \in (\text{ES}^d_\varphi \setminus \{d'\}), \neg \psi \in \mathcal{L}([d, d'']) \) (minimality);

- **(past)** a set \( \text{ES}^d_\varphi \subseteq \mathcal{D} \) is a past essential set for \( d \) if (i) for every \( \langle \overline{A} \rangle \psi \in \text{REQ}^d(d) \), there exists \( d' \in \text{ES}^d_\varphi \) such that \( \psi \in \mathcal{L}([d', d]) \) (fulfilling condition) and (ii) for every \( d' \in \text{ES}^d_\varphi \) there exists a formula \( \langle \overline{A} \rangle \psi \in \text{REQ}^d(d) \) such that, for every \( d'' \in (\text{ES}^d_\varphi \setminus \{d'\}), \neg \psi \in \mathcal{L}([d'', d]) \) (minimality).

From Definition 2.3.5 it follows that for all \( d' \in \text{ES}^d_\varphi \) (resp., \( d' \in \text{ES}^d_\varphi \)), there exists at least one formula \( \psi \) that belongs to \( \mathcal{L}([d, d']) \) (resp., \( \mathcal{L}([d', d]) \) only. On the contrary, we cannot exclude the existence of formulas \( \psi \) that belong to the labeling of more than one interval \([d, d']\) (resp., \([d', d]\)), with \( d' \in \text{ES}^d_\varphi \) (resp., \( d' \in \text{ES}^d_\varphi \)).

**Definition 2.3.6.** Given a PNL formula \( \varphi \), a fulfilling LIS \( L = \langle \mathcal{D}, [\mathcal{D}], \mathcal{L} \rangle \) that satisfies it, and \( d \in \mathcal{D} \), we define the sets:

- **Future**\(^d\)(\( d \)) = \{\text{REQ}^d(d') \mid d' > d\};

- **Past**\(^d\)(\( d \)) = \{\text{REQ}^d(d') \mid d' < d\}. 

2.3. Decidability of PNL over different classes of linear orderings

We first prove that the satisfiability problem for PNL over the class of all linear orders is decidable.

**Lemma 2.3.7.** Given a pseudo-model \( L = (\langle \mathbb{D}, \mathbb{L} \rangle, \mathcal{L}) \) for a PNL formula \( \varphi \), there exists a fulfilling LIS \( L' \) that satisfies \( \varphi \).

**Proof.** We show how to obtain a fulfilling LIS \( L' \) starting from the pseudo-model \( L \) as the limit of a possibly infinite sequence of pseudo-models \( L_0 = L, L_1, L_2, \ldots \). In the following, we describe how to obtain the pseudo-model \( L_{i+1} \) from the pseudo-model \( L_i \), for any \( i \geq 0 \).

Let \( Q_i \) be the queue of all points \( d \in D_i \) that must be checked for fulfillment (for \( i = 0 \), \( Q_i \) consists of all and only the points \( d \in D \) such that \( d \) is not fulfilled in \( L \)). If \( Q_i \) is empty, then we put \( L' = L_{i+1} \). Otherwise, \( L_{i+1} \) is built as follows. Let \( d \) be the first element of the queue \( Q_i \). If \( d \) is fulfilled, we remove it from the queue and put \( L_{i+1} = L_i \) (as we will later see, every point in the queue is not fulfilled at insertion time; however, it may happen that subsequent expansions of the domain make it fulfilled before the time at which it is taken into consideration). Otherwise, either there exists \( \langle A \rangle \varphi \in \text{REQL}_i^i(d) \) which is not fulfilled, or there exists \( \langle \overline{A} \rangle \varphi \in \text{REQL}_i^i(d) \) which is not fulfilled, or both.

Suppose that there exists a \( \langle A \rangle \)-formula in \( \text{REQL}_i^i(d) \) which is not fulfilled. Two cases may arise:

1) There exists \( d' > d \) such that \( \text{REQL}_i^i(d') = \text{REQL}_i^i(d) \) and \( d' \) is fulfilled. Let \( \mathcal{E}^d = \{d_1, \ldots, d_k\} \). For \( j = 1, \ldots, k \), we proceed as follows:

   a) If \( d_j \) is unique, then we put \( L_{i+1}([d, d_j]) = L_i([d', d_j]) \). We prove that such a replacement does not introduce new defects. Since the interval-tuple \( (\text{REQL}_i^i(d), L_i([d, d_j])) \), \( \text{REQL}_i^i(d_j) \) is fulfilled in \( L_i \), there exists an interval \( [d'', d'''] \) such that \( \langle \text{REQL}_i^i(d), L_i([d, d_j]) \rangle \) is fulfilled in \( L_i \) via \( [d'', d'''] \). Since \( d_j \) is unique, \( d''' = d_j \). However, since \( d \) is not fulfilled in \( L_i \), \( d'' \neq d \), and thus we can safely update the labeling of \( [d, d_j] \). This case is depicted in Figure 2.4.

   b) If \( d_j \) is not unique, then there exists \( \overline{d} \neq d_j \), with \( \text{REQL}_i^i(\overline{d}) = \text{REQL}_i^i(d_j) \). In such a case, we introduce a new point \( \overline{d} \) immediately after \( d_j \) with the same requests as \( d_j \), that is, we put \( D_{i+1} = D_i \cup \{\overline{d}\} \), with \( d_j < \overline{d} \) and for all \( d \), if \( d > d_j \), then \( d > \overline{d} \), and we force \( \text{REQL}_i^{i+1}(\overline{d}) \) to be equal to \( \text{REQL}_i^i(d_j) \). To this end, for every \( d'' \), with \( d'' \neq d \) and \( d'' \neq d_j \), and \( d'' \neq d' \), we put \( L_{i+1}([d'', \overline{d}]) = L_i([d'', d_j]) \) (when \( d'' < \overline{d} \)) and \( L_{i+1}([\overline{d}, d''']) = L_i([d_j, d''']) \) (when \( d''' > \overline{d} \)). Moreover, we put \( L_{i+1}([\overline{d}, d]) = L_i([d', d_j]) \) and \( L_{i+1}([d', \overline{d}]) = L_i([d, d_j]) \), as depicted in Figure 2.5. In such a way,
Figure 2.5: Labeling of the intervals \([d, \hat{d}]\) and \([d', \hat{d}]\) in Case 1b.

d satisfies over \([d, \hat{d}]\) the request that \(d'\) satisfies over \([d', d_j]\). At the same time, we guarantee that \(\hat{d}\) satisfies the same past requests that \(d_j\) satisfies: \(\hat{d}\) satisfies over \([d, \hat{d}]\) (resp., \([d', \hat{d}]\)) the request that \(d_j\) satisfies over \([d', d_j]\) (resp., \([d, d_j]\)) and it satisfies the remaining past requests over intervals that start at the same point where the intervals over which \(d_j\) satisfies them start. Finally, if \(\overline{a} > d_1\), we put \(L_{i+1}([d_j, d]) = L_i([d_j, \overline{a}])\), \(L_{i+1}([d, d_j]) = L_i([\overline{a}, d_j])\) otherwise. For all the remaining pairs \(d_i, d_j\) the labeling remains unchanged, that is, \(L_{i+1}([d_i, d_j]) = L_i([d_i, d_j])\). Now, we observe that, by definition of \(L_{i+1}\), if \(d_j\) is fulfilled (in \(L_i\)), then \(\hat{d}\) is fulfilled (in \(L_{i+1}\)), while if \(d_j\) is not fulfilled (in \(L_i\)), being \(\hat{d}\) fulfilled or not (in \(L_{i+1}\)) depends on the labeling of the interval \([d_i, \hat{d}]\). If \(\hat{d}\) is not fulfilled (in \(L_{i+1}\)), we insert it into the queue \(Q_{t+1}\).

2) For every \(d' > d\), with \(\text{REQ}_i(d') = \text{REQ}_i^+(d)\), \(d'\) is not fulfilled. Let \(d' < d\) such that \(\text{REQ}_i(d') = \text{REQ}_i^+(d)\), \(d'\) is fulfilled, and, for every \(d' < d'' < d\), if \(\text{REQ}_i^+(d'') = \text{REQ}_i^+(d)\), then \(d''\) is not fulfilled.

As a preliminary step, we prove that \(\text{Past}_i(d') = \text{Past}_i^+(d)\). Suppose, by contradiction, that there exists \(d' < d'' < d\) such that \(\text{REQ}_i^+(d'') \notin \text{Past}_i^+(d')\). Since \(L^i\) is a pseudo-model, there exist \(\overline{a}, \overline{a}' \in D_i\) such that the interval-tuple \((\text{REQ}_i(d''), L_i([d'', d]), \text{REQ}_i^+(d))\) is fulfilled in \(L^i\) via \([\overline{a}, \overline{a}]\). By definition, both \(\overline{a}\) and \(\overline{a}'\) are fulfilled; moreover, \(\text{REQ}_i^+(\overline{a}) = \text{REQ}_i^+(d'')\), \(\text{REQ}_i^+(\overline{a}') = \text{REQ}_i^+(d)\), and \(L_i([\overline{a}, \overline{a}']) = L_i([d'', d])\). Since \(\text{REQ}_i^+(d'') \notin \text{Past}_i^+(d')\), \(d'' < \overline{a}\) and thus \(d'' < \overline{a}'\). However, since \(d'\) is the largest fulfilled element in \(D_i\) with \(\text{REQ}_i^+(d') = \text{REQ}_i^+(d)\), \(\overline{a}'\) cannot be greater than \(d'\) (contradiction). Hence, \(\text{Past}_i^+(d') = \text{Past}_i^+(d)\).

Let \(E_S d' = \{d_1, \ldots, d_k\}\). For every \(j = 1, \ldots, k\), we proceed as follows:

a) If \(d_j\) is unique, then \(d_j > d\), since \(\text{Past}_i(d') = \text{Past}_i^+(d)\). We proceed as in Case 1a.

b) If \(d_j\) is not unique and \(d_j > d\), then we proceed as in Case 1b.

c) If \(d_j\) is not unique and \(d' < d_j < d\), then we introduce a new point \(\hat{d}\) immediately after \(d\) with the same requests as \(d_j\), that is, we put \(D_{i+1} = D_i \cup \{\hat{d}\}\), with \(d < \hat{d}\) and for all \(\hat{d}\), if \(d > d\), then \(\hat{d} > d\), and we force \(\text{REQ}_i^+([\hat{d}])\) to be equal to \(\text{REQ}_i^+(d)\). To this end, for every \(d''\), with \(d'' < d_j\) (resp., \(d'' > d\)), we put \(L_{i+1}([d'', \hat{d}]) = L_i([d'', d_j])\) (resp., \(L_{i+1}([\hat{d}, d'']) = L_i([d_j, d''])\)). Since \(\text{Past}_i(d') = \text{Past}_i^+(d)\), for all \(d_j < d'' < d\) there exists \(d'' < d'\) such that \(\text{REQ}_i(d'') = \text{REQ}_i^+(d'')\). Hence, we put \(L_{i+1}([d'', \hat{d}]) = \text{REQ}_i^+(d'')\).
2.3. Decidability of PNL over different classes of linear orderings

\[ L_i([d'', d_i]) \]. Moreover, we put \( L_{i+1}( [d, \hat{d}] ) = L_i([d', d_i]) \). Finally, if \( \hat{d} \) is not fulfilled, we insert it into the queue \( Q_{i+1} \). This case is depicted in Figure 2.6.

The case in which there exists a \( (\mathcal{A}) \)-formula in \( \text{REQ}^L(d) \) which is not fulfilled is completely symmetric, and thus its description is omitted. This concludes the construction of \( L_{i+1} \). Since \( d \) is fulfilled in \( L_{i+1} \), it can be safely removed from the queue.

Since all points which are fulfilled in \( L_i \) remain fulfilled in \( L_{i+1} \), it is immediate to conclude that \( L_{i+1} \) is a pseudo-model. Moreover, it can be easily checked that the proposed construction does not remove any point, but it can introduce new ones, possibly infinitely many. However, the choice of a queue to manage points which are (possibly) not fulfilled guarantees that the defects of each of them sooner or later will be fixed. To complete the proof, it suffices to show that the fulfilling LIS \( L' \) for \( \varphi \) we were looking for is the limit of this (possibly infinite) construction. Let \( L_i^- \) be equal to \( L_i \) devoid of the labeling of all intervals consisting of a (non-unique) point in \( Q_i \) and a unique point (in \( D_i \setminus Q_i \)). We define \( L' \) as the limit of \( \cup_{i \geq 0} L_i^- \) when \( t \) tends to infinity (if \( Q_i \) turns out to be empty for some \( i \), then \( L' \) is simply equal to \( \cup_{i \geq 0} L_i^- (= L_0) \)). It is trivial to check that for every pair \( D_i \), \( D_{i+1} \), \( D_i \subseteq D_{i+1} \). To prove that for every pair \( L_i^-, L_{i+1} \), it holds that \( L_i^- \subseteq L_{i+1} \), we observe that: (i) the labeling of intervals whose endpoints are both non-unique points (resp., unique points) never changes, that is, it is fixed once and for all, and (ii) for every pair of point \( d, d' \in D_i \setminus Q_i \) such that \( d \) is a non-unique point and \( d' \) is a unique one, if \( d < d' \) (resp., \( d' < d \)), then \( L_i([d, d']) = L_{i+1}([d', d]) \) (resp., \( L_{i+1}([d', d]) = L_i([d', d]) \)) for all \( j \geq i \), that is, the labeling of an interval consisting of a non-unique point and a unique one may possibly change when the non-unique point is removed from the queue and then it remains unchanged forever (notice that non-unique points which are fulfilled from the beginning never change “their labeling”). Finally, to prove that all points are fulfilled in \( \cup_{i \geq 0} L_i^- \), it is sufficient to observe that: (i) all unique points belong to \( D_0 \) and are fulfilled in the restriction of \( L_0 \) to \( D_0 \setminus Q_0 \) (and thus in \( L_0^- \)), and (ii) for every \( i \geq 0 \), all points in \( D_i \setminus Q_i \) are fulfilled in \( L_i^- \) and the first element of \( Q_i \) may be not fulfilled in \( L_i \) (and thus in \( L_i^- \)), but it is fulfilled in \( L_{i+1}^- \). Every point is indeed either directly inserted into \( D_i \setminus Q_i \) or added to \( Q_i \) (and thus it becomes the first element of \( Q_j \) for some \( j > i \)) for some \( i \geq 0 \).

\[ \square \]

**Lemma 2.3.8.** Given a PNL formula \( \varphi \) and a fulfilling LIS \( L = (\langle D,I(D) \rangle, \mathcal{L}) \) that satisfies it, there exists a pseudo-model \( L' \) for \( \varphi \), with \(|D'| \leq 2 \cdot |\varphi| \cdot 2^{3 |\varphi| + 1} \).
Proof. Let $\Gamma(L) = \{\langle R, A, R' \rangle \mid \langle R, A, R' \rangle \text{ appears in } L\}$ and let $D'$ be a minimal subset of $D$ such that for every $\langle R, A, R' \rangle \in \Gamma(L)$, there exist two points $d, d' \in D'$ such that $\text{REQ}^i(d) = R$, $\text{REQ}^i(d') = R'$, $\mathcal{L}(\langle d, d' \rangle) = A$, and $d, d'$ are fulfilled in $D'$, that is, $\mathcal{E}^i_d \cup \mathcal{E}^i_{d'} \subseteq \mathcal{E}^i_{d'} \subseteq D'$. For every $d, d' \in D'$, with $d < d'$, we define $\mathcal{L}'(\langle d, d' \rangle) = \mathcal{L}(\langle d, d' \rangle)$. It is easy to prove that $L'$ is a pseudo-model for $\varphi$. As for the size of $D'$, the number of distinct intervals in $L'$ is at most $2^{d(|\varphi|+1)}$ (the number of atoms is $2^{d(|\varphi|+1)}$ and the number of sets of requests is $2^{d(|\varphi|)}$) and, for every interval-tuple, at most $2 \cdot |\varphi|$ points must be added. Hence, $|D'| \leq 2 \cdot |\varphi| \cdot 2^{d(|\varphi|+1)}$. \qed

The decidability of PNL over the class of all linear orders immediately follows.

**Theorem 2.3.9.** The satisfiability problem for PNL over the class of all linear orders is decidable.

Proof. The thesis follows from Lemma 2.3.7, Lemma 2.3.8, and Theorem 2.2.7. \qed

### 2.3.2 Decidability of PNL over the class of dense linear orders

We now show how to tailor the decidability proof for PNL satisfiability over the class of all linear orders to prove its decidability over the subclass of dense linear orders. As a matter of fact, the proof is close to the one for the general case. The only additional requirement is that the domain must be dense.

**Definition 2.3.10.** Let $L = \langle D, \ll [D] \rangle, \mathcal{L} \rangle$ be a pseudo-model for a PNL formula $\varphi$ and $d \in D$. We say that $d$ is covered if either $d$ is not unique or ($d$ is unique and) both its immediate predecessor (if any) and successor (if any) are not unique. We say that $L$ is covered if every $d \in D$ is covered.

**Lemma 2.3.11.** Let $\varphi$ be a PNL formula. From any covered pseudo-model $L$ for $\varphi$, we can generate a dense fulfilling LIS $L'$ that satisfies $\varphi$.

Proof. We basically proceed as in the proof of Lemma 2.3.7, but for the insertion of a new point in between any pair of existing points of the current pseudo-model at the end of any iteration step. First of all, we observe that if $L_i$ is a covered pseudo-model, then $L_{i+1}$ is a covered pseudo-model as well, since all added points are not unique. Given that the initial pseudo-model $L$ is covered by hypothesis, this implies that $L_i$ is covered for all $i = 0, 1, \ldots$. Now, let $L_{i+1}$ be the pseudo-model obtained from $L_i$ at the $i+1$-th iteration of the construction step described in the proof of Lemma 2.3.7 and let $d_0 < d_1 < \ldots < d_k$ be the elements of the resulting domain $D_{i+1}$. For any pair of consecutive points $d_i, d_{i+1}$, with $0 \leq j < k$, we proceed as follows. First, we observe that at least one of them (possibly both) is not unique. Without any loss of generality, we may assume $d_i$ to be such a point. We add a new point $d$ in between $d_i$ and $d_{i+1}$ and we force $\text{REQ}^{i+1}(d)$ to be equal to $\text{REQ}^{i+1}(d_i)$. To this end, for every $d' < d_i$, we put $\mathcal{L}_{i+1}(\langle d', d_i \rangle) = \mathcal{L}_{i+1}(\langle d', d_i \rangle)$ and, for every $d' > d_i$, we put $\mathcal{L}_{i+1}(\langle d_i, d' \rangle) = \mathcal{L}_{i+1}(\langle d_i, d' \rangle)$. Finally, since $d_i$ is not unique, there exists at least one point $d$ such that $\text{REQ}^{i+1}(d) = \text{REQ}^{i+1}(d_i)$. If $d > d_i$, we put $\mathcal{L}_{i+1}(\langle d_i, d \rangle) = \mathcal{L}_{i+1}(\langle d_i, d \rangle)$, $\mathcal{L}_{i+1}(\langle d, d_i \rangle) = \mathcal{L}_{i+1}(\langle d, d_i \rangle)$ otherwise.

As in the case of Lemma 2.3.7, to conclude the proof it suffices to take the limit of this (always infinite) construction as the fulfilling dense LIS $L$ for $\varphi$ we were looking for. \qed

**Lemma 2.3.12.** Let $\varphi$ be a PNL formula. From any dense fulfilling LIS $L$ that satisfies $\varphi$, we can extract a covered pseudo-model $L'$ for $\varphi$, with $|D'| \leq 4 \cdot |\varphi| \cdot 2^{d(|\varphi|+1)}$.

Proof. As in the proof of Lemma 2.3.8, we can extract a pseudo-model $L'$ from the given dense fulfilling LIS $L$. However, such a pseudo-model is not necessarily covered, as we cannot exclude the presence of consecutive pairs of unique points in its domain. Nevertheless, since $L$ is dense, there exists at least one point in $D$ (in fact, infinitely many) in between any pair of consecutive points in the domain of $L'$ which is not unique (in $L'$). We add this point to the domain of $L'$ and
2.3. Decidability of PNL over different classes of linear orderings

extend its labeling accordingly. Once we have fixed all such defects, we obtain a covered pseudo-model \( L' \) for \( \varphi \). As for the size of the domain of \( L' \), it is immediate to prove that such a repair generates a pseudo-model whose domain is at most two times larger than the original one. The thesis immediately follows.

The decidability of PNL over the class of dense linear orders immediately follows.

**Theorem 2.3.13.** The satisfiability problem for PNL over the class of dense linear orders is decidable.

**Proof.** The thesis follows from Lemma 2.3.11, Lemma 2.3.12, and Theorem 2.2.7.

### 2.3.3 Decidability of PNL over the class of discrete linear orders

In this section, we focus our attention on the subclass of discrete linear orders and we show that the proof for the general case can actually be adapted to deal with it. Unlike the case of dense linear orders, however, such an adaptation requires a significant amount of additional work.

To start with, we introduce the notion of safe pseudo-model that will play a significant role in the decidability proof.

**Definition 2.3.14.** Let \( L = \langle (\mathcal{D}, \mathcal{I}(\mathcal{D})), \mathcal{L} \rangle \) be a pseudo-model for a PNL formula \( \varphi \) and \( d \in \mathcal{D} \). We say that \( d \) is safe if either \( d \) is not unique or (\( d \) is unique and) both its immediate predecessor (if any) and successor (if any) are fulfilled. We say that \( L \) is safe if every \( d \in \mathcal{D} \) is safe.

The following lemma provides a useful way of structuring (the domain of) a safe pseudo-model \( L = \langle (\mathcal{D}, \mathcal{I}(\mathcal{D})), \mathcal{L} \rangle \), provided that it does not contain isolated non-unique points, that is, for any non-unique point \( d \in \mathcal{D} \), if \( d \) is neither the least nor the greatest point in \( \mathcal{D} \), then either the immediate predecessor of \( d \), or the immediate successor of \( d \), or both are non-unique points as well.

**Lemma 2.3.15.** Let \( \varphi \) be a PNL formula and \( L \) be a safe pseudo-model for \( \varphi \). If there are not isolated non-unique points in (the domain \( \mathcal{D} \) of) \( L \), then we can partition \( \mathcal{D} \) into a sequence of intervals \( I_1, \ldots, I_n \) such that (i) for \( 1 \leq i, j \leq n \), with \( i < j \), if \( r \in I_i \) and \( s \in I_j \), then \( r < s \) (\( I_1 < I_j \) for short), and (ii) for \( i = 1, \ldots, n \), \( I_i = \{d_{i_1}, \ldots, d_{i_k}\} \), with \( d_{i_1} < \ldots < d_{i_k} \), has one of the following two forms:

1. if there are not unique points in \( I_i \), then all points in \( I_i \) have the same requests, that is, \( \text{REQ}^L(d_{i_1}) = \ldots = \text{REQ}^L(d_{i_{k_i}}) \);
2. if there exists at least one unique point in \( I_i \), then all points in \( I_i \) are fulfilled and there exist \( 1 \leq r \leq s \leq k_i \) such that \( d_{i_1}, \ldots, d_{i_{r-1}} \) are non-unique points, with \( \text{REQ}^L(d_{i_1}) = \text{REQ}^L(d_{i_{r-1}}), d_{i_r}, \ldots, d_{i_{s}} \) are unique points, and \( d_{i_{s+1}}, \ldots, d_{i_{k_i}} \) are non-unique points, with \( \text{REQ}^L(d_{i_{s+1}}) = \text{REQ}^L(d_{i_{k_i}}) \); in addition, we require that:
   - if \( n = 1 \), then \( r \geq 1 \) and \( s \leq k_1 \) (no further restriction);
   - if \( n > 1 \) and \( i = 1 \), then \( r \geq 1 \), but \( s < k_1 \);
   - if \( n > 1 \) and \( i = n \), then \( s \leq k_n \), but \( r > 1 \);
   - if \( n > 1 \) and \( 1 < i < n \), then both \( r > 1 \) and \( s < k_i \).

**Proof.** The existence of such a partition easily follows from the safety condition (both the immediate predecessor and successor of a unique point are fulfilled) and the lack of isolated non-unique points (it never happens that there exists a single non-unique point between two unique ones).
Let $\mathcal{P}$ be a partition of (the domain of) a pseudo-model $\mathcal{L}$ that satisfies conditions (I1) and (I2) of Lemma 2.3.15. We define a function $\text{interval}$ that for every point $d$ returns the interval $I_d$ of \(\mathcal{P}\) $d$ belongs to, that is, $\text{interval}(d) = I_d$ if $d \in I_d$. Moreover, for every interval $I_i$, we denote by $\max(I_i)$ (resp., $\min(I_i)$) the greatest (resp., least) point in $I_i$.

**Lemma 2.3.16.** Let $\varphi$ be a PNL formula. From any safe pseudo-model $\mathcal{L}$ for $\varphi$, we can generate a discrete fulfilling $\mathcal{L}'$ that satisfies $\varphi$.

**Proof.** The basic structure of the proof is the same as that of Lemma 2.3.7. In addition, we partition the domain of every pseudo-model belonging to the (possibly infinite) sequence of pseudo-models $\mathcal{L}_0(= \mathcal{L}), \mathcal{L}_1, \mathcal{L}_2, \ldots$ into a sequence of intervals that satisfies the conditions of Lemma 2.3.15 and we constrain additional points to be placed between existing intervals (or before, resp., after, the first, resp., last one), not inside them. Existing intervals represent consolidated components of the pseudo-model, whose inner points have their definitive immediate predecessor and successor. The addition of a new point forces the extension to the right (resp., left) of the interval that immediately precedes (resp., follows) it and, possibly, the introduction of a new interval. As we will see, such an organization of pseudo-model domains can be exploited to guarantee the discreteness of the resulting fulfilling $\mathcal{L}$. As a preliminary step, we replace the safe pseudo-model $\mathcal{L}_0$ by an extension of it, say, $\mathcal{L}_0'$, that does not contain isolated non-unique points (base case). $\mathcal{L}_0'$ can be built as follows. Suppose that there exists a non-unique point $d \in D_0$ whose immediate predecessor and successor are both unique. Since $d$ is not unique, there exists $d' \in D_0$, with $d' \neq d$, such that $\text{REQ}_0^d(d') = \text{REQ}_0^d(d)$. Let $\mathcal{L}_0([d,d']) = \mathcal{A}$ if $d < d'$, $\mathcal{L}_0([d',d]) = \mathcal{A}$ otherwise. We add a point $d''$ between $d$ and its immediate successor (in $D_0$) and we force $\text{REQ}_0^d(d'')$ to be equal to $\text{REQ}_0^d(d) (= \text{REQ}_0^d(d')$) by defining the labeling of the intervals starting or ending at $d''$ as follows:

$$
\mathcal{L}_0'(d_i,d_j) = \begin{cases} 
\mathcal{L}_0([d_i,d_j]) & d_i \neq d'' \land d_j \neq d'' \\
\mathcal{L}_0([d_i,d]) & d_j = d'' \land d_i < d \\
\mathcal{L}_0([d,d_j]) & d_i = d'' \\
\mathcal{A} & d_i = d \land d_j = d''
\end{cases}
$$

Since $d$ is the successor of a unique point (in $\mathcal{L}_0$) and $\mathcal{L}_0$ is a safe pseudo-model, $d$ is fulfilled (in $\mathcal{L}_0$ and thus in $\mathcal{L}_0'$). Given the above-defined labeling $\mathcal{L}_0'$, it immediately fulfills that $d''$ is fulfilled as well (in $\mathcal{L}_0'$). Let $\mathcal{L}_0'$ be the safe pseudo-model obtained by the application of such a step to all isolated non-unique points in $\mathcal{L}_0$. It is trivial to check that if $\mathcal{L}_0$ is a safe pseudo-model, then $\mathcal{L}_0'$ is a safe pseudo-model as well. Moreover, from Lemma 2.3.15 it immediately follows that we can partition the domain $D_0'$ of $\mathcal{L}_0'$ into a sequence of intervals that satisfies conditions (I1) and (I2).

Now, for any $i \geq 0$, we show that, given a safe pseudo-model $\mathcal{L}_i$, whose domain $D_i$ can be partitioned into a sequence of intervals satisfying conditions (I1) and (I2) of Lemma 2.3.15, we can produce a safe pseudo-model $\mathcal{L}_{i+1}$, whose domain $D_{i+1}$ satisfies the same conditions, which preserves the predecessor/successor relations for all inner points of the intervals belonging to the partition of $D_i$ (inductive step). Even though we basically follows the same path we followed in the proof of Lemma 2.3.7, the additional constraint about the organization of the matters a lot.

Let $I_1, \ldots, I_n$ be a partition of $D_1$ that satisfies conditions (I1) and (I2) and let $Q_0$ be the queue of all points $d \in D_1$ that must be checked for fulfillment. $I_{i+1}$ is built as follows. Let $d$ be the first element of the queue $Q_i$. If $d$ is fulfilled, we remove it from the queue and put $I_{i+1} = I_i$. Otherwise, either there exists $\langle \mathcal{A} \rangle \psi \in \text{REQ}_i^{-1}(d)$ which is not fulfilled, or there exists $\langle \mathcal{A} \rangle \psi \in \text{REQ}_i^{-2}(d)$ which is not fulfilled, or both. Since all points belonging to intervals containing unique points are fulfilled, $\text{interval}(d)$ only consists of non-unique points with the same requests as $d$.

Suppose that there exists a $\langle \mathcal{A} \rangle$-formula in $\text{REQ}_i^{-1}(d)$ which is not fulfilled. Two cases may arise:

1) There exists $d' > d$ such that $\text{REQ}_i^{-1}(d') = \text{REQ}_i^{-1}(d)$ and $d'$ is fulfilled. Let $E(d) = \{d_1, \ldots, d_k\}$. For $j = 1, \ldots, k$, we proceed as follows:
2.3. Decidability of PNL over different classes of linear orderings

a) If $d_i$ is unique, then we proceed as in the proof of Lemma 2.3.7 (no new points are added in such a case).

b) If $d_i$ is not unique, then there exists $\overline{a} \neq d_i$ such that $\text{REQ}^{k_1}(\overline{a}) = \text{REQ}^{k_1}(d_i)$. Let $\overline{a}_i$ be defined as follows:

$$
\overline{a}_i = \begin{cases} 
\max(\text{interval}(d_i)) & \text{if } \text{REQ}^{k_1}(d_i) = \text{REQ}^{k_1}(\max(\text{interval}(d_i))); \\
\min(\text{interval}(d_i)) & \text{otherwise.}
\end{cases}
$$

Let $\overline{a}_i'$ (resp., $\overline{a}_i''$) be the immediate successor (resp., predecessor) of $\overline{a}_i$ in $D_i$ (the solution for the general case can be easily tailored to the simpler cases in which $\overline{a}_i$ is the least, resp., the greatest, point in $D_i$; hence, proof details for these cases are omitted).

If $\overline{a}_i = \max(\text{interval}(d_i))$, we introduce a new point $\hat{d}$ between $\overline{a}_i$ and $\overline{a}_i'$, and we force $\text{REQ}^{k_1+1}(\hat{d})$ to be equal to $\text{REQ}^{k_1}(\overline{a}_i')$ by properly defining the labeling $\mathcal{L}_{i+1}$. To this end, for every $d''$, with $d'' \neq d, d'' \neq d'$, and $d'' \neq \overline{a}_i$, if $d'' < d_i$, we put $\mathcal{L}_{i+1}([d'', \overline{a}_i]) = \mathcal{L}_i([d'', d_i])$, if $d'' > \overline{a}_i$, we put $\mathcal{L}_{i+1}([\overline{a}_i, d'']) = \mathcal{L}_i([\overline{a}_i, d''])$, and if $d_i \leq d'' < \overline{a}_i$, we put $\mathcal{L}_{i+1}([d'', \overline{a}_i]) = \mathcal{L}_i([d'', d_i])$. Moreover, as in Lemma 2.3.7, we put $\mathcal{L}_{i+1}([\overline{a}_i, \hat{d}]) = \mathcal{L}_i([\overline{a}_i, \hat{d}])$ and $\mathcal{L}_{i+1}([d', \hat{d}]) = \mathcal{L}_i([d, d_i])$. In such a way, we force $\hat{d}$ to satisfy over $[d, \overline{a}_i]$ the request that $d_i$ satisfies over $[d', d_i]$. Finally, if $\overline{a}_i > \overline{a}_i'$, we put $\mathcal{L}_{i+1}([d_i, \overline{a}_i]) = \mathcal{L}_i([d_i, \overline{a}_i])$, $\mathcal{L}_{i+1}([\overline{a}_i, \overline{a}_i']) = \mathcal{L}_i([\overline{a}_i, \overline{a}_i'])$ otherwise. For all the remaining pairs $d_r, d_s$ the labeling remains unchanged, that is, $\mathcal{L}_{i+1}([d_r, d_s]) = \mathcal{L}_i([d_r, d_s])$. Such a labeling forces $\hat{d}$ to satisfy the same past requests that $d_i$ satisfies: $\hat{d}$ satisfies over $[d, \overline{a}_i]$ (resp., $[d', \overline{a}_i]$) the request that $d_i$ satisfies over $[d', d_i]$ (resp., $[d, d_i]$) and it satisfies the
remaining past requests over intervals that start at the same point where the intervals over which \( d_j \) satisfies them start. The labeling is depicted in Figure 2.7. If both \( d_j \) and \( \overline{d}_j \) are fulfilled (in \( L_i \)), then \( \overline{d} \) is fulfilled (in \( L_{i+1} \)), while if either \( d_j \), or \( \overline{d}_j \), or both are not fulfilled (in \( L_i \)), we must explicitly check whether or not \( \overline{d} \) is fulfilled (in \( L_{i+1} \)). In all cases, we can consistently extend \( \text{interval}(\overline{d}_j) \) (\( = \text{interval}(d_j) \)) to the right with \( \overline{d} \); moreover, if \( \overline{d} \) is not fulfilled (in \( L_{i+1} \)), we insert it into the queue \( Q_{i+1} \). Then, we introduce a point \( d' \) between \( \overline{d} \) and \( \overline{d}_j \) and we force \( \text{REQ}^{i+1}(\hat{d}') \) to be equal to \( \text{REQ}^i(\overline{d}_j) \). To this end, we put \( L_{i+1}([d'', \overline{d}_j]) = L_i([d'', \overline{d}_j]) \), for all \( d'' < \overline{d}_j \), and \( L_{i+1}([\hat{d}', d'']) = L_i([\overline{d}_j, d'']) \), for all \( d'' > \overline{d}_j \). As for the labeling of \([\hat{d}', \overline{d}_j]\), since \( \overline{d}_j \) is not unique, there exists \( d \neq \overline{d}_j \) with \( \text{REQ}^i(\overline{d}_j) = \text{REQ}^i(d) \). We put \( L_{i+1}([\hat{d}', \overline{d}_j]) = L_i([d, \overline{d}_j]) \) if \( d < \overline{d}_j \), \( L_{i+1}([\hat{d}', \overline{d}_j]) = L_i([\overline{d}_j, d]) \) otherwise. Again, if \( \overline{d}_j \) is fulfilled (in \( L_i \)), then \( \hat{d}' \) is fulfilled (in \( L_{i+1} \)), while if \( \overline{d}_j \) is not fulfilled (in \( L_i \)), being \( \hat{d}' \) fulfilled or not (in \( L_{i+1} \)) depends on the labeling of the interval \([\overline{d}_j, d]\). In all cases, we can consistently extend \( \text{interval}(\overline{d}_j) \) to the left with \( \hat{d}' \); moreover, if \( \hat{d}' \) is not fulfilled (in \( L_{i+1} \)), we insert it into the queue \( Q_{i+1} \).

The case in which \( \overline{d}_j = \min(\text{interval}(d_j)) \) is slightly more involved. First, we observe that either \( \text{interval}(d') \) \( < \) \( \text{interval}(\overline{d}_j) \), and thus \( d' < \overline{d}_j \), or \( \text{interval}(d') = \text{interval}(\overline{d}_j) \) and then, since \( \overline{d}_j = \min(\text{interval}(d_j)) \), \( d' \geq \overline{d}_j \). In the former case (Figure 2.8, top), from \( d < d' \), it immediately follows that \( d < \overline{d}_j \). In the latter case (Figure 2.8, bottom), to conclude that \( d < \overline{d}_j \) we must pair \( d < d' \) with the fact that \( d \), unlike \( \overline{d}_j \), is not fulfilled in \( L_i \), and thus \( \text{interval}(d) < \text{interval}(\overline{d}_j)(= \text{interval}(d')) \). In both cases, we introduce a new point \( \hat{d} \) between \( \overline{d}_j \) and its immediate predecessor \( \overline{d}_j'' \) (from \( d < \overline{d}_j \), it immediately follows that \( d < \hat{d} \)) and we force \( \text{REQ}^{i+1}(\hat{d}) \) to be equal to \( \text{REQ}^i(\overline{d}_j)(= \text{REQ}^i(d_j)) \). To this end, for every \( d'' \), with \( d'' \neq d \) and \( d'' \neq \overline{d}_j \), if \( d'' < \overline{d}_j \), we put \( L_{i+1}([d'', \overline{d}_j]) = L_i([d'', \overline{d}_j]) \), if \( d'' > \overline{d}_j \), we put...
Figure 2.9: Labeling of intervals starting/ending in $\bar{a}$ in case 1b (when $\bar{d}_1 = \min(\text{interval}(d_j))$).

$L_{i+1}([\bar{a}, d'']) = L_i([d_j, d''])$, and if $\bar{d}_1 < d'' \leq d_j$, we put $L_{i+1}([\bar{a}, d'']) = L_i([\bar{a}_j, d''])$.

Moreover, we put $L_{i+1}([d, \bar{a}_1]) = L_i([d', d_j])$, thus forcing $d$ to satisfy over $[d, \bar{a}]$ the request that $d'$ satisfies over $[d', d_j]$. Finally, if $\bar{a} > \bar{a}_1$, we put $L_{i+1}([d, \bar{a}_1]) = L_i([\bar{a}_1, \bar{a}])$, $L_{i+1}([d, \bar{a}_1]) = L_i([\bar{a}_1, \bar{a}_1])$ otherwise. The labeling of the other intervals remains unchanged. The labeling is depicted in Figure 2.9. Unlike the previous case, $\bar{d}$ is not necessarily fulfilled, as $L_{i+1}([d'', \bar{a}]) = L_i([d'', \bar{a}_1])$, for all $d'' \neq d$, and $L_{i+1}([d, \bar{a}_1]) = L_i([d, d_j])$. Since $L_i([d, d_j])$ may be different from $L_i([d, \bar{a}_1])$, past requests of $\bar{d}$ are not necessarily satisfied. If $\bar{d}$ is fulfilled, we can proceed exactly as in the previous case, while if it is not fulfilled, we introduce a new interval between $\text{interval}(\bar{a}_j)$ and $\text{interval}(\bar{a}_1)$, which contains $\bar{a}_1$ only, and we insert $\bar{a}_1$ into the queue $Q_{i+1}$. Moreover, we introduce a point $\bar{d}'$ (resp., $\bar{d}'$) between $\bar{a}$ and $\bar{a}_1$ (resp., $\bar{a}_1'$ and $\bar{d}$) and we force $\text{REQ}^1(\bar{d}'')$ (resp., $\text{REQ}^1(\bar{d}'')$) to be equal to $\text{REQ}^1(\bar{a}_1)$ (resp., $\text{REQ}^1(\bar{a}_1')$). As in the previous case, we can consistently extend $\text{interval}(\bar{a}_1)$ (resp., $\text{interval}(\bar{a}_1')$) to the left (resp., right) with $\bar{d}'$ (resp., $\bar{d}'$) and if $\bar{d}'$ (resp., $\bar{d}'$) is not fulfilled (in $Q_{i+1}$), we insert it into the queue $Q_{i+1}$.

2) For every $d' > d$, with $\text{REQ}^1(d') = \text{REQ}^1(d)$, $d'$ is not fulfilled. Let $d' < d$ such that $\text{REQ}^1(d') = \text{REQ}^1(d)$, $d'$ is fulfilled, and, for every $d' < d'' < d$, if $\text{REQ}^1(d'') = \text{REQ}^1(d)$, then $d''$ is not fulfilled. As in Lemma 2.3.7, we can prove that $\text{Past}^1(d') = \text{Past}^1(d)$. Let $\mathcal{E}_j = [d_1, \ldots, d_k]$. For every $j = 1, \ldots, k$, we proceed as follows:
a) If \( d_1 \) is unique, then \( d_1 > d \), since \( \text{Past}^{i_1}(d') = \text{Past}^{i_1}(d) \). We proceed as in Case 1a.

b) If \( d_1 > d \) and it is not unique, then we proceed as in Case 1b.

c) If \( d' < d_1 < d \), then we introduce a new point \( \hat{d} \) immediately after \( \hat{d} = \max(\text{interval}(d)) \) with the same requests as \( d_1 \). To this end, for every \( d'' \), with \( d'' < d_1 \) (resp., \( d'' > d_1 \)), we put \( L_{i+1}([d'', \hat{d}]) = L_i([d'', d_1]) \) (resp., \( L_{i+1}([\hat{d}, d'']) = L_i([d_1, d'']) \)). As for \( d_1 < d'' < \hat{d} \), since \( \text{Past}^{i_1}(d_1) = \text{Past}^{i_1}(\hat{d}) \), \( \text{Past}^{i_1}(d) \), for every such \( d'' \) there exists \( d''' < d_1 \) such that \( \text{REQ}^{i_1}(d'') = \text{REQ}^{i_1}(d''') \). In particular, \( \text{REQ}^{i_1}(d) = \text{REQ}^{i_1}(d') \). Hence, we put \( L_{i+1}([d, \hat{d}]) = L_i([d', d]) \) and for every \( d_1 < d'' < \hat{d} \), with \( d'' \neq \hat{d} \), we put \( L_{i+1}([d'', \hat{d}]) = L_i([d'', d_1]) \), for a suitable \( d''' < d_1 \) with \( \text{REQ}^{i_1}(d''') = \text{REQ}^{i_1}(d'') \). Moreover, since \( d_i \) is not unique, there exists \( d_1 \) with \( d_i \neq \hat{d} \), such that \( \text{REQ}^{i_1}(d_1) = \text{REQ}^{i_1}(d) \). We put \( L_{i+1}([d_1, \hat{d}]) = L_i([d_1, d]) \) if \( \hat{d} < d_1 \), \( L_{i+1}([\hat{d}, d_1]) = L_i([d_1, d]) \) otherwise. Finally, since \( \text{REQ}^{i_1}(\hat{d}) = \text{REQ}^{i_1}(d') \) and \( \text{REQ}^{i_1+1}(\hat{d}) = \text{REQ}^{i_1}(d_1) \), we put \( L_{i+1}([\hat{d}, \hat{d}]) = L_i([d_1, d]) \).

Now, let \( \overline{d} \) be the immediate successor of \( \hat{d} \) in \( D_i \). If \( \text{REQ}^{i_1+1}(\overline{d}) \neq \text{REQ}^{i_1}(\overline{d}) \), we introduce a new interval between \( \text{interval}(\overline{d}) \) and \( \text{interval}(\overline{d'}) \), which contains \( \overline{d} \) only, and if \( \overline{d} \) is not fulfilled, we insert it into the queue \( Q_{i+1} \). Moreover, as in Case 1b, we consistently extend \( \text{interval}(\overline{d}) \) (resp., \( \text{interval}(\overline{d'}) \)) to the right (resp., left) with \( \overline{d'} \) (resp., \( \overline{d''} \)) and if \( \overline{d'} \) (resp., \( \overline{d''} \)) is not fulfilled (in \( L_{i+1} \)), we insert it into the queue \( Q_{i+1} \). If \( \text{REQ}^{i_1+1}(\overline{d}) = \text{REQ}^{i_1}(\overline{d}) \), we can always extend \( \text{interval}(\overline{d}) \) to the right with \( \overline{d} \) (\( \text{interval}(\overline{d}) \) does not include unique points and thus \( \overline{d} \) does not need to be fulfilled) and if \( \overline{d} \) is not fulfilled (in \( L_{i+1} \)), we insert it into the queue \( Q_{i+1} \). Moreover, we extend \( \text{interval}(\overline{d'}) \) to the left with \( \overline{d''} \) and if \( \overline{d''} \) is not fulfilled (in \( L_{i+1} \)), we insert it into the queue \( Q_{i+1} \).

The case in which there exists a \( (\overline{d}) \)-formula in \( \text{REQ}^{i_1}(d) \) which is not fulfilled is completely symmetric, and thus its description is omitted. This concludes the construction of \( L_{i+1} \). Since \( d \) is fulfilled in \( L_{i+1} \), it can be safely removed from the queue.

Since all points which are fulfilled in \( L_i \) remain fulfilled in \( L_{i+1} \), it is immediate to conclude that \( L_{i+1} \) is a safe pseudo-model (\( L_{i+1} \) safety immediately follows from \( L_i \) one, as additional points are not unique). While the proposed construction does not remove any point, it possibly adds new points to fulfill \( d \) requests and, whenever a new point is introduced, the interval that immediately precedes (resp., follows) it is extended to the right (resp., left). The effect of these extensions is to assign the (definitive) successor (resp., predecessor) to the right (resp., left) endpoint of the interval that immediately precedes (resp., follows) the new point. Again, the choice of a queue to manage points which are (possibly) not fulfilled guarantees that the defects of each of them sooner or later will be fixed. To conclude the proof, it suffices to take the limit of this (possibly infinite) construction as the fulfilling discrete LIS \( L \) for \( \varphi \) we were looking for. This can be done as in Lemma 2.3.7.

\[\square\]

Lemma 2.3.17. Let \( \varphi \) be a PNL formula. From any discrete fulfilling LIS \( L \) that satisfies \( \varphi \), we can extract a safe pseudo-model \( L' \) for \( \varphi \), with \(|D'| \leq 2 \cdot |\varphi| \cdot 2^{4 \cdot |\varphi|+1} \).

Proof. As in the proof of Lemma 2.3.8, we can extract a pseudo-model \( \overline{D} \) from the given discrete fulfilling LIS \( L \). However, such a pseudo-model is not necessarily safe, as we cannot exclude the presence of unique points whose immediate successor or predecessor (in \( \overline{D} \)) are not fulfilled. Such defects can be fixed as follows. Let \( d' \) be a unique point in \( \overline{D} \) which is not safe. We add to \( \overline{D} \) the immediate successor \( d' \) and predecessor \( d'' \) of \( d \) if \( d \) are not already in \( \overline{D} \), and we appropriately extend the labeling. Moreover, we add to \( \overline{D} \) \( \{d', d''\} \) the sets \( \text{ES}^p_{d'} \), \( \text{ES}^p_{d''} \), \( \text{ES}^q_{d'} \), and \( \text{ES}^q_{d''} \), again appropriately extending the labeling. It is immediate to check that the resulting structure \( L' \) is a safe pseudo-model: all the added points, if any, are safe and thus termination of
the expansion process and safety of the resulting structure are guaranteed. As for the size of \( D' \), from Lemma 2.3.8 we have that \( |D| \leq 2 \cdot |\varphi| \cdot 2^{3|\varphi|+1} \). The number of immediate successors or predecessors of points in \( D \) to be added is bounded by \( 2 \cdot |\varphi| \). For each of them, the cardinality of the union of the future and past essential sets is bounded \( |\varphi| \). Hence, \(|D'| \leq 2 |\varphi| \cdot 2^{3|\varphi|+1} + 2 |\varphi|^2 \cdot 2^{|\varphi|} = 2 \cdot 4^{|\varphi|+1} \).

The decidability of PNL over the class of discrete linear orders immediately follows.

**Theorem 2.3.18.** The satisfiability problem for PNL over the class of discrete linear orders is decidable.

**Proof.** The thesis follows from Lemma 2.3.16, Lemma 2.3.17, and Theorem 2.2.7.}

2.3.4 Decidability of PNL over the integers

We have shown that the satisfiability problem for PNL, interpreted over various classes of interval structures, is decidable. As a matter of fact, one can also consider the satisfiability problem for PNL with respect to single temporal structures. In the following, we prove the decidability of PNL over the (interval structure built on the) integers\(^1\). Unlike the decidability results given so far, the next one reduces the satisfiability of a PNL formula to the existence of a finite fulfilling LIS of bounded size or of an infinite one with a finite representation of bounded size (both bounds depend on the size of the PNL formula). This LIS can be obtained by possibly removing exceeding points from a fulfilling LIS. Such a process is definitely not trivial as the removal of a point \( d \) from a fulfilling LIS may affect the satisfiability of formulae over intervals in the past as well as in the future of \( d \). We will show that, in order to fix the defects possibly caused by the removal of \( d \), there must exist sufficiently many points with the same characteristics as \( d \) both in the future and in the past of \( d \). In addition, we must guarantee that changing the valuation of intervals that either begin or end at these points does not generate new defects. In the following, we will describe the process of safely removing a point from a fulfilling LIS.

Let \( L \) be the LIS \( \langle \langle \mathbb{D}, \mathbb{I}(\mathbb{D}) \rangle, \mathcal{L} \rangle \) and \( d \in \mathbb{D} \). We denote by \( L_{-d} \) the set of all LIS \( L' = \langle \langle \mathbb{D}', \mathbb{I}(\mathbb{D}') \rangle, \mathcal{L}' \rangle \) such that \( \mathbb{D}' = \mathbb{D} \setminus \{d\} \) and \( \text{REQ}^L(d') = \text{REQ}^L(d) \), for all \( d' \in \mathbb{D} \setminus \{d\} \). Notice that even though \( L \) and any \( L' \) in \( L_{-d} \) agree on the sets of requests of points in \( \mathbb{D}' \), they do not necessarily agree on the labeling of the intervals over \( \mathbb{D}' \). It is not difficult to show that, given a fulfilling LIS \( L = \langle \langle \mathbb{D}, \mathbb{I}(\mathbb{D}) \rangle, \mathcal{L} \rangle \) and a point \( d \in \mathbb{D} \), \( L_{-d} \) does not necessarily contain a fulfilling LIS. The removal of \( d \) indeed causes the removal of all intervals that either begin or end at \( d \) and there may exist a point \( d' < d \) (resp., \( d' > d \)) in \( \mathbb{D} \) and a formula \( (\bar{A})\psi \in \text{REQ}^L(d') \) (resp., \( (\bar{A})\psi \in \text{REQ}^L(d') \)) such that \( (\bar{A})\psi \) (resp., \( (\bar{A})\psi \)) is fulfilled for \( d' \) in \( L' \), but it is not in any \( L' \in L_{-d} \). The next lemma provides a condition on the fulfilling LIS \( L = \langle \langle \mathbb{D}, \mathbb{I}(\mathbb{D}) \rangle, \mathcal{L} \rangle \) and the point \( d \in \mathbb{D} \) that guarantees the presence of a fulfilling LIS in \( L_{-d} \).

**Lemma 2.3.19.** Let \( L = \langle \langle \mathbb{D}, \mathbb{I}(\mathbb{D}) \rangle, \mathcal{L} \rangle \) be a fulfilling LIS, \( f \) be the number of \((\bar{A})\)-formulae in \( \text{TF}(\varphi) \), and \( p \) be the number of \((\bar{A})\)-formulae in \( \text{TF}(\varphi) \). If there exists a point \( d_c \in \mathbb{D} \) such that (i) there exist at least \( f \cdot p + p \) distinct points \( d < d_c \) with \( \text{REQ}^L(d) = \text{REQ}^L(d_c) \) and (ii) there exist at least \( f \cdot p + f \) distinct points \( d > d_c \) with \( \text{REQ}^L(d) = \text{REQ}^L(d_c) \), then there is at least one fulfilling LIS \( \tilde{L} \) in \( L_{-d_c} \).

**Proof.** Let \( L = \langle \langle \mathbb{D}, \mathbb{I}(\mathbb{D}) \rangle, \mathcal{L} \rangle \) be a fulfilling LIS and let \( d_c \in \mathbb{D} \) be a point such that there exist at least \( f \cdot p + p \) distinct points \( d < d_c \) with \( \text{REQ}^L(d) = \text{REQ}^L(d_c) \) and at least \( f \cdot p + f \) distinct points \( d > d_c \) with \( \text{REQ}^L(d) = \text{REQ}^L(d_c) \). Let \( D' = (\mathbb{D} \setminus \{d_c\}, <) \) and \( \mathcal{L}' = \mathcal{L}_{|_{\mathbb{D}'}}, \) that is, the restriction of \( \mathcal{L} \) to the intervals over \( \mathbb{D}' \). The LIS \( L' = \langle \langle \mathbb{D}', \mathbb{I}(\mathbb{D}') \rangle, \mathcal{L}' \rangle \) clearly belongs to \( L_{-d_c} \), but it is not necessarily a fulfilling LIS. We show how the defects possibly caused by the removal of \( d_c \) can be fixed one by one by properly redefining \( \mathcal{L}' \).

\(^1\)Such a decidability result first appeared (in a short conference version) in [16].
Suppose that there exist \( d < d_e \) and \( (A)\psi \in \text{REQ}_g^\ell(d) \) such that \( \psi \in \mathcal{L}([d, d_e]) \) and there is no \( d' \in D \setminus \{d_e\} \) such that \( \psi \in \mathcal{L}'([d, d']) \) (the symmetric case in which \( d > d_e \) and \( (A)\psi \in \text{REQ}_g^\ell(d) \) can be dealt with in the very same way). Let \( R = \{d_r \in D : d_r > d_e \land (A)\psi \in \text{REQ}_g^\ell(d_r)\} \). To satisfy the request \( (A)\psi \in \text{REQ}_g^\ell(d) \), we change the labeling \( \mathcal{L}'([d, d_e]) \) for some \( d_r \in R \). To prevent such a change from being made and request \( \text{REQ}_g^\ell(d) \), we preliminarily redefine the labeling \( \mathcal{L}' \) as follows. First, we take a minimal set of points \( p^d_e \subseteq D \setminus \{d_e\} \) such that, for every \( (A)\psi \in \text{REQ}_g^\ell(d_e) \), \( \psi \in \mathcal{L}([d_i, d_e]) \) for some \( d_i \in p^d_e \). We call \( p^d_e \) the set of \textit{necessary past points} for \( d_e \). Next, for every \( d_i \in p^d_e \), we take a minimal set of points \( F_{d_i} \subseteq D \) such that, for every \( (A)\psi \in \text{REQ}_g^\ell(d_i) \), \( \psi \in \mathcal{L}([d_i, d_{d_i}]) \) for all \( d_{d_i} \in F_{d_i} \). If all \( F_{d_i} \), \( d_{d_i} \notin F_{d_i} \), we are done. If there exist \( d' \in p^d_e \) and \( (A)\psi \in \text{REQ}_g^\ell(d') \) such that, for all possible choices of the minimal set \( F_{d'} \) for \( d' \), \( d_{d_i} \in F_{d_i} \), we modify the labeling \( \mathcal{L}' \) as follows. Since, by the minimality requirement, \( |p^d_e| \leq p \), for all \( d_i \in p^d_e \), \( \text{REQ}_g^\ell(d_i) \) contains at most \( f \) \( (A) \)-formulae, and, by hypothesis, there exist more than \( f \cdot p \) distinct points \( d > d_e \) with \( \text{REQ}_g^\ell(d) = \text{REQ}_g^\ell(d_e) \), there exists at least one \( \overline{d} > d_e \), with \( \text{REQ}_g^\ell(\overline{d}) = \text{REQ}_g^\ell(d_e) \), such that either \( \mathcal{L}'([d', \overline{d}]) \) fulfills no \( (A) \)-formulae or it only fulfills \( (A) \)-formulae which are also fulfilled by other \( \phi \)-atoms \( \mathcal{L}'([d', \overline{d}]) \) for some \( \overline{d} \neq \overline{d} \). For every \( d_i \in p^d_e \), we redefine \( \mathcal{L}'([d_i, \overline{d}]) \) as \( \mathcal{L}([d_i, d_e]) \). A pictorial account of the updated labeling function is given in Figure 2.10. On the one hand, such a change does not introduce any new defects as \( \text{REQ}_g^\ell(d_e) = \text{REQ}_g^\ell(\overline{d}) = \text{REQ}_g^\ell(\overline{d}) \); on the other hand, it allows us to replace \( d_e \) by \( \overline{d} \) in any set \( F_{d_i} \) that \( d_e \) belongs to (preserving minimality). For every \( d_i \in p^d_e \), we call the resulting set \( F_{d_i} \) the set of \textit{necessary future points} for \( d_i \). The set \( p^d_e \) and the sets \( F_{d_i} \) are graphically depicted in Figure 2.11. Notice that while all points in \( p^d_e \) are less than \( d_e \), for every \( d_i \in p^d_e \), the points in \( F_{d_i} \) are not necessarily greater than \( d_e \), the only condition being that \( d_e \notin F_{d_i} \). In Figure 2.11, we represent the worst case scenario, where all sets \( F_{d_i} \) only contain “copies” of \( d_e \), that is, for every \( F_{d_i} \) and every \( d \in F_{d_i} \), \( \text{REQ}_g^\ell(d) = \text{REQ}_g^\ell(d_e) \). Now let \( G = R \setminus \bigcup_{d_i \in p^d_e} F_{d_i} \). By the minimality requirement, for every \( d_i \in p^d_e \), \( |F_{d_i}| \leq f \) and thus \( |\bigcup_{d_i \in p^d_e} F_{d_i}| \leq f \cdot p \). From condition (ii), it immediately follows that \( |G| \geq f \). We now show that points in \( G \) can be used to fulfill the pending request in \( \text{REQ}_g^\ell(d) \), without generating any new defect. The argument is similar to the one we just used to appropriately redefine the sets \( F_{d_i} \). Since \( \text{REQ}_g^\ell(d) \) contains at most \( f \) \( (A) \)-formulae, there exists at least one \( \overline{d} \in G \) such that either \( \mathcal{L}'([d, \overline{d}]) \) fulfills no \( (A) \)-formulae or it only fulfills \( (A) \)-formulae which are also fulfilled by other \( \phi \)-atoms \( \mathcal{L}'([d, d']) \) for some \( d' \neq \overline{d} \). Let \( \overline{d} \) be one of such “useless” points. We
2.3. Decidability of PNL over different classes of linear orderings

Figure 2.11: Necessary past points for \( d_e \) and necessary future points for \( d_i \), for every \( d_i \in P^{d_e} \).

Figure 2.12: The relabeling that allows one to fix the \( d \)-defect caused by the removal of \( d_e \).

redefine \( L'([d, \bar{d}_i]) \) as \( L([d, d_e]) \), thus fixing the problem for \( \langle A \rangle \psi \in \text{REQ}^{d_e}(d) \). Since \( \text{REQ}^{d_e}(d_e) = \text{REQ}^{d_e}(\bar{d}_i)\) (= \( \text{REQ}^{d_e}(\bar{d}_i) \)), a such change has no impact on the right neighboring intervals of \([d, \bar{d}_i]\). Unfortunately, as an effect of the change in the labeling of \([d, \bar{d}_i]\), one or more \( (\overline{A}) \)-formulae in \( \text{REQ}^{d_e}(\bar{d}_i) \) may become unsatisfied. In such a case, we can recover satisfiability by putting \( L'([d, \bar{d}_i]) = L([d_i, d_e]) \) for all \( d_i \in P^{d_e} \). As points in \( G \) do not belong to the set \( F^{d_e} \) of necessary future points for any \( d_i \in P^{d_e} \), such a change does not introduce any new defect. Figure 2.12 graphically illustrates the above-described update process (for the sake of simplicity, we pictorially show how to update the labeling of one \( d_i \in P^{d_e} \), only; the same change must be applied to the other points in \( P^{d_e} \)).

All other possible defects introduced by the removal of \( d_e \) can be fixed in the same way. Let \( \hat{L} = (\langle \hat{B}, \mathcal{I}(\hat{B}) \rangle, \hat{\mathcal{L}}) \) be the LIS generated by such a recovery procedure. It is immediate to show that \( \hat{L} \) belongs to \( L_{-d_e} \) and it is fulfilling.

As a matter of fact, Lemma 2.3.19 makes no assumption on the underlying interval structure and thus it holds for all the considered (classes of) structures. In the following, we will use it prove that the satisfiability problem for PNL over the integers is decidable.

Since fulfilling LIs over the integers satisfying \( \varphi \) may be arbitrarily large or even infinite, we must find a way to finitely establish their existence. In the following, taking advantage of Lemma 2.3.19, we first give a bound on the size of finite fulfilling LIs that must be checked for satisfiability, when searching for finite \( \varphi \)-models (Theorem 2.3.20); then, we show that we can restrict ourselves to infinite fulfilling LIs with a finite bounded representation, when searching for infinite \( \varphi \)-models (Theorem 2.3.22).
Theorem 2.3.20. Let $\varphi$ be a PNL formula, $L = \langle (D, I(D)), L \rangle$ be a finite fulfilling LIS satisfying $\varphi$, $f$ be the number of $(A)$-formulae in $\text{TF}(\varphi)$, and $p$ be the number of $(\overline{A})$-formulae in $\text{TF}(\varphi)$. Then, there exists a finite fulfilling LIS $\hat{L} = \langle (\hat{D}, I(\hat{D})), \hat{L} \rangle$ satisfying $\varphi$ such that for every $d \in \hat{D}$, there exist at most $m = 2 \cdot f \cdot p + f + p$ pairwise distinct points $d' \in \hat{D}$ with $\text{REQ}^L(d') = \text{REQ}^\varphi(d)$.

Proof. Let $L = \langle (D, I(D)), L \rangle$ be a finite fulfilling LIS that satisfies $\varphi$. If for every $d \in D$, there exist at most $m$ pairwise distinct points $d' \in D$ such that $\text{REQ}^L(d') = \text{REQ}^L(d)$, we are done. If this is not the case, a finite fulfilling LIS with the requested property can be obtained by progressively removing exceeding points from $D$ as follows. Let $L_0 = L$ and let $R_0 = \{\text{REQ}_1, \text{REQ}_2, \ldots, \text{REQ}_k\}$ be the finite set of all and only the sets of requests that violate the property, that is, for every $\text{REQ}_L \in R_0$ there exist more than $m$ pairwise distinct points $d \in D_0$ such that $\text{REQ}^\varphi(d) = \text{REQ}^L(d)$.

Let $D_1$ contains exactly $m$ points $d$ such that $\text{REQ}^L(d) = \text{REQ}^L_1$, as follows. Since there exist more than $m$ pairwise distinct points $d \in D_0$ such that $\text{REQ}^\varphi(d) = \text{REQ}^L(d)$, there exists a point $d_e \in D_0$ such that $\text{REQ}^\varphi(d_e) = \text{REQ}^L_1$, there exist at least $f \cdot p + p$ distinct points $d < d_e$ with $\text{REQ}^\varphi(d) = \text{REQ}^L(d_e)$, and there exist at least $f \cdot p + f$ distinct points $d > d_e$ with $\text{REQ}^\varphi(d) = \text{REQ}^\varphi(d_e)$. From Lemma 2.3.19, it immediately follows that there exists a finite fulfilling LIS $L' \in L_{\ldots d_e}$. We show that $L'$ satisfies $\varphi$. Since $L_0$ satisfies $\varphi$, there exist $d_1, d_2 \in D_0$ such that $\varphi \in L_0([d_1, d_2])$. By Definition 2.2.1, $\langle \varphi \rangle \in \text{CL}(\varphi)$. Hence, by Definition 2.2.3, Definition 2.2.4, and Definition 2.2.5, $\langle \varphi \rangle \in \text{REQ}^\varphi(d_1)$. Two cases may arise: either $d_1 \neq d_e$ (and then $d_1 \in D'$) or $d_1 = d_e$ (and then $d_1 \notin D'$). If $d_1 \in D'$, then $\langle \varphi \rangle \in \text{REQ}^\varphi(d_1)$ and, since $L'$ is fulfilling, there exists $d_k \in D'$ such that $\varphi \in L'(d_1, d_k)$. If $d_1 \notin D'$, then there exist at least $m$ points $d \in L'$ such that $\text{REQ}^L(d) = \text{REQ}^\varphi(d_1)$. Since $L'$ is fulfilling, for any such $d$ there exist $d' \in D'_{\varphi}(d)$ such that $\varphi \in L'(d, d')$. In both cases, $L'$ satisfies $\varphi$. By iterating such a removal step a finite number of times ($D_0$ is finite), we obtain a finite fulfilling LIS $L_{L_0}$ satisfying $\varphi$ whose domain $D_1$ contains exactly $m$ points $d$ such that $\text{REQ}^L(d) = \text{REQ}^L_1$. $\square$

Let us consider now the case of infinite fulfilling LISs over the integers. The elements of the domain of any such LIS can be classified as follows.

Definition 2.3.21. Given an infinite LIS $L = \langle (D, I(D)), L \rangle$ over $\mathbb{Z}$, points in $D$ can be partitioned into the following sets:

- $\text{Fin}(L)$ is the set of all $d \in D$ such that there exist finitely many $d' \in D$ with $\text{REQ}^L(d') = \text{REQ}^L(d)$;
- $\text{Inf}_f(L)$ is the set of all $d \in D$ such that there exist infinitely many $d' \in D$ with $\text{REQ}^L(d') = \text{REQ}^L(d)$, but there exists $d'' \in D$ such that, for all $d' > d''$, $\text{REQ}^L(d') \neq \text{REQ}^L(d)$;
- $\text{Inf}_i(L)$ is the set of all $d \in D$ such that there exist infinitely many $d' \in D$ with $\text{REQ}^L(d') = \text{REQ}^L(d)$, but there exists $d'' \in D$ such that, for all $d' < d''$, $\text{REQ}^L(d') \neq \text{REQ}^L(d)$;
- $\text{Inf}(L)$ is the set of all $d \in D$ such that there exist infinitely many $d' \in D$ with $\text{REQ}^L(d') = \text{REQ}^L(d)$, and, for every $\overline{d} \in D$, there exists $d' < \overline{d}$ such that $\text{REQ}^L(d') = \text{REQ}^L(d)$ and there exists $d'' > \overline{d}$ such that $\text{REQ}^L(d'') = \text{REQ}^L(d)$.

It is worth pointing out that there is not any natural generalization of such a definition to arbitrary linearly-ordered infinite domains.

Given an infinite LIS $L = \langle (D, I(D)), L \rangle$ over $\mathbb{Z}$, let us denote by $d^L_{\min}$ (resp., $d^L_{\max}$) the greatest (resp., least) point in $D$ such that (i) for every $d \in \text{Fin}(L)$, $d^L_{\min} \leq d \leq d^L_{\max}$, (ii) for every $d \in \text{Inf}_f(L)$, there exist at least $f \cdot p + f$ pairwise distinct points $d^L_{\min} \leq d' \leq d^L_{\max}$ with $\text{REQ}^L(d') = \text{REQ}^L(d)$, and (iii) for every $d \in \text{Inf}_i(L)$, there exist at least $f \cdot p + p$ pairwise distinct points $d^L_{\min} \leq d' \leq d^L_{\max}$ with $\text{REQ}^L(d') = \text{REQ}^L(d)$.
2.3. Decidability of PNL over different classes of linear orderings

Theorem 2.3.22. Let \( \varphi \) be a PNL formula, \( L = \langle \langle D, I(D) \rangle, \mathcal{L} \rangle \) be an infinite fulfilling LIS over \( \mathbb{Z} \) satisfying \( \varphi \), \( f \) be the number of \( (\mathcal{A}) \)-formulae in \( TF(\varphi) \), and \( p \) be the number of \( (\overline{\mathcal{A}}) \)-formulae in \( TF(\varphi) \). Then, there exists an infinite fulfilling LIS \( \widehat{L} = \langle \langle \widehat{D}, I(\widehat{D}) \rangle, \overline{\mathcal{L}} \rangle \) over \( \mathbb{Z} \) satisfying \( \varphi \) such that:

- \( \forall d \in \text{Fin}(\widehat{D}) \ \| d' \in \widehat{D} \ | \ \text{REQ}^\mathcal{L}(d') = \text{REQ}^\mathcal{L}(d) \| \leq 2 \cdot p \cdot f + p + f; \)
- \( \forall d \in \text{Inf}_1(\widehat{D}) \ \| d' \in \text{Inf}_1(\widehat{D}) \ | \ \text{REQ}^\mathcal{L}(d') = \text{REQ}^\mathcal{L}(d) \land d_{\min}^L \leq d' \leq d_{\max}^L \| = p \cdot f + f; \)
- \( \forall d \in \text{Inf}_f(\widehat{D}) \ \| d' \in \text{Inf}_f(\widehat{D}) \ | \ \text{REQ}^\mathcal{L}(d') = \text{REQ}^\mathcal{L}(d) \land d_{\min}^L \leq d' \leq d_{\max}^L \| = p \cdot f + p; \)
- \( \forall d \in \text{Inf}_r(\widehat{D}) \ (d < d_{\min}^L \lor d > d_{\max}^L) \).

Proof. Let us define the set of unnecessary points in \( D \) (\( UP_\varphi \)) short as follows:

\[
UP_\varphi = \{ d \in D \ | d_{\min}^L \leq d \leq d_{\max}^L \land [d' > d \mid \text{REQ}^\mathcal{L}(d') = \text{REQ}^\mathcal{L}(d)] \geq p \cdot f + f \\
\land [d' < d \mid \text{REQ}^\mathcal{L}(d') = \text{REQ}^\mathcal{L}(d)] \geq p \cdot f + p \}.
\]

\( UP_\varphi \) is obviously finite (first conjunct). Moreover, for every \( d \in UP_\varphi \), \( i \) there exist at least \( f \cdot p + f \) distinct points \( d' > d \), which do not belong to \( UP_\varphi \), such that \( \text{REQ}^\mathcal{L}(d') = \text{REQ}^\mathcal{L}(d) \) (second conjunct) and \( ii \) there exist at least \( f \cdot p + p \) distinct points \( d' < d \), which do not belong to \( UP_\varphi \), such that \( \text{REQ}^\mathcal{L}(d') = \text{REQ}^\mathcal{L}(d) \) (third conjunct). Hence, we can repeatedly apply the removal procedure of Lemma 2.3.19 to elements in \( UP_\varphi \) until it is completely emptied out. Formally, let \( UP_\varphi = UP_\varphi \cup \Phi_0 \). By applying Lemma 2.3.19, we can obtain a fulfilling LIS \( L^i \in L_{d_\varphi} \). By definition of \( UP_\varphi \), it immediately follows that \( UP_\varphi = UP_\varphi \setminus \{ d_0 \} \). To show that the fulfilling LIS \( L^i \) satisfies \( \varphi \), we can proceed exactly as in Theorem 2.3.20. By iterating \( n \) times such a procedure, with \( n = |UP_\varphi| \), we can produce a sequence of fulfilling LIS satisfying \( \varphi \| D, L^1, \ldots, L^n \) such that, for \( i = 0, \ldots, n - 1 \), \( L^i \subseteq L_{d_{i+1}} \subseteq L_{d_1} \), and \( D_1 \supseteq D_2 \supseteq \ldots \supseteq D_{n+1} \), and \( UP_{\varphi} = \emptyset \).

It can be easily checked that \( L^n \) contains at most \( 2 \cdot p \cdot f + p \cdot f + p \) points belonging to \( \text{Fin}(L^n) \), exactly \( p \cdot f + f \) points \( d' \in \text{Inf}_1(L^n) \) such that \( (d_{\min}^L = d_{\min}^L \leq d' \leq d_{\max}^L = d_{\max}^L) \), exactly \( p \cdot f + p \) points \( d' \in \text{Inf}_f(L^n) \) such that \( (d_{\min}^L = d_{\min}^L \leq d' \leq d_{\max}^L = d_{\max}^L) \), and no points \( d' \in \text{Inf}_r(L^n) \) such that \( (d_{\min}^L = d_{\min}^L \leq d' \leq d_{\max}^L = d_{\max}^L) \). Hence, we can take \( L^n \) as the infinite fulfilling \( L \) over \( \mathbb{Z} \) satisfying \( \varphi \) that we were looking for.

\( \square \)

Theorem 2.3.23. Let \( \varphi \) be a PNL formula, \( f \) (resp., \( p \)) be the number of \( (\mathcal{A}) \)-formulae (resp., \( (\overline{\mathcal{A}}) \)-formulae) in \( TF(\varphi) \), and \( \mathcal{F}, \mathcal{I}_1, \mathcal{I}_r, \mathcal{J} \) be pairwise disjoint subsets of \( 2^{TF(\varphi)} \). It holds that there exists an infinite fulfilling LIS \( L = \langle \langle D, I(D) \rangle, \mathcal{L} \rangle \) over \( \mathbb{Z} \) satisfying \( \varphi \) such that:

a. for every \( d \in \text{Fin}(L) \), \( \text{REQ}^\mathcal{L}(d) \in \mathcal{F} \) and \( \| d' \in D \mid \text{REQ}^\mathcal{L}(d') = \text{REQ}^\mathcal{L}(d) \| \leq 2 \cdot p \cdot f + f + f, \)

b. for every \( d \in \text{Inf}_1(L) \), \( \text{REQ}^\mathcal{L}(d) \in \mathcal{I}_1 \) and \( \| d' \in D \mid d_{\min}^L \leq d' \leq d_{\max}^L \land \text{REQ}^\mathcal{L}(d') = \text{REQ}^\mathcal{L}(d) \| = p \cdot f + f, \)

c. for every \( d \in \text{Inf}_f(L) \), \( \text{REQ}^\mathcal{L}(d) \in \mathcal{I}_r \) and \( \| d' \in D \mid d_{\min}^L \leq d' \leq d_{\max}^L \land \text{REQ}^\mathcal{L}(d') = \text{REQ}^\mathcal{L}(d) \| = p \cdot f + p, \)

d. for every \( d \in \text{Inf}_r(L) \), \( \text{REQ}^\mathcal{L}(d) \in \mathcal{J} \), and there are not \( d' \in D \) such that \( d_{\min}^L \leq d' \leq d_{\max}^L \) and \( \text{REQ}^\mathcal{L}(d') = \text{REQ}^\mathcal{L}(d) \),

e. for every \( S \in \mathcal{F} \) (resp., \( \mathcal{I}_1, \mathcal{I}_r, \mathcal{J} \)), there exists \( d \in \text{Fin}(L) \) (resp., \( \text{Inf}_1(L), \text{Inf}_f(L), \text{Inf}_r(L) \)) such that \( \text{REQ}^\mathcal{L}(d) = S \),

if and only if the following conditions hold:

1. Consistency conditions

1a. there exists a finite LIS \( L' = \langle \langle D', I'(D') \rangle, \mathcal{L}' \rangle \) such that \( i \) for every \( d \in D' \), there exists \( S \in \mathcal{F} \cup \mathcal{I}_1 \cup \mathcal{I}_r \cup \mathcal{J} \) such that \( \text{REQ}^\mathcal{L}(d) = S \), \( ii \) for every \( S \in \mathcal{F} \), \( L' \) features at least one and at most \( 2 \cdot p \cdot f + f + f \) points \( d \) with \( \text{REQ}^\mathcal{L}(d) = S \), \( iii \) for every \( S \in \mathcal{I}_1 \), \( L' \) features exactly \( p \cdot f + f \) points \( d \) with \( \text{REQ}^\mathcal{L}(d) = S \), \( iv \) for every \( S \in \mathcal{I}_r \), \( L' \) features exactly \( p \cdot f + f \) points \( d \) with \( \text{REQ}^\mathcal{L}(d) = S \), and \( v \) for every \( S \in \mathcal{J} \), \( L' \) features no \( d \) with \( \text{REQ}^\mathcal{L}(d) = S \).
2a. for every $S_1 \in \mathcal{J}_t \cup \mathcal{J}_r$ and $\langle A \rangle \psi \in S_1$, there exists an interval-tuple $(S_1, A, S_2)$ with $S_2 \in \mathcal{J}_t \cup \mathcal{J}_r$ and $\psi \in A$, and for every $S_2 \in \mathcal{J}_t \cup \mathcal{J}_r$ and $\langle A \rangle \psi \in S_2$, there exists an interval-tuple $(S_1, A, S_2)$ with $S_1 \in \mathcal{J}_t \cup \mathcal{J}_r$ and $\psi \in A$.

2b. for every $S_2 \in \mathcal{J}_t \cup \mathcal{J}_r$ and $\langle A \rangle \psi \in S_2$, there exists an interval-tuple $(S_1, A, S_2)$ with $S_1 \in \mathcal{J}_t \cup \mathcal{J}_r$ and $\psi \in A$, and for every $S_1 \in \mathcal{J}_t \cup \mathcal{J}_r$ and $\langle A \rangle \psi \in S_1$, there exists an interval-tuple $(S_1, A, S_2)$ with $S_2 \in \mathcal{J}_t \cup \mathcal{J}_r$ and $\psi \in A$.

2c. for every non-unique point $d$ in $L'$ and every $\langle A \rangle \psi \in \text{REQ}^{L'}(d)$ which is not fulfilled for $d$ in $L'$, there exists an interval-tuple $(\langle A \rangle \psi), (d, A, S_2)$ with $S_2 \in \mathcal{J}_t \cup \mathcal{J}_r$ and $\psi \in A$, and for every non-unique point $d$ in $L'$ and every $\langle A \rangle \psi \in \text{REQ}^{L'}(d)$ which is not fulfilled for $d$ in $L'$, there exists an interval-tuple $(S_1, A, \text{REQ}^{L'}(d))$, with $S_1 \in \mathcal{J}_t \cup \mathcal{J}_r$ and $\psi \in A$.

2d. for every unique point $d$ in $L'$ and every $\langle A \rangle \psi \in \text{REQ}^{L'}(d)$ which is not fulfilled for $d$ in $L'$, there exists an interval-tuple $(\langle A \rangle \psi), (d, A, S_2)$ such that (i) $S_2 \in \mathcal{J}_t \cup \mathcal{J}_r$, (ii) $\psi \in A$, (iii) for every $\langle A \rangle \psi \in S_2$, with $1 \leq i \leq p' \leq p$, there exists an interval-tuple $(S_1, A, S_2)$, with $S_1 \in \mathcal{J}_t \cup \mathcal{J}_r$, and $\psi \in A_r$. (iv) if there exist $1 \leq j \leq p'$ and $d_1 \in D'$ such that $\text{REQ}^{L'}(d_1) = S_1$ and $d_1$ is unique, then $S_1 \neq S_2$ for every $i \neq j$, and (v) if there exists $j$ such that $S_1^j = \text{REQ}^{L'}(d)$, then $A_j = A$, and for every unique point $d$ in $L'$ and every $\langle A \rangle \psi \in \text{REQ}^{L'}(d)$ which is not fulfilled for $d$ in $L'$, there exists an interval-tuple $(S_1, A, \text{REQ}^{L'}(d))$ such that (i) $S_1 \in \mathcal{J}_t \cup \mathcal{J}_r$, (ii) $\psi \in A$, (iii) for every $\langle A \rangle \psi \in S_1$, with $1 \leq i \leq f' \leq f$, there exists an interval-tuple $(S_1, A, S_2)$, with $S_1 \in \mathcal{J}_t \cup \mathcal{J}_r$, and $\psi \in A_r$. (iv) if there exist $1 \leq j \leq f'$ and $d_1 \in D'$ such that $\text{REQ}^{L'}(d_1) = S_1^j$ and $d_1$ is unique, then $S_1^j \neq S_2^j$ for every $i \neq j$, and (v) if there exists $j$ such that $S_1^j = \text{REQ}^{L'}(d)$, then $A_j = A$.

Proof. The left-to-right direction is straightforward. Let $L'$ be the restriction of $L$ to the interval $[d_{\text{min}}, d_{\text{max}}]$. Truth of consistency conditions 1a-1c immediately follows from conditions a-e and Definition 2.2.5 (notion of LIS). Truth of fulfilling conditions 2a-2d easily follows from conditions a-e and Definition 2.2.6 (notion of fulfilling LIS).

Let us consider now the right-to-left direction. As a preliminary step, we replace $L'$ by an expanded LIS $L''$ that, for each $S \in \mathcal{J}_t \cup \mathcal{J}_r \cup \mathcal{J}_t \cup \mathcal{J}_r$, features sufficiently many points $d \in D''$ with $\text{REQ}^{L''}(d) = S$. Let $m = \max([p, f]) + 1$. We put $L_0 = L'$ and for $i \geq 0$, we iteratively define $L_{i+1}$ as follows:

- if for every $S \in \mathcal{F}$ such that there exist $d, d' \in D'$ with $d \neq d'$ and $\text{REQ}^{L'}(d) = \text{REQ}^{L'}(d') = S$, $|[d \in D_1 \mid \text{REQ}^{L'}(d) = S]| \geq m$, and for every $S \in \mathcal{J}$, $|[d \in D_1 \mid \text{REQ}^{L}(d) = S]| = m$, we put $L'' = L_1$ and we end the procedure;

- for every $S \in \mathcal{F}$ such that there exist $d, d' \in D'$ with $d \neq d'$ and $\text{REQ}^{L'}(d) = \text{REQ}^{L'}(d') = S$, and $|[d \in D_1 \mid \text{REQ}^{L'}(d) = S]| \geq m$, and $|[d \in D_1 \mid \text{REQ}^{L}(d) = S]| \geq 1$, we define $L_{i+1} = \langle \text{variants of } L_{i+1} \rangle$ as follows. Without loss of generality, we assume $d < d'$ and we add a new point $d''$ immediately after $d$ to $D_1$ (thus $D_{i+1} = D_1 \cup \{d''\}$), with $\text{REQ}^{L_{i+1}}(d'') = S$. Moreover, for every $\overline{d} \in D_1$, with $\overline{d} < d$, we put $L_{i+1}((\overline{d}, d'')) = L_i((\overline{d}, d))$, for every $\overline{d} \in D_1$, with $\overline{d} > d$, $L_{i+1}((d'', \overline{d})) = L_i((d, \overline{d}))$, for every $\overline{d}, \overline{d}' \in D_1$, we put $L_{i+1}((\overline{d}, \overline{d}')) = L_i((\overline{d}, \overline{d}'))$;

- for every $S \in \mathcal{J}$ such that $|[d \in D_1 \mid \text{REQ}^{L}(d) = S]| \geq m$, we define $L_{i+1} = \langle \text{variants of } L_{i+1} \rangle$, $L_{i+1}$ as follows. First, we add a new point $d'$ such that $d' > d$ for all $d \in D_1$ (thus $D_{i+1} = D_1 \cup \{d'\}$), with $\text{REQ}^{L_{i+1}}(d') = S$. Moreover, for every $\overline{d} \in D_1$, we put $L_{i+1}((\overline{d}, d')) = L_i((\overline{d}, d'))$. Finally, for every $d \in D_1$, either $d \in D'$ and thus $\text{REQ}^{L_i}(d) = \text{REQ}^{L'}(d) = S'$.
for some $S' \in J_1 \cup \mathcal{J} \cup J_r$ or $d$ has been added to the domain at some step $i' \leq i$ and thus 
$\text{REQ}^{-1}(d) \in \mathcal{J} \cup \mathcal{J}$. Hence, by consistency conditions 1b-1c, for every $d \in D_1$, there exists an interval-tuple $(\text{REQ}^{-1}(d), A, S)$ for some atom $A$ and thus we can put $L_{i+1}[(d, d')] = A$.

It can be easily checked that the above procedure terminates after a finite number of iterations and it preserves the validity of consistency conditions 1b-1c and fulfilling conditions 2a-2d. Let $L''$ be the resulting LIS. It can be easily checked that $L''$ satisfies all relevant properties of the initial LIS $L'$. In addition, for every $S \in \mathcal{J} \cup \mathcal{J} \cup J_1 \cup \mathcal{J}$ there exists at least $m$ points $d \in D''$ with $\text{REQ}^{L''}(d) = S$.

We show now how to turn $L''$ into an infinite fulfilling LIS $L$ over $Z$ satisfying $\psi$. To start with, we put $L_0 = L''$ and we insert all $d \in D_0$ which are not fulfilled into a queue $Q_0$. For every $i \geq 1$, $L_{i+1}$ is obtained from $L_i$ by adding at most one new point before the minimum or after the maximum of $D_i$ and suitably extending the labeling $L_i$. Moreover, for every $i \geq 0$ we guarantee that $L_i$ still satisfies consistency conditions 1b-1c and fulfilling conditions 2a-2d. $L_{i+1}$ is built on by executing one among the following three steps:

**Fulfilling** If $Q_i$ is empty, we simply put $Q_{i+1} = Q_i$ and $L_{i+1} = L_i$. Otherwise, let $d$ be the first element of $Q_i$. If $d$ is fulfilling, we simply remove it from $Q_i$, that is, $Q_{i+1} = Q_i \setminus \{d\}$, and we let $L_{i+1} = L_i$. Otherwise, suppose that there exists $(\overline{A})\psi \in \text{REQ}(d)$ which is not fulfilled in $L_i$ (the case of $(\overline{A})\psi \in \text{REQ}(d)$ is completely symmetric, and thus its analysis is omitted). Two cases may arise:

1) if $d$ is not unique, then, by fulfilling condition 2c, there exists an interval tuple $(\text{REQ}(d), A_{\psi}, S)$, with $S \in J_r \cup \mathcal{J}$ and $\psi \in (A)_S$, and, by the initial expansion step, there exists $d' \in D_i$ such that $\text{REQ}(d') = S$. Then, by taking advantage of fulfilling condition 2b, we define a set of $p' \leq p + 1$ interval-tuples $T = \{(S_1, A_1, S), \ldots, (S_{p'}, A_{p'}, S)\}$ such that:

* $S_1 = \text{REQ}(d), A_1 = A_{\psi}$, and for every $(\overline{A})\psi \in S$, there exists $(S_1, A_1, S) \in T$ with $\psi \in A_1$ and
* if $(\overline{A})\psi$ is fulfilled for $d'$ in $L_i$, then there exists a point $d'' \in D_i$, with $d'' < d'$, such that $(\text{REQ}(d''), L[(d'', d')]. \text{REQ}(d')) = (S_1, A_1, S)$, for some $(S_1, A_1, S) \in T$ with $\psi \in A_1$, while if $(\overline{A})\psi$ is not fulfilled for $d'$ in $L_i$, then there exists an interval tuple $(S_1, A_1, S)(\{S_1, A_1, \text{REQ}(d')\}) \in T$ such that $S_1 \in J_1 \cup J$ and $\psi \in A_1$.

It can be easily shown that if there exists a unique point $d'' \in D_i$ such that $\text{REQ}(d'') = S_1$, for some $(S_1, A_1, S) \in T$, then for every $(S_h, A_h, S) \in T$, with $h \neq i$, $S_h \neq S_1$. Since there are no repetitions of the set of requests associated with a unique point in the (first component of the interval-tuples belonging to) $T$, we can always choose $p'$ distinct points $d_1(= d), d_2, \ldots, d_{p'} \in D_i$ such that $\text{REQ}(d_j) = S_1$ for $j = 1, \ldots, p'$. In the worst case, we may have $\text{REQ}(d_1) = \ldots = \text{REQ}(d_{p'})$, with $p' = p + 1 \leq m$ (by the initial expansion step, any set of requests, which is not associated with a unique point, is associated with at least $m$ distinct points of $D_i$). Now, to fulfill $(\overline{A})\psi \in \text{REQ}(d)$, we add a point $d$ to $D_i$ such that $d > d$ for every $d \in D_i$ and we force $\text{REQ}(d)$ to be equal to $S$ by putting $L_{i+1}[d, d] = A_1$ for every $1 \leq j \leq p'$. Moreover, for every $d'' \in D_i$ such that $d'' \neq d_1, \ldots, d_{p'}, \text{consistency conditions guarantee that there exists an}$ interval-tuple $(\text{REQ}(d''), A, S)$ and thus we can put $L_{i+1}[(d'', d)] = A$.

2) if $d$ is unique, then, by fulfilling condition 2d, there exists an interval tuple $(\text{REQ}(d), A_{\psi}, S)$, with $S \in J_r \cup \mathcal{J}$ and $\psi \in (A)_S$, and, by the initial expansion step, there exists $d' \in D_i$ such that $\text{REQ}(d') = S$. Moreover, by taking advantage of fulfilling condition 2d, we can also define a set of $p' \leq p + 1$ interval-tuples $T = \{(S_1, A_1, S), \ldots, (S_{p'}, A_{p'}, S)\}$ such that:

(i) $S_1 = \text{REQ}(d)$ and $A_1 = A_{\psi}$; (ii) for every $(\overline{A})\psi \in S$, there exists $(S_1, A_1, S) \in T$ with $\psi \in A_1$; and (iii) if there exists a unique point $d'' \in D_i$ such that $\text{REQ}(d'') = S_1$, for some $(S_1, A_1, S) \in T$, then for every $(S_h, A_h, S) \in T$, with $h \neq i$, $S_h \neq S_1$. As in the previous case, since there are no repetitions of the set of requests associated with a unique point in $T$, we can always choose $p'$ distinct points $d_1(= d), d_2, \ldots, d_{p'} \in \mathcal{J}$ for every $d \in D_i$ and we force $\text{REQ}(d)$ to be equal to $S$ by putting $L_{i+1}[d, d] = A_1$ for every $1 \leq j \leq p'$. Moreover, for every $d'' \in D_i$ such that $d'' \neq d_1, \ldots, d_{p'}, \text{consistency conditions guarantee that there exists an}$ interval-tuple $(\text{REQ}(d''), A, S)$ and thus we can put $L_{i+1}[(d'', d)] = A$. 

2.3. Decidability of PNL over different classes of linear orderings
D_{i+1}

L_{i+1}([d_{p'}, \bar{d}]) = A_{p'}

L_{i+1}([d_1, \bar{d}]) = A_j

L_{i+1}([d_{j-1}, \bar{d}]) = A_{j-1}

L_{i+1}([d_2, \bar{d}]) = A_2

L_{i+1}([d, \bar{d}]) = A_1

d_1 = d \quad d_2 \quad d_{j-1} \quad d_1 \quad d_{p'} \quad \bar{d}

\ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots

p'-1 \leq p

D_i

Figure 2.13: The fulfilling step: addition of a new point and fulfillment of its past requests.

D_i such that REQ(d_j) = S_j for j = 1, \ldots, p'. In the worst case, we may have REQ(d_2) = \ldots =REQ(d_{p'}), with p' = p + 1 \leq m. As before, to fulfill \langle A \rangle \psi \in REQ(d), we add a point \bar{d} to D_i such that \bar{d} > d for every d \in D_i and we force REQ(\bar{d}) to be equal to S by putting \mathcal{L}_{i+1}[d_j, \bar{d}] = A_j for every 1 \leq j \leq p'. Moreover, for every d'' \in D_i such that d'' \neq d_1, \ldots, d'' \neq d_{p'}, consistency conditions guarantee that there exists an interval-tuple (REQ(d''), A, S) and thus we can put \mathcal{L}_{i+1}([d'', \bar{d}]) = A.

It can be easily checked that all \langle \bar{A} \rangle \psi \in REQ(\bar{d}) are fulfilled for \bar{d} in L_{i+1} and thus L_{i+1} satisfies conditions 1b-1c and 2a-2d.

**Witness future S** Let S \in J \cup S. By the initial expansion step, there exists d' \in D_i such that REQ(d) = S. Again, by using fulfillment condition 2b, we define a set of p' \leq p interval-tuples T = [{S_1, A_1, S}, \ldots, {S_{p'}, A_{p'}, S}] such that for every \langle A \rangle \psi \in S, there exists \langle S_j, A_j, S \rangle \in T with \psi \in A_j, and if there exists a unique point d'' \in D_i such that REQ(d'') = S_j, for some \langle S_j, A_j, S \rangle \in T, then for every \langle S_h, A_h, S \rangle \in T, with h \neq j, S_h \neq S_j. Next, we choose p' distinct points d_1(= d), d_2, \ldots, d_{p'} \in D_i such that REQ(d_j) = S_j for j = 1, \ldots, p'. In the worst case, we may have REQ(d_1) = \ldots =REQ(d_{p'}), with p' = p \leq m. As before, to fulfill \langle A \rangle \psi \in REQ(d), we add a point \bar{d} to D_i such that \bar{d} > d for every d \in D_i and we force REQ(\bar{d}) to be equal to S by putting \mathcal{L}_{i+1}[d_j, \bar{d}] = A_j for every 1 \leq j \leq p'. Moreover, for every d'' \in D_i such that d'' \neq d_1, \ldots, d'' \neq d_{p'}, consistency conditions guarantee that there exists an interval-tuple (REQ(d''), A, S) and thus we can put \mathcal{L}_{i+1}([d'', \bar{d}]) = A. It can be easily checked that all \langle \bar{A} \rangle \psi \in REQ(\bar{d}) are fulfilled for \bar{d} in L_{i+1} and thus L_{i+1} satisfies conditions 1b-1c and 2a-2d.

**Witness past S** Given S \in J \cup S, such a step adds a new point \bar{d} to D_i such that \bar{d} < d for every d \in D_i and REQ(\bar{d}) = S. It is fully symmetric to the previous step, and thus its description is omitted.
Apart from the fulfilling step when the queue is empty, the execution of one of the above steps adds a point \( d \) greater than or less than all points in the domain of the current LIS and it properly extends the labeling. Changes to the current labeling are never requested. Any infinite sequence of steps where the fulfilling witness future \( S \), for every \( S \in \mathcal{I} \cup \mathcal{J} \), and witness past \( S \), for every \( S \in \mathcal{I} \cup \mathcal{J} \), steps occur infinitely often generates (as a limit construction) a fulfilling LIS \( L_\omega (\langle (D_\omega, I_\omega), (D_\omega, I_\omega) \rangle) \) over \( Z \) satisfying \( \varphi \).

Putting together the above results, we can prove the decidability of PNL over the integers.

**Theorem 2.3.24.** The satisfiability problem for PNL over \( Z \) is decidable.

**Proof.** The thesis follows from Theorem 2.3.20, Theorem 2.3.22, and Theorem 2.3.23.

### 2.4 Tableau-based decision procedures for PNL

In this section, we develop tableau-based decision procedures for PNL over all linear orders, dense, and discrete linear orders as well as over the integers. We first describe the tableau system for the case of all linear orders, and then we present those for the dense, discrete, and integer cases, which are obtained as specializations of the general one. All tableau descriptions are organized as follows: first, we give the rules of the tableau system; then, we describe expansion strategies and blocking conditions; finally, we prove termination, soundness, and completeness of the method. We conclude the section by proving the optimality of all the proposed tableau-based decision procedures.

#### 2.4.1 A tableau-based decision procedure for PNL over all linear orders

As a preliminary step, basic notation and definitions are given (they apply to all tableau systems described in the following). A tableau for a PNL formula \( \varphi \) is a special decorated tree \( \mathcal{T} \). We associate a finite linear order \( D_B \) and a request function \( \text{REQ}_\varphi : D_B \rightarrow \text{REQ}_\varphi \) with every branch \( B \) of \( \mathcal{T} \). Every node \( n \) in \( B \) is labeled with a pair \( \langle d_l, d_r \rangle, A_n \rangle \) such that the triple \( \langle \text{REQ}_B(d_l), A_n, \text{REQ}_B(d_r) \rangle \) is an interval-tuple. The initial tableau for \( \varphi \) consists of a single node (and thus of a single branch \( B \)) labeled with the pair \( \langle d_0, d_1 \rangle, A_0 \rangle \), where \( D_B = \{ d_0 < d_1 \} \) and \( \varphi \in A_0 \).

Given a point \( d \in D_B \) and a formula \( (A)\psi \in \text{REQ}_\varphi (d) \), we say that \( (A)\psi \) is fulfilled in \( B \) for \( d \) if there exists a node \( n' \in B \) such that \( n' \) is labeled with \( \langle d_0, d_1 \rangle, A_n \rangle \) and \( \psi \in A_n \). Similarly, given a point \( d \in D_B \) and a formula \( (\overline{A})\psi \in \text{REQ}_\varphi (d) \), we say that \( (\overline{A})\psi \) is fulfilled in \( B \) for \( d \) if there exists a node \( n' \in B \) such that \( n' \) is labeled with \( \langle d_0, d_1 \rangle, A_n \rangle \) and \( \psi \in A_n \). Given a point \( d \in D_B \), we say that \( d \) is fulfilled in \( B \) if every \( (A)\psi \) (resp., \( (\overline{A})\psi \)) in \( \text{REQ}_\varphi (d) \) is fulfilled in \( B \) for \( d \).

Let \( \mathcal{T} \) be a tableau and \( B \) be a branch of \( \mathcal{T} \), with \( D_B = \{ d_0, \ldots, d_k \} \). We denote by \( B \cdot n \) the expansion of \( B \) with an immediate successor node \( n \) and by \( B \cdot n_1, \ldots, n_h \) the expansion of \( B \) with \( h \) immediate successor nodes \( n_1, \ldots, n_h \). To possibly expand \( B \), we apply one of the following expansion rules:

1. **(A)-rule.** If there exist \( d_l \in D_B \) and \( (A)\psi \in \text{REQ}_\varphi (d_l) \) such that \( (A)\psi \) is not fulfilled in \( B \) for \( d_l \), we proceed as follows. If there is not an interval-tuple \( \langle \text{REQ}_B(d_l), A_\psi, S \rangle \), with \( \psi \in A_\psi \), we close the branch \( B \). Otherwise, let \( \langle \text{REQ}_B(d_l), A_\psi, S \rangle \) be an interval-tuple such that \( \psi \in A_\psi \). We take a new point \( d \) and we expand \( B \) with \( h = k + 1 \) immediate successor nodes \( n_1, \ldots, n_h \) such that, for every \( 1 \leq l \leq h \), \( D_{B \cdot n_l} = D_B \cup \{ d_1, \ldots, d_h \} \) for \( l = h \), we simply add a new point \( d ', d_f > d_k \), to the linear order), \( n_1 = \langle d_l, d_r \rangle, A_\psi \rangle \), with \( \psi \in A_\psi \), \( \text{REQ}_B (n_1) \rangle = S \), and \( \text{REQ}_B (n_1) \rangle = \text{REQ}_B (d') \rangle = \text{REQ}_B (d') \rangle = D_B \).
2. **(\overline{A})-rule.** If there exist \( d_l \in D_B \) and \( (\overline{A})\psi \in \text{REQ}_\varphi (d_l) \) such that \( (\overline{A})\psi \) is not fulfilled in \( B \) for \( d_l \), we proceed as follows. If there is not an interval-tuple \( \langle S, A_\psi', \text{REQ}_\varphi (d_l) \rangle \), with \( \psi \in A_\psi \), we close the branch \( B \). Otherwise, let \( \langle S, A_\psi', \text{REQ}_\varphi (d_l) \rangle \) be an interval-tuple such that \( \psi \in A_\psi \). We take a new point \( d \) and we expand \( B \) with \( h = j + 1 \) immediate successor nodes
nodes \( n_0, \ldots, n_i \) such that, for every \( 0 \leq l < h \), \( \mathbb{D}_B{n_l} = \mathbb{D}_B \cup \{ d_{l-1} < d < d_l \} \) (for \( l = 0 \), we simply add a new point \( d \), with \( d < d_0 \), to the linear order), \( n_1 = (\{ d_l \}, A_\psi) \), with \( \psi \in A_\psi \), \( \text{REQ}_B{n_l}(d) = S \), and \( \text{REQ}_B{n_l}(d') = \text{REQ}_B{d'}(d) \) for every \( d' \in \mathbb{D}_B \).

3. **Fill-in rule.** If there exist two points \( d_1, d'_1 \), with \( d_1 < d'_1 \), such that there is not a node in \( B \) decorated with the interval \( [d_1, d_1] \) and there exists an interval-tuple \( (\text{REQ}_B(d_1), A, \text{REQ}_B(d'_1)) \), then we expand \( B \) with a node \( n = (\{ d_1, d_1 \}, A) \) (if such an interval-tuple does not exist, then we *close* the branch \( B \)).

The application of any of the above rules may result in the replacement of the branch \( B \) with one or more new branches, each one featuring one new node \( n \). However, while the **Fill-in rule** decorates such a node with a new interval whose endpoints already belong to \( \mathbb{D}_B \), the \( (A) \)-rule (resp., \( \overline{A} \))-rule adds a new point \( d \) to \( \mathbb{D}_B \) which becomes the ending (resp., beginning) point of the interval associated with the new node.

We say that a node \( n = (\{ d_1, d_1 \}, A) \) in a branch \( B \) is *active* if for every predecessor \( n' = (\{ d,d' \}, A') \) of \( n \) in \( B \), the interval-tuples \( (\text{REQ}_B(d_1), A, \text{REQ}_B(d_1)) \) and \( (\text{REQ}_B(d), A', \text{REQ}_B(d')) \) are different. Moreover, we say that a point \( d \in \mathbb{D}_B \) is *active* if and only if there exists an active node \( n \) in \( B \) such that \( n = (\{ d, d' \}, A) \) or \( n = (\{ d', d \}, A) \), for some \( d' \in \mathbb{D}_B \) and some atom \( A \). Given a non-closed branch \( B \), we say that \( B \) is *complete* if for every \( d_1, d_1 \in \mathbb{D}_B \), with \( d_1 < d_1 \), there exists a node \( n \) in \( B \) labeled with \( n = (\{ d_1, d_1 \}, A) \), for some atom \( A \). It can be easily seen that if \( B \) is complete, then the tuple \( (\mathbb{D}_B, I(\mathbb{D}_B), \mathcal{L}_B) \) such that, for every \( [d_1, d_1] \in I(\mathbb{D}_B), \mathcal{L}(d_1, d_1) = A \) if and only if there exists a node \( n \) in \( B \) labeled with \( (\{ d_1, d_1 \}, A) \), is a LIS. Given a non-closed branch \( B \), we say that \( B \) is *blocked* if \( B \) is complete and for every active point \( d \) in \( B \) we have that \( d \) is fulfilled in \( B \). We start from an initial tableau for \( \varphi \) and we apply the expansion rules to all the non-blocked and non-closed branches \( B \). The expansion strategy is the following:

- apply the **Fill-in rule** until it generates no new nodes in \( B \);
- if there exist an active point \( d \in \mathbb{D}_B \) and a formula \( (A)\psi \in \text{REQ}_B(d) \) such that \( (A)\psi \) is not fulfilled in \( B \) for \( d \), then apply the \( (A) \)-rule on \( d \);
- if there exist an active point \( d \in \mathbb{D}_B \) and a formula \( \overline{(A)}\psi \in \text{REQ}_B(d) \) such that \( \overline{(A)}\psi \) is not fulfilled in \( B \) for \( d \), then apply the \( \overline{(A)} \)-rule on \( d \).

A tableau \( T \) for \( \varphi \) is *final* if and only if every branch \( B \) of \( T \) is closed or blocked.

**Theorem 2.4.1** (Termination). Let \( T \) be a final tableau for a PNL formula \( \varphi \) and \( B \) be a branch of \( T \). We have that \( |B| \leq (2 \cdot |\varphi| \cdot 2^{|\varphi|+1}) \cdot (2 \cdot |\varphi| \cdot 2^{|\varphi|+1} - 1)/2 \).

**Proof.** By the very same argument of Lemma 2.3.8, for any branch \( B \), \( |\mathbb{D}_B| \leq 2 \cdot |\varphi| \cdot 2^{|\varphi|+1} \). As we have exactly one node for any interval over \( \mathbb{D}_B \), the length of \( B \) is at most \((2 \cdot |\varphi| \cdot 2^{|\varphi|+1}) \cdot (2 \cdot |\varphi| \cdot 2^{|\varphi|+1} - 1)/2 \). \( \square \)

**Theorem 2.4.2** (Soundness). Let \( T \) be a final tableau for a PNL formula \( \varphi \). If \( T \) features one blocked branch, then \( \varphi \) is satisfiable over all linear orders.

**Proof.** Let \( B \) be a blocked branch in \( T \). By construction, for every pair \( d, d' \in \mathbb{D}_B \), with \( d < d' \), there exists one and only one node \( n \) in \( B \) labeled with \( (\{ d, d' \}, A) \). Let \( L = (\{ [d, d'] \}, I(\mathbb{D}_B), \mathcal{L}_B) \), where, for every pair \( d, d' \in \mathbb{D}_B \), we put \( \mathcal{L}(d, d') = A \). Since \( B \) is complete, it immediately follows that \( L \) is a finite LIS. Since \( B \) is blocked, all its active points are fulfilled in \( B \) and thus in \( L \). Hence, every interval-tuple \( (\{ A \}, A') \) in \( L \) is fulfilled. This allows us to conclude that \( L \) is a pseudo-model for \( \varphi \). From Lemma 2.3.7 and Theorem 2.2.7, it immediately follows that \( \varphi \) is satisfiable. \( \square \)

**Theorem 2.4.3** (Completeness). Let \( \varphi \) be a PNL formula which is satisfiable over the class of all linear orders. Then, there exists a final tableau for \( \varphi \) with at least one blocked branch.

**Proof.** First, by Theorem 2.2.7, we have that if \( \varphi \) is satisfiable, then there exists a fulfilling LIS \( L = (\{ [d, d] \}, I(\mathbb{D})) \), \( \mathcal{L}_B \) for it. Next, we show that such a fulfilling LIS can be exploited to construct a final tableau \( T \) for \( \varphi \), consisting of single blocked branch \( B \). Formally, by an induction on the
number \( i \) of expansion steps, we prove that for all \( d \in D_B, d \notin D \) and \( \operatorname{REQ}_B(d) = \operatorname{REQ}^+\{d\} \), and for all \( d, d' \in D_B \), if \( d < d' \) in \( D \), then \( d < d' \) in \( D_B \) and if there exists a node \( n \) in \( B \) labeled with \( [(d, d'), A] \), then \( A = \mathcal{L}([(d, d')]) \). The base case is straightforward. Since \( \mathcal{L} \) is a LIS for \( \varphi \), there exist two points \( d, d' \in D \) such that \( \varphi \in \mathcal{L}([(d, d')]) \). We start with an initial tableau \( T_0 \), consisting of a single branch \( B_0 \) whose unique node \( n_0 \) is labeled with \( [(d, d'), A] \), with \( \operatorname{REQ}_B(d) = \operatorname{REQ}^+\{d\} \), \( \operatorname{REQ}_{B_0}(d') = \operatorname{REQ}^+\{d'\} \), and \( A = \mathcal{L}([(d, d')]) \). Let \( T_i \) be the tableau generated at the \( i \)-th step of the expansion process and let \( B_i \) be its unique branch consisting of \( i \) nodes. We expand it as follows:

- If the Fill-in rule is applicable, then there exists a pair of points \( d, d' \in D_B \), such that there exists no node in \( B_i \) labeled with \( [(d, d')] \). By the inductive hypothesis, \( d, d' \in D \). We expand \( B_i \) with a new node \( n \) labeled with \( [(d, d')] \).

- If the \( (A) \)-rule is applicable, then there exist an active point \( d \in D_B \) and a formula \( (A) \psi \in \operatorname{REQ}_B(d) \), which is not fulfilled in \( B_i \) for \( d \). By the inductive hypothesis, \( d \in D \). Since \( \mathcal{L} \) is fulfilling, there exists \( d' \in D \), with \( d' > d \), such that \( \psi \in \mathcal{L}([(d, d')]) \). Since \( (A) \psi \) is not fulfilled in \( B_0 \) for \( d \) and \( B_0 \) is complete (the Fill-in rule is not applicable), \( d' \notin D_B \). We add a new point \( d' \) to \( D_B \) in such a way that for all \( d'' \in D_B \), if \( d'' < d' \) in \( D \), then \( d'' < d' \) in \( D_B \). \( d' \in D_B \), \( d' < d'' \) in \( D_B \), otherwise, and we apply the \( (A) \)-rule to expand \( B_i \) with a new node \( n = [(d, d')] \).

- The case of the \( (\overline{A}) \)-rule is completely symmetric, and thus its description is omitted.

By Theorem 2.4.1, the expansion of the tableau (branch) terminates in a finite (bounded) number of steps. Since no contradiction is introduced by any of the above steps, the final branch \( B \) is necessarily blocked.

### 2.4.2 A tableau-based decision procedure for PNL over dense linear orders

The tableau for the dense case can be obtained from that for the general case by enlarging the set of rules and revising the notion of blocked branch. First, we add the following rule:

**4. Dense-rule:** If there exist two consecutive points \( d_1, d_{i+1} \in D_B \) which are not covered, we proceed as follows. If there is not an interval-tuple \( \langle \operatorname{REQ}_B(d), A, S \rangle \) for some \( S \in \operatorname{REQ}_\varphi \) and some atom \( A \in A_\varphi \), we close the branch \( B \). Otherwise, let \( \langle \operatorname{REQ}_B(d), (d_1, A, S) \rangle \) be such an interval tuple. We expand \( B \) with a node \( n \), labeled with \( \langle [d_1, d], A \rangle \), such that \( \operatorname{REQ}_B(n)(d) = S \) and \( D_B = D_B \cup \{d_1 < d < d_{i+1}\} \).

Then, we revise the notion of blocked branch to take the Dense-rule into account. Given a non-closed branch \( B \), we say that \( B \) is blocked if (i) \( B \) is complete, (ii) all active points \( d \in D_B \) are fulfilled in \( B \), and (iii) the Dense-rule is not applicable to \( B \) anymore. As in the general case, we start from an initial tableau for \( \varphi \) and we apply the expansion rules to all the non-blocked and non-closed branches \( B \). The expansion strategy is the following one:

- apply the Fill-in rule until it generates no new nodes in \( B \);
- if there exists an active point \( d \in D_B \) for which there exists \( (A) \psi \in \operatorname{REQ}_B(d) \) which is not fulfilled in \( B \) for \( d \), then apply the \( \langle A \rangle \)-rule on \( d \);
- if there exists an active point \( d \in D_B \) for which there exists \( (\overline{A}) \psi \in \operatorname{REQ}_B(d) \) which is not fulfilled in \( B \) for \( d \), then apply the \( (\overline{A}) \)-rule on \( d \);
- apply the Dense-rule to \( B \).

A tableau \( T \) for \( \varphi \) is final if and only if every branch \( B \) of \( T \) is closed or blocked.

**Theorem 2.4.4 (Termination).** Let \( T \) be a final tableau for a PNL formula \( \varphi \) and \( B \) be a branch of \( T \). We have that \( |B| \leq (4 \cdot |\varphi| \cdot 2^{|\varphi|+1} - 1) \cdot (4 \cdot |\varphi| \cdot 2^{|\varphi|+1} - 2)/2 \).

**Proof.** As in the general case, \( 2 \cdot |\varphi| \cdot 2^{|\varphi|+1} \) is the maximum number of points that may be introduced by the application of the \( (A) \)-rule and the \( (\overline{A}) \)-rule. Since the Dense-rule introduces at most one point in between any two given points (introduced by the \( (A) \)-rule and the \( (\overline{A}) \)-rule),
it may add at most \(2 \cdot |\varphi| \cdot 2^{|\varphi|+1} - 1\) additional points. As we have exactly one node for any interval over \(D_B\), the length of \(B\) is at most \(4 \cdot |\varphi| \cdot 2^{|\varphi|+1} - 1 \cdot (4 \cdot |\varphi| \cdot 2^{|\varphi|+1} - 2)/2\).

\[\text{Theorem 2.4.5 (Soundness). Let } \mathcal{T} \text{ be a final tableau for a PNL formula } \varphi. \text{ If } \mathcal{T} \text{ features one blocked branch, then } \varphi \text{ is satisfiable over dense linear orders.}\]

\[\text{Proof. Let } B \text{ be a blocked branch in } \mathcal{T}. \text{ By the same argument of Theorem 2.4.2, we can prove that } L = \langle (D_B, \mathbb{I}(D_B)), \mathcal{L}_B \rangle \text{ is a pseudo-model for } \varphi. \text{ Moreover, since the Dense-rule is not applicable to the blocked branch } B, L \text{ is covered. From Lemma 2.3.11 and Theorem 2.2.7, it immediately follows that } \varphi \text{ is satisfiable.}\]

\[\text{Theorem 2.4.6 (Completeness). Let } \varphi \text{ be a PNL formula which is satisfiable over the class of dense linear orders. Then, there exists a final tableau for } \varphi \text{ with at least one blocked branch.}\]

\[\text{Proof. The proof is essentially the same as that for all linear orders, except for the fact that we must include the Dense-rule among the set of expansion rules we may apply. We deal with it as follows:}\]

\[- \text{ if the Dense-rule is applicable, then there exist two points } d_i, d_{i+1} \in D_B, \text{ which are not covered. By the inductive hypothesis, } d_i, d_{i+1} \in D. \text{ Since } D \text{ is dense, there exists a point } d \in D \text{ such that } d_i < d < d_{i+1} \text{ in } D. \text{ The application of the Dense-rule results in the addition of a new point } d \text{ to } D_B, \text{ such that } d_i < d < d_{i+1} \text{ in } D_B. \text{ Then } \mathcal{R} \text{ is applicable to the new point } d \text{ and for it we can prove that } \mathcal{R} \text{ is applicable to the new point } d \text{ and for it we can prove that } \varphi' \text{ is satisfiable over dense linear orders.}\]

\[\text{2.4.3 A tableau-based decision procedure for PNL over discrete linear orders}\]

As for the dense case proposed in Section 2.4.2, the tableau method for the discrete case is a modification of the one proposed for the general one. We equipped every branch \(B\) with a function \(\text{next}_B: D_B \rightarrow D_B \cup \{\text{undefined}\}\) that may prevent insertions between a point and its immediate successor in \(D_B\). To manage this new function, we have to rewrite the (\(A\))-rule and the (\(\overline{A}\))-rule.

1. \((A)\)-rule: let \(d_j \in D_B\) be a point for which there exists a non-fulfilled future existential formula \(\langle A \rangle \psi \in \mathcal{R} \mathcal{Q} \mathcal{B} (d_j)\) in \(B\) for it. Let \(A \psi\) be an atom with \(\psi \in \mathcal{A} \psi\) and \(S \in \mathcal{R} \mathcal{Q} \mathcal{A} \psi\) a set of requests such that \(\mathcal{R} \mathcal{Q} \mathcal{B} (d_j, A \psi, S)\) is an interval tuple (if such object does not exist then \(B\) is closed). Let \(F(d_j) = \{d'_j, \ldots, d_k\}\) be the set of all points \(d'_j \in D_B\) such that \(d'_j \geq d_j \) and \(\text{next}_B (d'_j) = \text{undefined}\). We take a fresh point \(d\) and we expand \(B\) with \(h = 4 \cdot (k - j + 1)\) nodes \(B \cdot n_1^{h} | n_2^{h} | n_3^{h} \cdots | n_i^{h} | n_j^{h} | n_k^{h} | n_i^{3} \) where for every \(1 \leq i \leq h\) and for every \(0 \leq i, j \leq 3\) we have \(n_1^{h} = n_1^{j} = \langle (d_j, d), \mathcal{A} \psi\rangle\), \(\mathcal{D} \mathcal{B} \mathcal{N} \mathcal{I} = \mathcal{D} \mathcal{B} \mathcal{N} \mathcal{I} = \mathcal{D} \mathcal{B} \cup \{d_{j+1} \leq d < d_{j+1}\}\) and \(\mathcal{R} \mathcal{Q} \mathcal{B} \mathcal{N} \mathcal{I} (d) = \mathcal{R} \mathcal{Q} \mathcal{B} \mathcal{N} \mathcal{I} (d) = S\). Finally we have:

\[\text{next}_B (d_{n+1}^{j} (d - 1)) = \begin{cases} d & \text{if } i \in \{2, 3\} \\ \text{undefined} & \text{otherwise} \end{cases}\]

\[\text{next}_B (d_{n+1}^{j} (d)) = \begin{cases} d + 1 & \text{if } i \in \{1, 3\} \\ \text{undefined} & \text{otherwise} \end{cases}\]

2. \((\overline{A})\)-rule: let \(d_j \in D_B\) be a point with a non-fulfilled past existential formula \(\langle \overline{A} \rangle \psi \in \mathcal{R} \mathcal{Q} \mathcal{B} (d_j)\) in \(B\) for it. Let \(A \psi\) be an atom with \(\psi \in \mathcal{A} \psi\) and \(S \in \mathcal{R} \mathcal{Q} \mathcal{A} \psi\) a set of requests such that \(\mathcal{R} \mathcal{Q} \mathcal{B} (d_j, A \psi, S)\) is an interval tuple (if such a pair does not exist then \(B\) is closed). Let \(F(d_j) = \{d'_j, \ldots, d_k\}\) be the set of all points \(d'_j \in D_B\) such that \(d'_j \geq d_j \) and \(\text{next}_B (d'_{j-1}) = \text{undefined} \) or \(d'_j = d_0\) then we take a fresh point \(d\) and we expand \(B\) with \(h = k + 1\) nodes \(B \cdot n_1^{h} | n_2^{h} | n_3^{h} \cdots | n_i^{h} | n_j^{h} | n_k^{h} | n_i^{3} \) where for every \(0 \leq i < h\) and for every
0 ≤ i, j ≤ 3 we have \( n_i^1 = n_i^j = ([d, d_i], A_\psi) \), \( D_{B \cdot n_i^1} = D_{B \cdot n_i^j} = D_B \cup \{d_{i-1} < d < d_i\} \) and \( \text{REQ}_{B \cdot n_i^1}(d) = \text{REQ}_{B \cdot n_i^j}(d) = S \). Finally we define:

\[
\text{next}_{B \cdot n_i^j}(d-1) = \begin{cases} 
  d & \text{if } i \in \{2, 3\} \\
  \text{undefined} & \text{otherwise}
\end{cases}
\]

\[
\text{next}_{B \cdot n_i^j}(d) = \begin{cases} 
  d+1 & \text{if } i \in \{1, 3\} \\
  \text{undefined} & \text{otherwise}
\end{cases}
\]

The Fill-in rule remains the same for this tableau. As in the case of the tableau proposed in Section 2.4.2, to ensure the safety property of the resulting model we have to introduce two new rules.

1. **Discrete-future rule**: let \( d \in D_B \) be a point which is unique in \( D_B \), with an immediate successor \( d' \) in \( D_B \) which is not fulfilled and \( \text{next}_B(d) = \text{undefined} \). Let \( \langle \text{REQ}_B(d), A, S \rangle \) and \( \langle \text{Req}_B(d'), A', S \rangle \) be two interval-tuples (if such a pair does not exist, then \( B \) is closed). let \( d'' \) be a fresh point and let \( n_1 = ([d, d''], A) \) and \( n_2 = ([d'', d'), A') \) be two nodes. We apply the expansion \( B \cdot n_1 \cdot n_2 \cdot n_1' \cdot n_2' \), where \( n_1 = n_1', \ n_2 = n_2' \). Moreover, we define \( D_{B \cdot n_1 \cdot n_2} = D_{B \cdot n_1' \cdot n_2'} = D_B \cup \{d < d'' < d'\} \) and \( \text{REQ}_{B \cdot n_1 \cdot n_2}(d'') = \text{REQ}_{B \cdot n_1' \cdot n_2'}(d'') = S \). Finally, we define \( \text{next}_B n_1 \cdot n_2(d) = \text{next}_B n_1 \cdot n_2'(d') = d'' \) and \( \text{next}_B n_1 \cdot n_2'(d'') = \text{undefined} \).

2. **Discrete-past rule**: let \( d \in D_B \) be a point which is unique in \( D_B \), with an immediate predecessor \( d' \) in \( D_B \) which is not fulfilled and \( \text{next}_B(d') = \text{undefined} \). Let \( \langle S, A, \text{Req}_B(d) \rangle \) and \( \langle \text{REQ}(d'), A', S \rangle \) two interval-tuples (if such a pair does not exist, then \( B \) is closed). Let \( d'' \) be a fresh point and \( n_1 = ([d'', d], A) \) and \( n_2 = ([d', d''], A) \) be two nodes. We apply the expansion \( B \cdot n_1 \cdot n_2 \cdot n_1' \cdot n_2' \) where \( n_1 = n_1', \ n_2 = n_2' \). Moreover, we define \( D_{B \cdot n_1 \cdot n_2} = D_{B \cdot n_1' \cdot n_2'} = D_B \cup \{d < d'' < d'\} \) and \( \text{REQ}_{B \cdot n_1 \cdot n_2}(d'') = \text{REQ}_{B \cdot n_1' \cdot n_2'}(d'') = S \). Finally, we define \( \text{next}_B n_1 \cdot n_2(d) = \text{next}_B n_1 \cdot n_2'(d'') = d, \text{next}_B n_1 \cdot n_2'(d'') = \text{undefined} \).

We extend the notion of active point, while the blocking condition remains the same as the one proposed in Section 2.4.1. A point \( d \in D_B \) is active on \( B \) if and only if there exists an active node \( n \) in \( B \) for which \( n = ([d, d'], A) \) or \( n = ([d', d], A) \) or for some unique point \( d' \) in \( D_B \) we have \( d = \text{next}_B(d') \) or \( \text{next}(d) = d' \). The expansion of a tableau \( T \) is obtained by the application of the expansion rules to all the not-blocked and not-closed branches \( B \) of \( T \). The expansion strategy is the following:

- Apply the Fill-in rule until it generates new nodes in \( B \);
- if there exists an active point \( d \in D_B \) for which there exists \( \langle A \rangle \psi \in \text{Req}_B(d) \) that is not fulfilled in \( B \) for \( d \) then apply the \( \langle A \rangle \)-rule on \( d \);
- if there exists an active point \( d \in D_B \) for which there exists \( \langle \overline{A} \rangle \psi \in \text{Req}_B(d) \) that is not fulfilled in \( B \) for \( d \) then apply the \( \langle \overline{A} \rangle \)-rule on \( d \);
- apply the Discrete-future rule on \( B \);
- apply the Discrete-past rule on \( B \).

A tableau \( T \) for \( \psi \) is final if and only if every branch \( B \) of \( T \) is closed or blocked.

**Theorem 2.4.7.** Let \( T \) be a final tableau for a PNL formula \( \psi \) if \( T \) provides one blocked branch then \( \psi \) is satisfiable over discrete linear orders.

**Proof.** Since \( B \) is complete, by the construction applied in Theorem 2.4.2 we obtain that \( L = ([D_B, I(D_B)], L_B) \) is a pseudo model for \( \psi \), since a blocked branch implies that all the rules are no more applicable on \( B \). In particular, both the Discrete-future and the Discrete-past rules are no more applicable, then we have that for every unique point \( d \in D_B \) both its immediate predecessor (if any) and immediate successor (if any) are fulfilled and thus \( L \) is a safe pseudo model for \( \psi \). □
Theorem 2.4.8. Let \( \varphi \) be a PNL formula satisfiable over general linear orders then there exists a final tableau for \( \varphi \) with at least one blocked branch.

Proof. Let \( L = ( (D \sqcup (D)), L) \) be a fulfilling discrete LIS for \( \varphi \) we construct a complete blocked branch \( B \) for some tableau \( \mathcal{T} \). Since \( L \) is a fulfilling for \( \varphi \) we have that there exists two points \( d, d' \in D \) for which \( \varphi \in L([d, d']) \). We start with an initial tableau \( \langle [d_0, d_1], A \rangle \) with \( \text{Req}_B(d_0) = \text{Req}(d), \text{Req}_B(d_1) = \text{Req}(d') \) and \( L([d_0, d_1]) = A \). At every step we guarantee that for every two consecutive point \( d_i, d_{i+1} \in D \) we proceed inductively as follows:

- if the Fill-in rule is applicable, we proceed as in the proof of Theorem 2.4.6;
- if the \( (A) \)-rule is applicable for some active node \( d \in D_B \) and some formula \( \langle A \rangle \psi \in \text{Req}_B(d) \) then by inductive hypothesis we have \( d \in D \). If there exists \( d' > d \) in \( D \) for which \( \psi \in L([d, d']) \), we have that \( d' \notin D_B \) (otherwise \( (A) \psi \) is fulfilled for \( d \) then we introduce the point \( d' \) in a position for which \( d' > d'' \) for every \( d'' \in D_B \) with \( d' > d'' \) in \( D \) and \( d'' < d'' \) for every \( d'' \in D_B \) with \( d'' < d'' \) in \( D \). We extend \( B \) with a node with the node \( n = ([d, d'], L([d, d'])) \) and \( \text{Req}_B(d') = \text{Req}_B(d'') \). Let \( \overline{d} \) be the immediate predecessor of \( d' \) in \( D \), if \( d \in D_B \) then we define next, \( \overline{d} = d' \). Finally let \( \overline{d} \) be the immediate successor if any of \( d' \) in \( D \) if \( \overline{d} \in D_B \) then we define next, \( d = \overline{d} \);
- if the \( (A) \)-rule is applicable we operate in a symmetric way of the \( (A) \)-rule;
- if the Discrete-future rule is applicable we have that there exists a unique point \( d \in D_B \) with an immediate successor \( d' \) in \( D_B \), for which next, \( d \) is undefined. By inductive hypothesis \( d' \) is not the immediate successor of \( d \) in \( D \) then there exists \( d'' \in D \) with \( d < d'' < d' \). We expand the branch \( B \) with two consecutive nodes \( n = ([d, d'], L([d, d''])) \) and \( n' = ([d'', d'), L([d'', d'])) \), moreover we put \( D_B \cdot n = D_B \cup \{d < d'' < d'\} \). Finally we define \( \text{Req}_B(n, n')(d'') = \text{Req}_B(d''), \) next, \( n, n'(d'') = d'' \) if \( d' \) is the immediate successor of \( d'' \) in \( D \) (next, \( n, n'(d'') = \text{undefined} \) otherwise).

At the end of this construction we have that the resulting branch \( B \) is blocked.

\[ \square \]

Theorem 2.4.9. For every final tableau \( \mathcal{T} \) for a PNL formula \( \varphi \) and for every branch \( B \) of \( \mathcal{T} \) we have that \( |B| \leq 2 \cdot (|\varphi| + 2) \cdot |\varphi| \cdot 2^{3|\varphi|+1} - 1) / 2 \)

Proof. We have to add to the maximum number of points \( 2 \cdot |\varphi| \cdot 2^{3|\varphi|+1} \) needed for guarantee all the possible interval tuples to the maximum number of successors \( 2 \cdot |\varphi| \cdot 2^{3|\varphi|+1} \) on which are required the fulfilling, then we have to add at most \( |\varphi| \cdot 2 \cdot |\varphi| \cdot 2^{3|\varphi|+1} \) for the fulfilling of these points. Finally we have that \( |D_B| \leq 2 \cdot (|\varphi| + 2) \cdot |\varphi| \cdot 2^{3|\varphi|+1} \) and thus \( |B| \leq (2 \cdot (|\varphi| + 2) \cdot |\varphi| \cdot 2^{3|\varphi|+1} - 1) / 2 \).

\[ \square \]

2.4.4 A tableau-based decision procedure for PNL over the integers

The tableau-based decision procedure for PNL over the integers is quite different from the ones proposed in the previous subsections. For every branch \( B \), we identify two points \( d_{\text{prefix}} \) and \( d_{\text{suffix}} \) in its finite linear order \( D_B \) defined as follows:

- \( d_{\text{prefix}} \) is the greatest point in \( D_B \) for which for every point \( d \in D_B \) such that \( d < d_{\text{prefix}} \) there exists a point \( d' \geq d_{\text{prefix}} \) with \( \text{Req}_B(d) = \text{Req}_B(d') \);
- \( d_{\text{suffix}} \) is the smallest point in \( D_B \) for which for every point \( d \in D_B \) such that \( d > d_{\text{suffix}} \) there exists a point \( d' \leq d_{\text{suffix}} \) with \( \text{Req}_B(d) = \text{Req}_B(d') \).

Recall that, given a PNL formula \( \varphi \), we denote by \( m \) the sum \( 2 \cdot f \cdot p + f + p \), where \( f \) (resp., \( p \)) is the number of \((A)\)-formulae (resp., \((\overline{A})\)-formulae) in \( CL(\varphi) \).

We say that a branch \( B \) is future fulfilling if for every point \( d \in D_B \) and every \( \langle A \rangle \psi \in \text{Req}_B(d) \) one of the following conditions hold:

- there exists a node \( n = ([d, d'], A) \) in \( B \) for which \( \psi \in A \);
2.4. Tableau-based decision procedures for PNL

- d is not unique in $D_B$ and there exists an interval tuple $\langle \text{Req}_B(d), A, S \rangle$ with $\psi \in A$ and $S = \text{Req}_B(d')$ for some $d' > d^\text{infix}_\text{start}$.
- d is unique in $D_B$ and there exist $p' \leq p + 1$ interval-tuples $(S_1, A_1, S), \ldots, (S_{p'}, A_{p'}, S)$ with:
  - $\psi \in A_1, S_1 = \text{Req}_B(d), S = \text{Req}_B(d')$ for some $d' > d^\text{infix}_\text{start}$ and for every $j$ we have $S_j = \text{Req}_B(d''')$ for some $d''' \in D_B$;
  - for every $\langle \overline{A} \rangle \psi \in S$ there exists $j$ for which $\psi \in A_j$;
  - if $S_j = S_1$ for some pair $i, j$ with $i \neq j$ then there exists two distinct points $d', d'' \in D_B$ with $\text{Req}_B(d') = \text{Req}_B(d'') = S_j$.

We say that a branch $B$ is past fulfilling if for every point $d \in D_B$ and every $\langle \overline{A} \rangle \psi \in \text{Req}_B(d)$ one of the following conditions hold:

- there exists a node $n = \langle [d', d], A \rangle$ in $B$ for which $\psi \in A$;
- d is not unique in $D_B$ and there exists an interval tuple $\langle S, A, \text{Req}_B(d) \rangle$ with $\psi \in A$ and $S = \text{Req}_B(d')$ for some $d' < d^\text{prefix}_\text{start}$.

We redefine the initial tableau for the integer case as a single node $\langle [d_0, d_1], A \rangle$ where $D_B = \{d_0, d_1\}$ and $\langle A \rangle \varphi \in \text{Req}_B(d_0)$. In order to deal with the particular case of $Z$ (and its subsets) we introduce two new rules:

1. **Future-rule:** let $d$ be the greatest point in $D_B$ and there exists an interval tuple $\langle \text{Req}_B(d), A, S \rangle$ for some atom $A$ and some set $S \in \text{Req}_B$ (if such interval tuple does not exist then $B$ is closed). Let $d'$ be a fresh point we expand $B$ with one node $n = \langle [d, d'], A \rangle$, moreover we define $D_{B,n} = D_B \cup \{d < d'\}$ and $\text{Req}_{B,n}(d') = S$;

2. **Past-rule:** let $d$ be the smallest point in $D_B$ and there exists an interval tuple $\langle S, A, \text{Req}_B(d) \rangle$ for some atom $A$ and some set $S \in \text{Req}_B$ (if such interval tuple does not exist then $B$ is closed). Let $d'$ be a fresh point we expand $B$ with one node $n = \langle [d', d], A \rangle$, moreover we define $D_{B,n} = D_B \cup \{d' < d\}$ and $\text{Req}_{B,n}(d') = S$;

Since the Fill-in rule remains essentially the same, as in the case of the tableau proposed in Section 2.4.1, we have to completely redefine the notion of blocked branch.

Given a branch $B$ that is not closed we say that $B$ is blocked if it is fulfilling or there exists $m = 2 \cdot p \cdot f + p + f + 1$ distinct points $d_1, \ldots, d_m$ in $D_B$ for which $\text{Req}_B(d_1) = \ldots = \text{Req}_B(d_m)$. We start with an initial tableau $\mathcal{T}$ for $\varphi$ and we apply the expansion rules to all the non-blocked and non-closed branches $B$ of $\mathcal{T}$. The expansion strategy is the following:

- apply the Fill-in rule until it generates new nodes in $B$;
- if $B$ is not past fulfilling neither future fulfilling apply both Future-rule and Past-rule on $B$;
- if $B$ is past fulfilling and not future fulfilling apply the Future-rule to it;
- if $B$ is future fulfilling and not past fulfilling apply the Past-rule to it;

A tableau $\mathcal{T}$ for $\varphi$ is final if and only if every branch $B$ of $\mathcal{T}$ is closed or blocked.

**Theorem 2.4.10.** Let $\mathcal{T}$ be a final tableau for a PNL formula $\varphi$ if $\mathcal{T}$ provides one fulfilling branch then $\varphi$ is satisfiable over $Z$ or over one of its subsets.

**Proof.** Let $B$ be a fulfilling branch in $\mathcal{T}$. We build a finite LIS $L = \langle (D_B, \mathcal{L}(D_B)), \mathcal{L}_B \rangle$ as in Theorem 2.4.2. At the end of such a construction, we define $R_r = \{\text{Req}_B(d) \mid d > d^\text{infix}_\text{start}\}$, $R_l = \text{...}$. The...
Theorem 2.4.11. Let \( \varphi \) be a PNL formula satisfiable over \( \mathbb{Z} \) or over one of its subsets. Then, there exists a final tableau for \( \varphi \) with at least one fulfilling branch.

Proof. Since \( \varphi \) is satisfiable, there exists a finite LIS \( \mathcal{L} = (\langle D, \ll \rangle, \mathcal{L}) \), and four sets \( \mathcal{I}, \mathcal{J}, \mathcal{J}_r, \mathcal{J}_l \) which satisfy the conditions of Theorem 2.3.23. Let \( m_r = [\mathcal{J}_r \cup \mathcal{J}] \) and \( \{S_1, \ldots, S_m\} \) be an arbitrary order all the elements in \( \mathcal{J}_r \cup \mathcal{J} \) and \( m_l = [\mathcal{J}_l \cup \mathcal{J}] \) and \( \{S_1, \ldots, S_m\} \) be an arbitrary order all the elements in \( \mathcal{J}_l \cup \mathcal{J} \). We build a finite structure \( \mathcal{L}' = (\langle D', \ll \langle D' \rangle \rangle, \mathcal{L}') \) as follows:

- \( D' = D \cup \{d_1, \ldots, d_m, d_1', \ldots, d_m', d_1'' \ldots, d_m'' \} \) and for every \( i \) we have \( \text{REQ}_{\mathcal{L}'}(d_i) = \text{REQ}^{\mathcal{L}'}(d_i') = S_1 \) and \( \text{REQ}^{\mathcal{L}'}(d_i') = \text{REQ}^{\mathcal{L}'}(d_i'') = S_1 \);
- let \( d, d' \) be respectively the smallest and the greatest point in \( D \) we define \( D' = D \cup \{d_1 < \ldots < d_m < d_1' < \ldots < d_m' < d_1'' \ldots < d_m'' < d' \} \); where
- \( \mathcal{L}'[\langle d, d' \rangle] = \mathcal{L}[\langle d, d' \rangle] \) for every \( d, d' \in D \) and for every pair \( d, d' \in D' \) for which \( d \notin D \) or \( d' \notin D \) by consistency condition of Theorem 2.3.23 we have that there exists an interval-tuple \( \langle S, A, S' \rangle \) with \( \text{REQ}^{\mathcal{L}'}(d) = S \) and \( \text{REQ}^{\mathcal{L}'}(d') = S' \), then we put \( \mathcal{L}'[\langle d, d' \rangle] = A \).

We build a fulfilling branch \( B \) for some tableau \( \mathcal{T} \). By construction we have that there exists a point \( d \in D' \) for which \( \langle A, \varphi \rangle \in \text{REQ}^{\mathcal{L}'}(d) \) or \( \langle A \rangle \varphi \in \text{REQ}^{\mathcal{L}'}(d) \). Suppose without loss of generality that \( \langle A \rangle \varphi \in \text{REQ}^{\mathcal{L}'}(d) \) and \( d' \) is the immediate successor of \( d \) in \( D \). Start with an initial tableau \( \langle d_0, d_1, A \rangle \) with \( \text{REQ}_{B}(d_0) = \text{REQ}(d), \text{REQ}_{B}(d_1) = \text{REQ}(d') \) and \( \mathcal{L}[\langle d, d' \rangle] = A \).

We proceed inductively as follows:

- if the Fill-in rule is applicable, then there exists a pair of points \( d, d' \in D_B \) for which does not exists a node in \( B \) labeled with the interval \( \langle d, d' \rangle \), then we expand \( B \) with node \( n = \langle d, d' \rangle, \mathcal{L}[\langle d, d' \rangle] \);
- if the Future-rule is applicable, let \( d \in D \) be the greatest point in \( D_B \), then we take \( d' \) as the immediate successor of \( d \) in \( D' \). Since \( L' \) is a LIS we have that \( \langle \text{REQ}^{\mathcal{L}'}(d), \mathcal{L}'[\langle d, d' \rangle] \rangle \), \( \text{REQ}^{\mathcal{L}'}(d') \) is an interval tuple. We extend \( B \) with the node \( \langle d, d', \mathcal{L}'[\langle d, d' \rangle] \rangle \), and we put \( \text{REQ}_{B}(d') = \text{REQ}^{\mathcal{L}'}(d') \) and \( D_B = D_B \cup \{d < d' \} \);
- if the Past-rule is applicable, we operate in a symmetric way of the Future-rule.

It is worth to notice that by conditions of Theorem 2.3.23, by the definition of fulfilling branch, and by construction we have that this expansion cannot consider a point \( d > d'_r \) and \( d < d'_l \). Moreover we have that since every set of requests occurs at most \( 2 \cdot p \cdot f + p + f + 1 \) times in \( L' \) then \( B \) is not blocked. Since \( D_B \subseteq D' \) we have that in the worst case \( D_B = D' \) and, since \( L' \) is fulfilling, we can conclude that \( B \) is fulfilling.

Theorem 2.4.12. For every final tableau \( \mathcal{T} \) for a PNL formula \( \varphi \) and for every branch \( B \) of \( \mathcal{T} \) we have that \( |B| \leq (2 \cdot p \cdot f + p + f) \cdot 2^{\omega_1} + 1 \cdot (2 \cdot p \cdot f + p + f) \cdot 2^{\omega_1} / 2 \)

Proof. By the behavior of the procedure we have that only a set of requests can occur \( 2 \cdot p \cdot f + p + f + 1 \) times in \( D_B \) the remaining can occur at most \( 2 \cdot p \cdot f + p + f \). We have that \( |D_B| \leq (2 \cdot p \cdot f + p + f) \cdot 2^{\omega_1} + 1 \) and thus \( |B| \leq (2 \cdot p \cdot f + p + f) \cdot 2^{\omega_1} + 1 \cdot (2 \cdot p \cdot f + p + f) \cdot 2^{\omega_1} / 2 \)

It is worth to notice that all tableau procedures proposed in Section 2.4.1, Section 2.4.2, Section 2.4.3, and Section 2.4.4 belong to the NEXPTIME complexity class, as formally stated by the following theorem.

Theorem 2.4.13. The satisfiability problem for PNL over all the considered classes of linear orders belongs to the NEXPTIME complexity class.

In Section 2.5, we will prove that the satisfiability problem for PNL over all considered classes of linear orders is actually NEXPTIME-complete.
2.5 Optimality of the proposed methods

We now provide a NEXPTIME lower bound for the complexity of the satisfiability problem for PNL by reducing to it the exponential tiling problem, which is known to be NEXPTIME-complete [9], [59].

Let us denote by $\mathbb{N}_m$ the set of natural numbers less than $m$ and by $\mathbb{N}(m)$ the grid $\mathbb{N}_m \times \mathbb{N}_m$. A domino system is a triple $\mathcal{D} = (\mathcal{C}, \mathcal{H}, \mathcal{V})$, where $\mathcal{C}$ is a finite set of colors and $\mathcal{H}, \mathcal{V} \subseteq \mathcal{C} \times \mathcal{C}$ are the horizontal and vertical adjacency relations. We say that $\mathcal{D}$ tiles $\mathbb{N}(m)$ if there exists a mapping $\tau : \mathbb{N}(m) \to \mathcal{C}$ such that, for all $(x, y) \in \mathbb{N}(m)$:

1. if $\tau(x, y) = c$ and $\tau(x + 1, y) = c'$, then $(c, c') \in \mathcal{H}$;
2. if $\tau(x, y) = c$ and $\tau(x, y + 1) = c'$, then $(c, c') \in \mathcal{V}$.

The exponential tiling problem consists in determining, given a natural number $n$ and a domino system $\mathcal{D}$, whether $\mathcal{D}$ tiles $\mathbb{N}(2^n)$ or not. Proving that the satisfiability problem for PNL is NEXPTIME-hard can be done by encoding the exponential tiling problem with a formula $\varphi(\mathcal{D})$, of length polynomial in $n$, which uses propositional letters to represent positions in the grid and colors, and by showing that $\varphi(\mathcal{D})$ is satisfiable if and only if $\mathcal{D}$ tiles $\mathbb{N}(2^n)$. Such a formula consists of three main parts. The first part imposes a sort of locality principle; the second part encodes the domino system; the third part imposes that every point of the grid is tiled by exactly one color and that the colors respect the adjacency conditions. Intervals are exploited to express relations between pairs of points.

**Theorem 2.5.1.** The satisfiability problem for PNL over all classes of linear orders is NEXPTIME-hard.

**Proof.** Given a domino system $\mathcal{D} = (\mathcal{C}, \mathcal{H}, \mathcal{V})$, we build a PNL formula $\varphi$, of length polynomial in $n$, that is satisfiable if and only if $\mathcal{D}$ tiles $\mathbb{N}(2^n)$.

The models for $\varphi$ encode a tiling $\tau : \mathbb{N}(2^n) \to \mathcal{C}$ in the following way. First, we associate with every point $z = (x, y) \in \mathbb{N}(2^n)$ a $2n$-bit word $(z_{2n-1} z_{2n-2} \ldots z_1 z_0) \in \{0, 1\}^{2n}$ such that $x = \sum_{i=0}^{n-1} z_i 2^i$ and $y = \sum_{i=n}^{2n-1} y_i 2^{i-n}$. Pairs of points $[z, t]$ of $\mathbb{N}(2^n)$ are represented as intervals by means of the propositional letters $Z_i, \tau_i$, with $0 \leq i \leq 2n - 1$, as follows:

$$Z_i : z_i = 1; \quad \tau_i : \tau_i = 1.$$

Moreover, the colors of $z = (x, y)$ and $t = (x', y')$ are expressed by means of the propositional letters $Z_c, \tau_c$, with $c \in \mathcal{C}$, as follows:

$$Z_c : \tau(x, y) = c; \quad \tau_c : \tau(x', y') = c.$$

To ease the writing of the formula $\varphi$ encoding the tiling problem, we use the auxiliary propositional letters $Z^*_i$ (for $0 \leq i \leq 2n - 1$) and $ZH^*_i$ (for $n \leq i \leq 2n - 1$), with the following intended meaning:

$$Z^*_i : \text{for all } 0 < j < i, z_j = 1; \quad ZH^*_i : \text{for all } n \leq j < i, z_j = 1.$$

To properly encode the tiling problem, we must constrain the relationships among these propositional letters.

**Definition of auxiliary propositional letters.** As a preliminary step, we define the auxiliary propositional letters $Z^*_i$, with $0 \leq i \leq 2n - 1$, and $ZH^*_i$, with $n \leq i \leq 2n - 1$, as follows:

$$[\mathcal{A}] [\mathcal{A}] \left( Z^*_0 \land \bigwedge_{i=1}^{2n-1} (Z^*_i \leftrightarrow (Z^*_{i-1} \land Z^*_{i+1})) \right)$$

$$[\mathcal{A}] [\mathcal{A}] \left( ZH^*_n \land \bigwedge_{i=n+1}^{2n-1} (ZH^*_i \leftrightarrow (ZH^*_{i-1} \land Z^*_{i-1})) \right).$$

Let us call $\alpha$ the conjunction of the above two formulae.
2. Propositional Neighbourhood Logic over linear orders

Locality conditions. Then, we impose a sort of “locality principle” on the interpretation of the propositional letters. Given an interval \([z, t]\), we encode the position \(z = (x, y)\) (resp., \(t = (x', y')\)) and its color \(\tau(x, y)\) (resp., \(\tau(x', y')\)) by means of the propositional letters \(Z_i, Z_i^+, ZH_i^+,\) and \(Z_c\) (resp., \(T_i\) and \(T_c\)) by imposing the following constraints:

- all intervals \([z, w]\) starting in \(z\) must agree on the truth value of \(Z_i, Z_i^+, ZH_i^+,\) and \(Z_c\);
- for every pair of neighboring intervals \([z, t], [t, w]\), the truth value of \(T_i\) and \(T_c\) over \([z, t]\) must agree with the truth value of \(Z_i\) and \(Z_c\) over \([t, w]\).

From the above constraints, it easily follows that all intervals \([w, t]\) ending in \(t\) must agree on the truth value of \(T_i\) and \(T_c\).

Such constraints are encoded by the conjunction of the following formulae (let us call it \(\beta\)):

\[
\bigwedge_{i=0}^{2n-1} (\langle A \rangle Z_i \rightarrow [A]Z_i) \land \\
\bigwedge_{i=0}^{2n-1} ([A]\langle A \rangle Z_i \rightarrow [A]Z_i)
\]

\[
\bigwedge_{i=0}^{2n-1} (\langle A \rangle Z_i^+ \rightarrow [A]Z_i^+) \land \\
\bigwedge_{i=0}^{2n-1} ([A]\langle A \rangle Z_i^+ \rightarrow [A]Z_i^+)
\]

\[
\bigwedge_{i=n}^{2n-1} (\langle A \rangle ZH_i^+ \rightarrow [A]ZH_i^+) \land \\
\bigwedge_{i=n}^{2n-1} ([A]\langle A \rangle ZH_i^+ \rightarrow [A]ZH_i^+)
\]

\[
\bigwedge_{c \in C} ([A]\langle A \rangle Z_c \rightarrow [A]Z_c) \land \\
\bigwedge_{c \in C} ([A][\langle A \rangle Z_c \rightarrow [A]Z_c)
\]

\[
\bigwedge_{i=0}^{2n-1} ([A]\langle T_i \rightarrow [A]Z_i) \land \\
\bigwedge_{i=0}^{2n-1} ([A]T_i \rightarrow [A]Z_i)
\]

\[
\bigwedge_{c \in C} ([A]\langle T_c \rightarrow [A]Z_c) \land \\
\bigwedge_{c \in C} ([A][\langle T_c \rightarrow [A]Z_c)
\]

Encoding of the grid. Next, we must guarantee that every point \(z = (x, y) \in \mathbb{N}(2^n)\), with the exception of the upper-right corner \((2^n - 1, 2^n - 1)\), has a “successor” \(t = (x', y')\), that is, if \(x \neq 2^n - 1\), then \((x', y') = (x + 1, y);\) otherwise \((x = 2^n - 1), (x', y') = (0, y + 1)\). Note that, thanks to our encoding of \(z\) and \(t\), the binary encoding of the successor of \(z\) is equal to the binary encoding of \(z\) incremented by 1. Such a successor relation can be encoded as follows. Given two 2n-bit words \(z = \sum_{i=0}^{2n-1} z_i 2^i\) and \(t = \sum_{i=0}^{2n-1} t_i 2^i\), we have that \(t = z + 1\) if and only if there exists some \(0 \leq i < 2n - 1\) such that:

1. \(z_i = 0\) and, for all \(i < j, z_i = 1\);
2. \(t_j = 1\) and, for all \(i < j, t_i = 0\);
3. for all \(j < k \leq 2n - 1, z_k = t_k\).

It is easy to show that, for every \(i, with 0 \leq i \leq 2n - 1\), we can write \(t_i = z_i \oplus \bigwedge_{k < i} z_k\), where \(\oplus\) denotes the exclusive or. Taking advantage of this fact, the successor relation can be expressed by the following formula (let us call it \(\gamma\)):

\[
[A] \left( \langle A \rangle \neg (Z_{2n-1} \land Z_{2n-1}) \rightarrow \langle A \rangle \bigwedge_{i=0}^{2n-1} (T_i \leftrightarrow (Z_i \oplus Z_i^+)) \right).
\]

Furthermore, the left conjunct of the following formula (let us call it \(\delta\)) encodes the initial point \((0, 0)\) of the grid, while the right one encodes the final point \((2^n - 1, 2^n - 1)\):

\[
\langle A \rangle \bigwedge_{i=0}^{2n-1} \neg Z_i \land \langle A \rangle \bigwedge_{i=0}^{2n-1} Z_i.
\]
Grid coloring. To complete the reduction, we must properly define the tiling of the grid. To this end, we preliminarily need to express the relations of right (horizontal) neighborhood and upper (vertical) neighborhood over the grid. We have that the following formulas $\psi_H$ (resp., $\psi_V$) holds over any interval $[z, t]$ such that $t$ is the right (resp. upper) neighbor of $z$ in $\mathbb{N}(2^n)$:

$$\psi_H := \bigwedge_{i=n}^{2n-1} (Z_i \leftrightarrow T_i) \land \bigwedge_{i=0}^{n-1} (T_i \leftrightarrow (Z_i \oplus Z_i'))$$

$$\psi_V := \bigwedge_{i=0}^{n-1} (Z_i \leftrightarrow T_i) \land \bigwedge_{i=n}^{2n-1} (T_i \leftrightarrow (Z_i \oplus Z_i'))$$

By using $\psi_H$ and $\psi_V$, we can impose the adjacency conditions by means of the following formula (let us call $c$):

$$\mathcal{A}(A) \left( (\psi_H \rightarrow \bigvee_{(c, c') \in H} Z_c \land T_{c'}) \land (\psi_V \rightarrow \bigvee_{(c, c') \in V} Z_c \land T_{c'}) \right).$$

The fact that every point is tiled by exactly one color can be forced by the following formula (let us call it $\zeta$):

$$\mathcal{A}(A) \left( \bigvee_{c \in C} Z_c \land \bigvee_{c \in C} T_c \right),$$

where $\bar{V}$ is a generalized exclusive or which is true if and only if exactly one of its arguments is true.

Let us define $\phi$ as the conjunction $\alpha \land \beta \land \gamma \land \delta \land c \land \zeta$. The length of $\phi$ is polynomial in $n$ as requested. It remains to show that $\phi$ is satisfiable if and only if $D$ tiles $\mathbb{N}(2^n)$. As for the implication from left to right, if a correct tiling exists, then let $\mathcal{M} = (D, \prec)$ be a linear order such that:

1. $D = \{d_0, d_1\} \cup \mathbb{N}(2^n) \cup \{d_\top\}$;
2. $d_0 < d_1 < (x, y) < d_\top$, for every $(x, y) \in \mathbb{N}(2^n)$;
3. given two points $(x, y)$ and $(x', y')$ of $\mathbb{N}(2^n)$, $(x, y) < (x, y')$ if and only if $y < y' \lor (y = y' \land x < x')$.

Notice that we take as the domain of the interval structure the set of elements of the grid extended with the elements $d_0, d_1, \text{ and } d_\top$. The elements $d_0, d_1$ define the initial interval $[d_0, d_1]$ over which our formula will be interpreted. The element $d_\top$ is the right endpoint of the only interval having the last point of the grid as its left endpoint, namely, $[\mathbb{N}(2^n - 1, 2^n - 1, \mathbb{N})]$.

As for the valuation $V$, for any interval $[z, t]$, with $z = (x, y), t = (x', y')$, and $z, t \in \mathbb{N}(2^n)$, $Z_i \in V([z, t])$ if and only $z_i = 1$ and $T_i \in V([z, t])$ if and only $t_i = 1$. Moreover, $Z_c \in V([z, t])$ (resp., $T_c \in V([z, t])$) if and only if $\tau(x, y) = c$ (resp., $\tau(x', y') = c$). Whenever, the left (resp., right) endpoint of an interval does not belong to $\mathbb{N}(2^n)$, the valuation of the propositional letters $Z_i$ and $Z_c$ (resp., $T_i$ and $T_c$) over the interval is arbitrary. It is not difficult to prove that $M = (\mathcal{D}, \mathcal{I}(\mathcal{D}^-), \mathcal{V})$ is a model of $\phi$, that is, $M, [d_0, d_1] \models \phi$.

Conversely, let $M = (\mathcal{D}, \mathcal{I}(\mathcal{D}^-), \mathcal{V})$ be a model for $\phi$, that is, $M, [d_0, d_1] \models \phi$. To provide a tiling of $\mathbb{N}(2^n)$, we first define a function $f : \mathbb{N}(2^n) \rightarrow D$ that associates a point $d \in D$ with every point $(x, y) \in \mathbb{N}(2^n)$ in such a way that:

1. the binary representation of $(x, y)$ coincides with the sequence of truth values of the propositional letters $Z_{2n-1}, Z_{2n-2}, \ldots, Z_0$ over the intervals $[d, d'] \in \mathcal{I}(\mathcal{D})$;
2. for every $(x, y), (x', y') \in \mathbb{N}(2^n)$, $(x, y) < (x', y')$ if and only if $f(x, y) < f(x', y')$.

The formula $\phi$ ensures that such a function exists. Note that the definition of $f$ guarantees commutativity: by moving first one step right and then one step up on the grid one reaches the same point that can be reached by moving first one step up and then one step right. On the basis of such a function, we define the tiling $\tau(x, y) = c$, where $c$ is the unique element of $C$ such that
M, [f(x, y), d′] ⊨ Zc, for every d′ > f(x, y). It is not difficult to prove that τ defines a tiling of \( N(2^n) \). Finally it is worth to notice that we only assume that \( \varphi \) is interpreted over some kind of linear order and thus this result holds for all the considered classes of linear orders.

From Theorems 2.4.13 and 2.5.1, we have the following corollary.

**Corollary 2.5.2.** The satisfiability problem for PNL over all considered classes of linear orders is NEXPTIME-complete.
In this chapter, 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.

### 3.1 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 \((\mathbb{N}, <)\) and we allow any node to have infinitely many (possibly, uncountably many) successors.

A directed graph is a pair \(G = (G, S)\), 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 \(G\) is a sequence of nodes \(g_1, g_2, \ldots, g_n\), with \(n \geq 2\), such that \(S(g_i, g_{i+1})\) for \(i = 1, \ldots, n - 1\). Infinite \(S\)-sequences can be defined analogously. A path \(\rho\) in \(G\) 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, there is a strictly increasing path \(\rho\) from \(g_i\) to \(g_j\). Now, we can say that \(g_i\) is \(S\)-reachable from \(g_j\) 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 3.1.1.** A tree is a directed graph \(T = (T, S)\). 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 \(T = (T, S)\), 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 \(\rho\) in \(T\), \((\rho, <)\) is isomorphic to \((\mathbb{N}, <)\). Given a tree \(T = (T, S)\) and the corresponding strict partial ordering \((T, <)\), 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 \(I(T)\). The pair \((T, I(T))\) is called an interval structure. For every pair of intervals \([t_i, t_j]\), \([t'_i, t'_j]\) \(\in I(T)\), 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., \(t'_i = t_j\)). The vocabulary of Right Propositional Neighborhood Logic \([20]\) (RPNL for short) consists of a set \(AP\) of propositional letters, the Boolean

---

1It is easy to see that, as far as RPNL is concerned, such trees are indistinguishable from finitely branching trees.
connectives $\neg$ and $\lor$, and the modal operator $\langle A \rangle$. \textit{Formalae} of RPNL, denoted by $\varphi, \psi, \ldots$, are recursively defined by the following grammar:

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

The other Boolean connectives, the logical constants $\top$ (true) and $\bot$ (false), and the dual modal operator $\langle A \rangle$ are defined as usual. We denote by $|\varphi|$ the length of $\varphi$, that is, the number of symbols in $\varphi$ (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. Formalae of the forms $\langle A \rangle \varphi$ or $\langle A \rangle \psi$ are called \textit{temporal formulae} (from now on, we identify $\neg \langle A \rangle \psi$ with $\langle A \rangle \neg \psi$ and $\neg \langle A \rangle \psi$ with $\langle A \rangle \neg \psi$); formalae of the form $\langle A \rangle \psi$ are called \textit{temporal requests}.

A \textit{model} for an RPNL formula is a pair $M = (\langle T, I(\langle T \rangle) \rangle, \mathcal{V})$, where $\langle T, I(\langle T \rangle) \rangle$ is an interval structure and $\mathcal{V} : I(\langle T \rangle) \to 2^{AP}$ is a \textit{valuation function} assigning to every interval the set of propositional letters true on it. Given a model $M = (\langle T, I(\langle T \rangle) \rangle, \mathcal{V})$ and an interval $[d_1, d_3] \in I(\langle T \rangle)$, the semantics of RPNL is recursively defined by the \textit{satisfaction relation} $\models$: as follows:

- for every $p \in AP$, $M, [t_1, t_2] \models p$ iff $p \in \mathcal{V}([t_1, t_2]);$
- $M, [t_1, t_2] \models \neg \psi$ iff $M, [t_1, t_2] \not\models \psi$;
- $M, [t_1, t_2] \models \psi_1 \lor \psi_2$ iff $M, [t_1, t_2] \models \psi_1$ or $M, [t_1, t_2] \models \psi_2$;
- $M, [t_1, t_2] \models \langle A \rangle \psi$ iff $\exists t_k \in T$, with $t_k > t_2$, such that $M, [t_1, t_k] \models \psi$.

We place ourselves in the most general setting and we do not impose any constraint on the valuation function. In particular, given an interval $[d_1, d_3]$, it may happen that $p \in \mathcal{V}([d_1, d_3])$ and $p \not\in \mathcal{V}([d_1', d_3'])$ for all intervals $[d_1', d_3']$ (properly included in $[d_1, d_3]$).

### 3.1.1 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 $\varphi_1 \equiv \langle A \rangle [A]p \land \langle A \rangle [A]\neg p \land [A]\langle A \rangle \top$ 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 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, $\varphi_1$ 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 3.1. 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 3.2. 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
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).

3.2 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 \( \varphi \) be an RPNL formula to check for satisfiability. Let \( \text{CL}(\varphi), \text{TF}(\varphi) \)
defined as in Definition 2.2.1 and Definition 2.2.2 respectively. The definition of a \( \varphi \)-atom \( A \) is the same
of Definition 2.2.3 as well as \( A_{\varphi} \). Moreover for any \( \varphi \)-atom \( A \), let \( \text{REQ}(A) \) be the set of temporal
formule in \( A \), that is, \( \text{REQ}(A) = A \cap \text{TF}(\varphi) \). Given a model \( M = \langle \langle T, I(T) \rangle, V \rangle \) for \( \varphi \), we define a
function \( V_A : I(T) \to A_{\varphi} \) that associates an atom \( A \) with every \( [d_1, d_3] \in I(T) \) in such a way that
for every \( \psi \in \text{CL}(\varphi) \), \( \psi \in A_{\varphi} \) if, and only if, \( M, [d_1, d_3] \models \psi \).

The following theorem proves that atoms associated with intervals whose right endpoints coincide
feature the same temporal formulæ.

**Proposition 3.2.1.** Let \( M = \langle \langle T, I(T) \rangle, V \rangle \) be a model for \( \varphi \) and \( d_1, d_3, d_4 \in T \) be such that both
\( d_1 < d_3 \) and \( d_3 < d_4 \). If \( V_A([d_1, d_4]) = A \) and \( V_A([d_1, d_3]) = A' \), then \( \text{REQ}(A) = \text{REQ}(A') \).

**Proof.** Suppose, by contradiction, that \( V_A([d_1, d_3]) = A \), \( V_A([d_1, d_4]) = A' \), and \( \text{REQ}(A) \neq \text{REQ}(A') \).
Without loss of generality, we can assume that there exists \( \langle A \rangle \psi \) such that \( \langle A \rangle \psi \in \text{REQ}(A) \) and \( \langle A \rangle \psi \notin \text{REQ}(A') \). By definition of atom, it follows that \( [A] \neg \psi \in \text{REQ}(A') \). Since \( M \) is a model for \( \varphi \), there exists \( d_h > d_4 \) such that \( M, [d_4, d_h] \models \psi \) (for \( \langle A \rangle \psi \in \text{REQ}(A) \)) and for every \( d_k > d_e M, [d_e, d_k] \models \neg \psi \) (for \( [A] \neg \psi \in \text{REQ}(A') \)). For \( k = h \), we have \( M, [d_e, d_h] \models \neg \psi \) (contradiction).

Atoms are connected by the following binary relation.

**Definition 3.2.2.** Let \( R_\varphi \) be a binary relation over \( A_{\varphi} \) such that, for every pair of atoms \( A, A' \in A_{\varphi} \), \( A R_\varphi A' \) if, and only if, for every \( [A] \psi \in \text{CL}(\varphi) \), if \( [A] \psi \in A \), then \( \psi \in A' \).
3.2.1 The tableau system

A tableau for an RPNL formula $\varphi$ is a suitable decorated tree $T$. A finite prefix of the natural numbers $\mathbb{D}_B = (\mathbb{D}_B, <)$ is associated with every branch $B$ of $T$. The decoration of a node $n$ in $T$, denoted by $\nu(n)$, is a pair $\langle [d_l, d_i], \mathcal{A} \rangle$, where $\mathcal{A}$ is an atom and $d_l, d_i \in \mathbb{D}_B$ for every branch $B$ containing $n$. Given a node $n$, we denote by $\mathcal{A}(n)$ 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 $T$ with decoration $\langle [d_i, d_i], \mathcal{A}_n \rangle$. Since $n$ is a leaf, there is a unique branch $B$ containing $n$ in $T$. 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 $d \in \mathbb{D}_B$, with $d_0 < d < d_i$, be such that there are no ancestors $n'$ of $n$ with decoration $\langle [d, d_i], \mathcal{A}' \rangle$, for a suitable $\mathcal{A}'$. If there exists an atom $\mathcal{A}''$ such that $\text{REQ}(\mathcal{A}'') = \text{REQ}(\mathcal{A}_n)$ and for all ancestors $\pi$ of $n$ with decoration $\langle [\bar{d}, d], \bar{\mathcal{A}} \rangle$, for suitable $\bar{d}$ and $\bar{\mathcal{A}}$, $\bar{\mathcal{A}} \not\subseteq \mathcal{A}''$, we add an immediate successor $n''$ to $n$ with decoration $\langle [d, d_i], \mathcal{A}'' \rangle$.

2. Step rule. Let $\langle \{\varphi_1, \ldots, \varphi_k\} \rangle$, with $k \geq 1$, be the set of $\langle \mathcal{A} \rangle$-formulae in $\mathcal{A}_n$. If there exist $k$ atoms $\mathcal{A}_{n_1}^{h_1}, \ldots, \mathcal{A}_{n_k}^{h_k}$ such that, for $1 \leq h \leq k$, $\mathcal{A}_{n_h} \not\subseteq \mathcal{A}_n$ and $\varphi_h \in \mathcal{A}_{n_h}$, we add $k$ immediate successors $n'_h$ with decoration $\langle [d_i, d_{i+1}], \mathcal{A}_{n_h}' \rangle$, to $n$. Let $B_1, \ldots, B_k$ 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 $d_{i+1}$ greater than all points in $\mathbb{D}_B$.

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 $d_{i+1}$, 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 $d_i$ to agree on their temporal requests.

Definition 3.2.3. For all branches $B$ in $T$ and for all $d > d_0$ in $\mathbb{D}_B$, we define the set of temporal formulae associated with $d$, denoted $\text{REQ}(d)$, as the set $\text{REQ}(\mathcal{A}(n))$ for all nodes $n$ in $B$ with decoration $\langle [\bar{d}, d], \bar{\mathcal{A}} \rangle$.

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 $\langle [d_i, d_i], \mathcal{A}_n \rangle$ belonging to a branch $B$ is blocked if there exists an ancestor $n'$ of $n$ with decoration $\langle [d_k, d_i], \mathcal{A}_{n'} \rangle$, with $d_k < d_i$ in $\mathbb{D}_B$, such that:

$$\text{REQ}(\mathcal{A}_n) = \text{REQ}(\mathcal{A}_{n'})$$

and

$$\forall d_h (d_h < d_i \rightarrow \exists d_g (d_g < d_i \land \text{REQ}(d_h) = \text{REQ}(d_g))).$$

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 $T$ and a branch $B$ of $T$ 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 $\langle [d_i, d_i], \mathcal{A}_n \rangle$, 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 $d_k < d_i$, such that there are no ancestors of $n$ with decoration $\langle [d_k, d_i], \mathcal{A}' \rangle$, for a suitable $\mathcal{A}'$, 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.
A tableau system for RPNL over trees

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

Pruning the tableau. Given a tableau \( T \) for \( \varphi \), 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 \( (A)\varphi \in A_n \) such that there is no successor \( n' \) of \( n \) with \( \varphi \in A(n') \);
4. remove every node which is not reachable from the root.

We shall prove that an RPNL formula \( \varphi \) is satisfiable if, and only if, there exists a non-empty pruned tableau for it.

3.2.2 An example

We conclude the section by applying the proposed tableau method to the formula \( \varphi_2 = (A)(\langle A \rangle \top \wedge [A][\langle A \rangle \top] \wedge \langle A \rangle[A] \perp) \). A non-empty pruned tableau for \( \varphi_2 \) is depicted in Figure 3.3. 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: \( A_0 = \{\top, \langle A \rangle \top, \langle A \rangle[A] \perp, \langle A \rangle \psi, \neg \psi, \varphi_2\} \), \( A_1 = \{\top, \langle A \rangle \top, [A][\langle A \rangle \top], \langle A \rangle \psi, \neg \psi, \neg \varphi_2\} \), and \( A_2 = \{\top, [A] \perp, [A][\langle A \rangle \top], [A] \neg \psi, \neg \psi, \neg \varphi_2\} \), where \( \psi \) is a shorthand for \( \langle A \rangle \top \wedge [A][\langle A \rangle \top] \). The relation \( R_\varphi \) 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. \( \langle A \rangle[A] \perp \) is dealt with by node \( n_3 \). Since \( A_2 \) does not contain

\[
\begin{array}{c}
\text{n}_1 \\
(\langle d_1, d_2, A_0 \rangle, D_2) \\
\text{blocked}
\end{array}
\begin{array}{c}
\text{n}_2 \\
(\langle d_1, d_2, A_1 \rangle, D_2) \\
\text{block}
\end{array}
\begin{array}{c}
\text{n}_3 \\
(\langle d_1, d_2, A_2 \rangle, D_2)
\end{array}
\begin{array}{c}
\text{n}_4 \\
(\langle d_2, d_3, A_0 \rangle, D_3)
\end{array}
\begin{array}{c}
\text{n}_5 \\
(\langle d_2, d_3, A_1 \rangle, D_3) \\
\text{block}
\end{array}
\begin{array}{c}
\text{n}_6 \\
(\langle d_1, d_3, A_0 \rangle, D_3)
\end{array}
\begin{array}{c}
\text{n}_7 \\
(\langle d_3, d_4, A_0 \rangle, D_4)
\end{array}
\begin{array}{c}
\text{n}_8 \\
(\langle d_3, d_4, A_1 \rangle, D_4)
\end{array}
\begin{array}{c}
\text{n}_9 \\
(\langle d_3, d_4, A_2 \rangle, D_4)
\end{array}
\text{closed}
\]
\(\langle A\rangle\)-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 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 \(\langle A\rangle \bot\), there exists a formula in \(A_0\), namely, \(\langle A\rangle \langle A\rangle \bot\), such that there are no successors \(n'\) of the node \(n_6\) with \(\langle A\rangle \bot \in A(n')\). 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.

### 3.2.3 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 \(\varphi\) from a non-empty pruned tableau \(\mathcal{T}\) for it. Conversely, completeness is proved by showing that, for any satisfiable formula \(\varphi\), there exists a non-empty pruned tableau for it.

**Theorem 3.2.4 (Soundness).** Given a formula \(\varphi\) and a pruned tableau \(\mathcal{T}\) for it, if \(\mathcal{T}\) is non-empty, then \(\varphi\) is satisfiable.

**Proof.** Let \(\mathcal{T}\) be a non-empty pruned tableau for \(\varphi\). We show that we can build a model for \(\varphi\) 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 \(D_B = \{d_0 < d_1\}\). We start the construction with a partial model, which consists of a two-node tree \(\mathcal{T}_0 = \{d_0, d_1\}\) and a valuation \(V[d_0, d_1] = \{p : p \in A_0\}\). Then, we progressively turn it into a model for \(\varphi\) by a depth-first visit of \(\mathcal{T}\). Let \(M\) be the current partial model, \(n\) be the current node of \(\mathcal{T}\), with decoration \(\langle [d_i, d_k], A_n\rangle\), and \(D_B\) 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 \(n_\psi\) of \(n\) such that \(\psi \in A(n_\psi)\). We expand the partial model by adding an immediate successor \(d_\psi\) to \(d_j\) and by putting \(V[d_j, d_\psi] = \{p : p \in A(n_\psi)\}\).

- **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 \(V[d_k, d_j] = \{p : p \in A'(n)\}\).

- **The node \(n\) is blocked.** In such a case, there exists an ancestor \(n'\) of \(n\), with decoration \(\langle [d_k, d_j], A_{n'}\rangle\), such that \(\text{REQ}(A_n) = \text{REQ}(A_{n'})\) and for all \(d_k < d_i \in D_B\) there exists \(d_m < d_i\) with \(\text{REQ}(d_k) = \text{REQ}(d_m)\). 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 \(\varphi\).

**Theorem 3.2.5 (Completeness).** Given a satisfiable formula \(\varphi\), there exists a non-empty pruned tableau \(\mathcal{T}\) for \(\varphi\).
3.2. A tableau system for RPNL over trees

Proof. Let \( \varphi \) be a satisfiable formula and let \( M = \langle \langle \mathbb{T}, \mathbb{I}(\mathbb{T}) \rangle, \mathbb{V} \rangle \) be a model for it. We prove that there exists a non-empty pruned tableau \( \mathcal{T} \) corresponding to \( M \). Since \( M \) is a model for \( \varphi \), there exists an interval \([d_0, d_1]\) such that \( M, [d_0, d_1] \models \varphi \). Let \( A_0 = \{ \psi \in \text{CL}(\varphi) : M, [d_0, d_1] \models \psi \} \). We start the construction of \( \mathcal{T} \) with a partial tableau which includes a single node with decoration \((d_0, d_1], A_0)\) and \( \mathbb{D}_\mathcal{B} = \{ d_0 < d_1 \} \). Then, we proceed by induction on expansion rules. Let \( \mathcal{T} \) be the current partial tableau, \( n \) be a leaf node with decoration \((d_1, d_j], A_n)\), and \( \mathbb{D}_\mathcal{B} \) 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 \([d_k, d_1]\). We add a successor \( n' \) to \( n \) with decoration \((d_k, d_1], A')\), where \( A' = \{ \psi \in \text{CL}(\varphi) : M, [d_k, d_1] \models \psi \} \).

- **The step rule is applicable to \( n \).** For every \( \langle A \rangle \psi \in A_n \), there exists an interval \([d_1, d_q]\) such that \( M, [d_1, d_q] \models \psi \). Let \( A_q = \{ \theta \in \text{CL}(\varphi) : M, [d_1, d_q] \models \theta \} \). For every \( \langle A \rangle \psi \in A_n \), we add a successor \( n_q \) to \( n \) with decoration \((d_1, d_q], A_q)\) and, for every new branch \( B' \), we define \( \mathbb{D}_\mathcal{B}' = \mathbb{D}_\mathcal{B} \cup \{ d_q \} \).

- **No rule is applicable to \( n \).** Since \( M \) is a model for \( \varphi \), \( n \) 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 \( \varphi \) to which no removal step can be applied. \( \square \)

3.2.4 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 \( \varphi \).

Let \( n = |\varphi| \). We have that \( \text{REQ}(d) \) can take \( 2^{\text{TF}(\varphi)} \), where \( |\text{TF}(\varphi)| \leq 2 \cdot (n-1) \), different values and there can be at most \( 2^{\text{TF}(\varphi)} \) 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 \( m = |D_1| \) and \( D_1 \) 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 \( (A) \)-formulae in \( \text{CL}(\varphi) \), the size of a tableau is thus \( O(2^{2n}) \). The following theorem immediately follows.

**Theorem 3.2.6.** 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. \( \square \)

To prove EXPSPACE-hardness of the decision problem for RPNL interpreted over trees, we exploit Alternating Turing Machines (ATM for short) [24]. An ATM is a tuple \( M = (Q, \Gamma, \delta, q_0, g) \), where \( (Q, \Gamma, \delta, q_0) \) is a one-tape non-deterministic Turing Machine and \( g \) is a function \( g : Q \rightarrow \{ \text{accept}, \text{reject} \} \) that classifies the states of \( M \). Given an input word \( w \) and the computation tree of the one-tape non-deterministic Turing Machine \( (Q, \Gamma, \delta, q_0) \) on \( w \), we say that a configuration \( C = (q, v, i) \), that is, a node of the computation tree, is accepting if either \( g(q) = \text{accept} \), or \( g(q) = \text{reject} \) and at least one successor of \( C \) is accepting, or \( g(q) = \text{accept} \) 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 \( \text{AEXPTIME} \), that is, the class of the problems that can be decided in exponential time by an ATM, corresponds to the complexity class EXPSPACE [24].

**Theorem 3.2.7.** The decision problem for RPNL over trees is EXPSPACE-hard.
The future fragment of PNL interpreted over trees

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 \(n \in \mathbb{O}[|w|]\). We build a formula \(\varphi\) 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 \(2^n - 1\) means of \(n\) propositional letters \(C_1, \ldots, C_n\). Moreover, we encode every position \(p\) in the tape by means of \(n\) propositional letters \(P_1, \ldots, P_n\). 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 propositional letters \(Q_1, \ldots, Q_m\), with \(m = |Q|\), for the states in the alphabet \(\Gamma\), \(m\) propositional letters \(A_1, \ldots, A_h\), with \(h = |\Gamma|\), for the symbols in the alphabet \(\Gamma\), \(m\) propositional letters \(C_1, \ldots, C_n\), with \(m = |\Gamma|\), 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 \([20]\), 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 \([d_1, d_1]\) if, and only if, \(R\) holds over \([d_1, d_k]\) for every \(d_k > d_1\). 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 \(\big((A)R \rightarrow [A]\big) \land \big([A]\big) \rightarrow [A]R\). Let \(\psi_{\text{loc}}\) 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 \(\psi^p_1 \equiv \bigwedge_{i=1}^n C_i \iff [A]C_i\) and \(\psi^c_1 \equiv \bigwedge_{i=1}^n P_i \iff [A]P_i\) such that, for any interval \([d_1, d_1]\), \(\psi^c_1\) (resp., \(\psi^p_1\)) holds over \([d_1, d_1]\) if, and only if, \(c(d_1) = c(d_1)\) (resp., \(p(d_1) = p(d_1)\)). Next, we introduce the auxiliary formulae \(\psi^p_1\) and \(\psi^c_1\) such that, for any interval \([d_1, d_1]\), \(\psi^p_1\) (resp., \(\psi^c_1\)) holds over \([d_1, d_1]\) if, and only if, \(c(d_1) = c(d_1) + 1\) (resp., \(c(d_1) = c(d_1) - 1\)). Such formulae are defined as follows:

\[
\begin{align*}
\psi^p_1(n) &= -P_n \land [A]P_n \\
\psi^p_1(k) &= (\neg P_k \land [A]P_k \land \bigwedge_{i=k+1}^n [A]P_i) \lor ((P_k \land [A]P_k) \land \psi^p_1(k+1)) \\
\psi^c_1(n) &= P_n \land [A]P_n \\
\psi^c_1(k) &= (P_k \land [A]P_k \land \bigwedge_{i=k+1}^n [A]P_i) \lor ((\neg P_k \land [A]P_k) \land \psi^c_1(k+1)) \\
\psi^p_{-1} &= \psi^p_1(1) \\
\psi^c_{-1} &= \psi^c_1(1)
\end{align*}
\]

Analogously, we introduce a formula \(\psi^c_1\) that holds over an interval \([d_1, d_1]\) if, and only if, \(c(d_1) = c(d_1) + 1\), which can be obtained by substituting \(C\) for \(P\) everywhere in the previous formulae (notice that a formula \(\psi^c_1\) 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 \(\psi \land [A]\psi \land [A][A]\psi\). First, we impose that every element of the model \(d_1\) where at least one among \(C_1, \ldots, C_n, P_1, \ldots, P_n\) evaluates to false has a successor \(d_1\). Moreover, if every \(P_t\) evaluates to true in \(d_1\), that is, if \(d_1\) represents the last tape cell, then every \(P_t\) evaluates to false in \(d_1\) and \(c(d_1) = c(d_1) + 1\); otherwise, \(c(d_1) = c(d_1)\) and \(p(d_1) = p(d_1) + 1\). Such a condition is imposed by the formula \([U]\psi_{\text{suc}}\), where \(\psi_{\text{suc}}\) is defined as follows:

\[
\begin{align*}
\psi_{\text{suc}} &= (\psi_{\text{next}} \land \neg \bigwedge_{i=1}^n (P_t \land [A]C_i)) \rightarrow (A)\psi_{\text{next}} \\
\psi_{\text{next}} &= (\bigwedge_{i=1}^n (P_t \land [A]P_t) \land \psi^c_1) \lor (\psi^c_1 \land \psi^p_1)
\end{align*}
\]

Next, we impose that every tape cell contains only one symbol of \(\Gamma\) and that, in a given configuration, the head is associated only with one tape position by means of the formulae \([U]\psi_{\text{a}}\), where
\[\psi_\alpha = \bigvee_{i=1}^{h} A_i \land \bigwedge_{i=1}^{h} (A_i \rightarrow \bigwedge_{j \neq i} \neg A_j)\]

Furthermore, we associate the state of the machine \( M \) with the head position by means of the formula \( [U] \psi_{\text{head}} \), where \( \psi_{\text{head}} = (H \land \psi_C) \rightarrow [A] \rightarrow H \). Furthermore, we associate the state of the machine \( M \) with the head position by means of the formula \( [U] \psi_{\text{state}} \), where \( \psi_{\text{state}} = (H \leftrightarrow \bigvee_{i=1}^{n} Q_i) \land \bigwedge_{i=1}^{n} (Q_i \rightarrow \bigwedge_{j \neq i} \neg Q_j) \). 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 \( [U] \psi_{\text{pos}} \), where \( \psi_{\text{pos}} = (\psi_{\text{pre}} \land \psi_{\text{post}} \land \neg H) \rightarrow \bigwedge_{i=1}^{n} (A_i \rightarrow [A] \land A_i) \). By definition of \( \text{ATM} \), 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 \( \delta(Q, A) = (Q', A', \sim), (Q'', A'', \sim) \), where \( \sim, \sim \in \{\rightarrow, \leftarrow\} \), be a transition of \( M \). Suppose that \( \sim_1 = \rightarrow \) and \( \sim_2 = \leftarrow \) (the other cases are similar). Moreover, let:

\[
\psi^{\delta}_1 = (Q \land A \land H \land \psi_{\text{pre}} \land \psi_{\text{post}}) \rightarrow (A)(A' \land \psi_{\text{next}} \land (A)(Q' \land H)) \]

\[
\psi^{\delta}_2 = (Q \land A \land H \land \psi_{\text{pre}} \land \psi_{\text{post}}) \rightarrow (A)(Q'' \land H \land \psi_{\text{next}} \land (A)(A''))
\]

If \( Q \) is an \( \lor \)-state, we encode the transition \( \delta(Q, A) \) with the formula \( \psi^{\delta}_Q = [U](\psi^{\delta}_1 \lor \psi^{\delta}_2) \), while if \( Q \) is an \( \land \)-state, we encode the transition with the formula \( \psi^{\delta}_Q = [U](\psi^{\delta}_1 \land \psi^{\delta}_2) \). Let \( w \) be 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 \( \psi_{\text{init}} \) defined as follows:

\[
\psi_{\text{init}} = \psi_{\text{init}}(1) \land H \land Q_0 \land \bigwedge_{i=1}^{n} (\neg C_i \land \neg P_i);
\]

\[
\psi_{\text{init}}(|w|) = \psi_{\text{next}} \land A_{w(|w|)} \land [A]\land \bigwedge_{i=1}^{n} \neg C_i \rightarrow A_{\text{blank}} \land [A]\land \bigwedge_{i=1}^{n} \neg C_i \rightarrow A_{\text{blank}};
\]

\[
\psi_{\text{init}}(k) = \psi_{\text{next}} \land A_{w(k)} \land (A)\psi_{\text{init}}(k+1),
\]

with \( k < |w| \),

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

In this chapter, 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.
3. The future fragment of PNL interpreted over trees
WSpPNL: a decidable spatial extension of PNL

The management of qualitative spatial information is an important research area in computer science and AI. Modal logic provides a natural framework for the formalization and implementation of qualitative spatial reasoning. Unfortunately, when directional relations are considered, modal logic systems for spatial reasoning usually turn out to be undecidable (often even not recursively enumerable). In this chapter, we give a first example of a decidable modal logic for spatial reasoning with directional relations, called Weak Spatial Propositional Neighborhood Logic (WSpPNL for short). WSpPNL features two modalities, respectively an east modality and a north modality, to deal with non-empty rectangles over \( \mathbb{N} \times \mathbb{N} \). We first show the NEXPTIME-completeness of WSpPNL, then we develop an optimal tableau method for it.

4.1 A brief introduction on spatial logics

The main goal of qualitative spatial representation and reasoning techniques is to capture commonsense knowledge about space and to provide a calculus of spatial information without referring to a quantitative model. Even though quantitative models provide a more accurate description of spatial domains, qualitative models are often the best or the only choice. In many cases, indeed, there is a lack of quantitative models or existing ones turn out to be intractable. In addition, qualitative models make it possible to cope with spatial data indeterminacy and to reason about incomplete spatial knowledge. The problem of representing and reasoning about qualitative spatial information can be viewed under three different points of view: (i) the algebraic perspective, that is, purely existential theories formulated as constraint satisfaction systems over jointly exhaustive and mutually disjoint sets of topological, directional, or combined relations; (ii) the first-order perspective, that is, first-order theories of topological, directional, or combined relations; (iii) the modal logic perspective, where a propositional modal language is interpreted over (a representation of space via) suitable Kripke structures. The increase in expressiveness of the two latter approaches is paired by an increase in computational complexity, which often makes them impracticable. Depending on the considered class of spatial relations, we can further distinguish between topological and directional spatial reasoning. While topological relations between pairs of spatial objects (viewed as sets of points) are preserved under translation, scaling, and rotation, directional relations depend on the relative spatial position of the objects.

A comprehensive and sufficiently up-to-date survey, which covers topological, directional, and combined constraint systems and relations, can be found in [27]. Deductive systems for reasoning about topological relations have been proposed in various papers, including Bennett’s work [6, 7], later extended by Bennett et al. [8], Nutt’s systems for generalized topological relations [51], the modal logic systems for a number of mathematical theories of space described in [2], the logic of connectedness constraints developed by Kontchakov et al. [38], and Lutz and Wolter’s modal logic of topological relations [40]. Directional relations have been mainly dealt with following both the algebraic approach and the modal logic one. As for the first one, the most important contributions
are those by G{"u}sgen [35] and by Mukerjee and Joe [50], that introduce Rectangle Algebra (RA),
later extended by Balbiani et al. in [4, 5]. As for the second one, we mention Venema’s Compass Logic [60],
whose undecidability has been shown by Marx and Reynolds in [42], and Spatial Propositional Neighborhood Logic (SpPNL for short) by Morales et al. [48], that generalizes the logic of
temporal neighborhood [33] to the two-dimensional space. SpPNL makes it possible to reason
about regions, approximated by their minimum bounding boxes, by taking advantage of four modal
operators that allow one to move along the x- and the y-axis. In [48], the authors analyze the
expressive power of the logic, provide a representation theorem, and devise a (non-terminating) sound
and complete tableau system.

In this chapter, we focus our attention on a proper syntactical and semantical fragment of SpPNL,
called Weak SpPNL (WSpPNL for short). SpPNL has been proved to be undecidable over most
relevant class of frames [48]. To recover decidability, we restrict ourselves to the class of frames iso-
morphic to $\mathcal{D} \times \mathcal{D}$, where $\mathcal{D}$ is either $\mathbb{N}$ or a prefix of it, and consider a syntactic fragment of SpPNL
with two modalities only, namely $\langle E \rangle$ (east) and $\langle N \rangle$ (north), with a weakened semantics. We show
that WSpPNL is NEXPTIME-complete, and we provide it with a sound and complete tableau
system. Both the decidability proof and the tableau system can be viewed as non-trivial adapta-
tions of those for Propositional Neighborhood Logic (PNL) [16]. We also show that WSpPNL is
expressive enough to support a (weak form of) universal operator and nominals. At the best of
our knowledge, WSpPNL is the first example of a decidable modal logic for directional reasoning
that deals with extended regions.

4.2 SpPNL and WSpPNL

The language of Spatial Propositional Neighborhood Logic (SpPNL) consists of a set of proposi-
tional variables $\mathcal{A}$, the logical connectives $\neg$ and $\lor$, and the modalities $\langle E \rangle$, $\langle W \rangle$, $\langle N \rangle$, and $\langle S \rangle$.
The other logical connectives, as well as the logical constants $\top$ and $\bot$, can be defined in the usual
way. Let $p \in \mathcal{A}$. SpPNL formulas, denoted by $\varphi, \psi, \ldots$, are recursively defined as follows:

$$
\varphi ::= p \mid \neg \varphi \mid \varphi \lor \psi \mid \langle E \rangle \varphi \mid \langle W \rangle \varphi \mid \langle N \rangle \varphi \mid \langle S \rangle \varphi.
$$

Let $\mathcal{D}_h = \langle D_h, < \rangle$ and $\mathcal{D}_v = \langle D_v, < \rangle$, where $D_h$ (resp., $D_v$) is (a prefix of) the set of natural
numbers $\mathbb{N}$ and $<$ is the usual linear order. Elements of $\mathcal{D}_h$ (resp., $\mathcal{D}_v$) will be denoted by $h_a, h_b, \ldots$
(resp., $v_a, v_b, \ldots$). A spatial frame is a structure $\mathcal{F} = (\mathcal{D}_h, \times \mathcal{D}_v)$. The set of objects (rectangles) is
the set $\mathcal{O}(\mathcal{F}) = \{ (h_a, v_b), (h_c, v_d) \} | h_a < h_c, v_b < v_d, h_a, h_c \in D_h, v_b, v_d \in D_v \}$. The semantics
of SpPNL over $\mathcal{O}(\mathcal{F})$ is given in terms of spatial models $M = (\mathcal{F}, \mathcal{O}(\mathcal{F}), \mathcal{V})$, where $\mathcal{F}$ is a spatial
frame, $\mathcal{O}(\mathcal{F})$ is the set of relevant objects, and $\mathcal{V} : \mathcal{O}(\mathcal{F}) \rightarrow 2^\mathcal{A}$ is a spatial valuation function. The pair
$(\mathcal{F}, \mathcal{O}(\mathcal{F}))$ is called spatial structure. Given a model $M$ and an object $\langle (h_a, v_b), (h_c, v_d) \rangle$, the
truth relation for SpPNL formulas is defined as follows:

$$
M, \langle (h_a, v_b), (h_c, v_d) \rangle \models p \iff p \in \mathcal{V}(\langle (h_a, v_b), (h_c, v_d) \rangle), \text{ for any } p \in \mathcal{A};
$$

$$
M, \langle (h_a, v_b), (h_c, v_d) \rangle \models \neg \varphi \iff M, \langle (h_a, v_b), (h_c, v_d) \rangle \nmodels \varphi;
$$

Figure 4.1: The semantics of $\langle E \rangle$ and $\langle N \rangle$ in SpPNL (left) and WSpPNL (right).
\[ M, \langle (h_a, v_b), (h_c, v_d) \rangle \models \phi \lor \psi \iff M, \langle (h_a, v_b), (h_c, v_d) \rangle \models \phi \lor M, \langle (h_a, v_b), (h_c, v_d) \rangle \models \psi; \]
\[ M, \langle (h_a, v_b), (h_c, v_d) \rangle \models \langle E \rangle \psi \iff \text{there exists } h_e \in D_h \text{ such that } h_e < h_c, \text{ and } M, \langle (h_e, v_b), (h_c, v_d) \rangle \models \psi; \]
\[ M, \langle (h_a, v_b), (h_c, v_d) \rangle \models \langle W \rangle \psi \iff \text{there exists } h_e \in D_h \text{ such that } h_e < h_a, \text{ and } M, \langle (h_a, v_b), (h_e, v_d) \rangle \models \psi; \]
\[ M, \langle (h_a, v_b), (h_c, v_d) \rangle \models \langle N \rangle \psi \iff \text{there exists } v_e \in D_v \text{ such that } v_d < v_e, \text{ and } M, \langle (h_a, v_d), (h_c, v_e) \rangle \models \psi; \]
\[ M, \langle (h_a, v_b), (h_c, v_d) \rangle \models \langle S \rangle \psi \iff \text{there exists } v_e \in D_v \text{ such that } v_e < v_b, \text{ and } M, \langle (h_a, v_e), (h_c, v_b) \rangle \models \psi. \]

As an example, the semantics of \( E \) (resp., \( N \)) is graphically depicted in Figure 4.1 (left): if \( \langle (h_a, v_b), (h_c, v_d) \rangle \) satisfies \( E \) p (resp., \( N \) p), then there exists a rectangle whose left (resp., bottom) edge coincides with the right (resp., top) edge of \( \langle (h_a, v_b), (h_c, v_d) \rangle \) that satisfies p.

Both the strength (expressiveness) and the weakness (undecidability) of the logic SpPNL originate from the fact that its operators allow one to move (in one step) from one rectangle to a right (resp., left, top, bottom) adjacent one. As an example, when we apply the operator \( E \) to move to the right of the current rectangle, three out of four coordinates of the resulting rectangle, namely, \( h_c, v_b, v_d \), are determined by (coincide with) those of the current one. The computational behavior of the logic can be improved by relaxing such a constraint. Let us define the east (resp., west, north, south) of a rectangle as the entire area to the right of it (resp., to the left of it, over it, under it) and redefine the semantics of the modal operators accordingly. The revised semantics of \( E \) (resp., \( N \)) is graphically depicted in Figure 4.1 (right). According to it, only one out of four coordinates of the resulting rectangle, namely, \( h_c \) (resp., \( v_d \)), is determined by (coincide with) those of the current one.

Weak SpPNL (WSPPNL for short) features the east \( E \) and north \( N \) modalities only, endowed with the above-described weakened semantics. The language of WSPPNL consists of a set of propositional variables \( AP \), the logical connectives \( \land, \lor, \land, \lor \), and the modalities \( E \) and \( N \). The other logical connectives and the logical constants \( \top \) and \( \bot \) are defined in the usual way. WSPPNL formulas are recursively defined as follows:

\[ \varphi ::= p \mid \neg \varphi \mid \varphi \lor \psi \mid \langle E \rangle \varphi \mid \langle N \rangle \varphi. \]

Given a model \( M \) and an object \( \langle (h_a, v_b), (h_c, v_d) \rangle \), the clauses for the two modal operators are revised as follows:

\[ M, \langle (h_a, v_b), (h_c, v_d) \rangle \models \langle E \rangle \psi \iff \text{there exist } h_e \in D_h \text{ and } v_f, v_g \in D_v \text{ such that } h_e < h_c, v_f < v_g, \text{ and } M, \langle (h_e, v_f), (h_c, v_g) \rangle \models \psi; \]
\[ M, \langle (h_a, v_b), (h_c, v_d) \rangle \models \langle N \rangle \psi \iff \text{there exist } v_e \in D_v \text{ and } h_f, h_g \in D_h \text{ such that } v_d < v_e, h_f < h_g, \text{ and } M, \langle (h_r, v_d), (h_g, v_e) \rangle \models \psi. \]

Let \( [E] \) and \( [N] \) be the duals of \( E \) and \( N \), respectively. We say that \( \varphi \) is a horizontal formula if \( \varphi = \langle E \rangle \varphi \) or \( \varphi = [\neg E] \varphi \) for some \( \psi \) (notice that \( \neg \langle E \rangle \varphi \) is equivalent to \( \langle E \rangle \neg \varphi \) and \( \neg [\neg E] \varphi \) is equivalent to \( \langle E \rangle \neg \varphi \); similarly, we say that \( \varphi \) is a vertical formula if \( \varphi = \langle N \rangle \varphi \) or \( \varphi = \langle N \rangle \varphi \) for some \( \psi \). Spatial formulas are horizontal and vertical formulas.

Since WSPPNL features only north/east operators, we can restrict our attention to the initial object \( \langle (0, 0), (1, 1) \rangle \), as stated by the following proposition.

**Proposition 4.2.1.** Let \( \varphi \) be a WSPPNL-formula. Then, \( \varphi \) is satisfiable if and only if the WSPPNL-formula \( \varphi' = \varphi \lor \langle E \rangle \varphi \lor \langle E \rangle \varphi \lor \langle N \rangle \varphi \lor \langle N \rangle \varphi \) is satisfiable over the initial object.
Proof. Let $M$ be any spatial model and $(\langle h_a, v_b \rangle, \langle h_c, v_d \rangle)$ be any object on it. By contradiction, suppose that, for a given WSpPNL-formula $\psi$, it is the case that $M, (\langle h_a, v_b \rangle, \langle h_c, v_d \rangle) \models [WU]\psi$ and there exists an object $(\langle h_a', v_b', h_c', v_d' \rangle)$ such that $h_a' \neq 0$ or $v_b' \neq 0$ and $M, (\langle h_a', v_b', h_c', v_d' \rangle) \not\models \psi$. Suppose $h_a' \neq 0$ (the other case can be treated in a similar way). Since $M, (\langle h_a, v_b \rangle, \langle h_c, v_d \rangle) \models [WU]\psi$, then, in particular, $M, (\langle h_a, v_b \rangle, (h_c, v_d)) \models [N][E]\psi$. This means that for every $v_c > v_d$ and every $h_f, h_g, M, (\langle h_f, v_d \rangle, (h_g, v_c)) \models [E]\psi$. Thus, we have that, for every $v_c > v_d$, $M, (\langle 0, v_d \rangle, (h_c, v_c)) \models [E]\psi$. This implies that for every $h_t > h_c'$ and every $v_c', v_m$, $M, (\langle h_t', v_c', (h_t, v_m) \rangle) \models \psi$. In particular, we have that $M, (\langle h_t', v_c', (h_t, v_m) \rangle) \models \psi$, which is a contradiction.

The opposite direction can be proved in a similar way. It is worth pointing out that the assumption that $h_a \neq 0$ or $v_b \neq 0$ plays an essential role in the implication from right to left, while it plays no role in the direction from left to right. \hfill \square

Thanks to Proposition 4.2.1, satisfiability of a WSpPNL-formula $\varphi$ thus reduces to the existence of a model $M$ such that $M, (\langle 0, 0 \rangle, (1, 1)) \models \varphi'$.

### 4.3 WSpPNL Expressiveness

In this section we show that, despite its simplicity, WSpPNL is expressive enough to capture various interesting spatial notions. First of all, it makes it possible to define a sort of pseudo-universal modal operator. As implicitly stated by Proposition 4.2.1, the lack of the south/west operators prevents WSpPNL from accessing objects whose left bottom corner is equal to $(0, 0)$. However, WSpPNL can access every other object of the frame.

**Definition 4.3.1.** Given a WSpPNL-formula $\psi$, we say that $\psi$ is true almost everywhere in a model $M$ if and only if for every object $(\langle h_a, v_b \rangle, (h_c, v_d))$ such that $h_a \neq 0$ or $v_b \neq 0$, $M, (\langle h_a, v_b \rangle, (h_c, v_d)) \models \psi$.

Let $[WU]$ (weakly universal) be the following derived operator of WSpPNL:

$$[WU] \psi := \psi \land [N][E]\psi \land [E][N]\psi.$$ 

The next proposition shows that the operator $[WU]$ captures the notion introduced by Definition 4.3.1.

**Proposition 4.3.2.** Let $M$ be a spatial model and $(\langle h_a, v_b \rangle, (h_c, v_d))$ be one of its objects, with $h_a \neq 0$ or $v_b \neq 0$. We have that $M, (\langle h_a, v_b \rangle, (h_c, v_d)) \models [WU]\psi$ if and only if $\psi$ is true almost everywhere in $M$.

Moreover, for any propositional letter $p \in AP$, WSpPNL allows one to express a weak nominal $\text{wn}(p)$.

**Definition 4.3.3.** Given a propositional variable $p \in AP$, we say that $p$ is true almost only on $(\langle h_a, v_b \rangle, (h_c, v_d))$, with $h_a \neq 0$ or $v_b \neq 0$, in a model $M$ if and only if $M, (\langle h_a, v_b \rangle, (h_c, v_d)) \not\models p$ and, for every object $(\langle h_a', v_b' \rangle, (h_c', v_d')) \neq (\langle h_a, v_b \rangle, (h_c, v_d))$, with $h_a' \neq 0$ or $v_b' \neq 0$, $M, (\langle h_a', v_b' \rangle, (h_c', v_d')) \not\models \neg p$.

Given a propositional variable $p$, the operator $\text{wn}(p)$ ($p$ is a weak nominal) can be expressed in WSpPNL by taking advantage of two special propositional variables $\bar{p}_r$ and $\bar{p}_v$ as follows:

$$\text{wn}(p) := \text{wn}_1(p) \land \text{wn}_2(p) \land \text{wn}_3(p),$$

where

$$\text{wn}_1(p) := p \land (E)p_r \land (N)p_v,$$

$$\text{wn}_2(p) := [WU][(E)p \rightarrow \neg (E)p] \land (N)p \rightarrow \neg (N)(N)p],$$

$$\text{wn}_3(p) := [WU](p \rightarrow \neg (E)p_r \land \neg (N)(N)p_v).$$
4.4. WSpPNL Decidability and Complexity

Proposition 4.3.4. Let $M$ be a spatial model and $\langle (h_a, v_b), (h_c, v_d) \rangle$ be one of its objects, with $h_a \neq 0$ or $v_b \neq 0$. It holds that if $M, \langle (h_a, v_b), (h_c, v_d) \rangle \vdash \text{wn}(p)$, then $M, \langle (h_a, v_b), (h_c, v_d) \rangle \vdash p$ and, for any object $\langle (h'_a, v'_b), (h'_c, v'_d) \rangle \neq \langle (h_a, v_b), (h_c, v_d) \rangle$, with $h'_a \neq 0$ or $v'_b \neq 0$, $M, \langle (h'_a, v'_b), (h'_c, v'_d) \rangle \nvdash p$.

Proof. Let $M$ be any spatial model and $\langle (h_a, v_b), (h_c, v_d) \rangle$ be one of its objects, with $h_a \neq 0$ or $v_b \neq 0$. Suppose that for a given propositional letter $p$, it is the case that $M, \langle (h_a, v_b), (h_c, v_d) \rangle \vdash \text{wn}(p)$. Clearly, $M, \langle (h_a, v_b), (h_c, v_d) \rangle \vdash p$. Suppose, by contradiction, that for some $h'_a, h'_c, v'_b, v'_d$, with $h'_a \neq 0$ or $v'_b \neq 0$, $M, \langle (h'_a, v'_b), (h'_c, v'_d) \rangle \nvdash p$. Suppose $h_a, h'_a \neq 0$ and $h_a < h'_a$ (the other cases can be treated in a similar way). From $M, \langle (h_a, v_b), (h_c, v_d) \rangle \vdash \text{wn}(p)$, it follows that $M, \langle (h_a, v_b), (h_c, v_d) \rangle \nvdash [\text{WU}]\langle (E)p \rightarrow \neg(E)(E)p \rangle$. Hence, we have that $M, \langle (0, v_b), (h_a, v_d) \rangle \vdash (E)p \rightarrow \neg(E)(E)p$, and, since $(E)p$ is satisfied, we have a contradiction with $M, \langle (h'_a, v'_b), (h'_c, v'_d) \rangle \vdash p$ and thus $\neg(E)(E)p$ does not hold with $\langle (0, v_b), (h_a, v_d) \rangle$. Notice that the case in which $h'_a = h_a$ and $v'_b = v_b$, $\text{wn}(2)p$ comes into play.

As shown in [48], one of the possible measures of the expressive power of a directional-based spatial logic for rectangles is the comparison with Rectangle Algebra (RA) [50]. In RA, one considers a finite set of objects (rectangles) $O_1, \ldots, O_n$ and a set of constraints between each pair of objects. Each constraint is a pair of Allen’s IA relations that capture the relationships between the projections on the $x$- and the $y$-axes of the objects. As an example, $O_1(r_1, r_2)O_2$ means that $r_1$ (resp., $r_2$) is the interval relation between the $x$-projections (resp., $y$-projections) of $O_1$ and $O_2$. In general, given an algebraic constraint network, the main problem is to establish whether the network is consistent, that is, if all constraints can be jointly satisfied. In [48], it has been shown that SpPNL is powerful enough to express and check the consistency of an RA-constraint network. In the following, we show that the same can be done in WSpPNL as well, exploiting the weakly universal operator and the weak nominals\(^1\). To this end, we take advantage of the technique used in [48]. Given an RA-constraint network with objects $O_1, \ldots, O_n$, we introduce a propositional variable for every object and we force it to be a weak nominal. Moreover, we introduce additional nominals for every constraint of the network whenever necessary. In such a way, we are able to represent the network as a conjunction of WSpPNL formulas which is satisfiable if and only if the network is consistent. On the one hand, such an encoding of the consistency problem involves a blow-up in computational complexity: while an RA-constraint network can be checked for consistency in NP-time, the satisfiability problem for WSpPNL is, as we will see, NEXPTIME-complete. On the other hand, WSpPNL allows one to express a number of conditions, such as, for instance, arbitrary logical disjunctions, negations, and universal properties [26], that cannot be encoded in an RA-constraint network. Let us show now, as a source of exemplification, how WSpPNL can express the RA-constraint $O_1(d, b)O_2$ between two objects $O_1$ and $O_2$, that is, the $x$-projection (resp., $y$-) of $O_1$ is during (resp., before) the $x$-projection (resp., $y$-) of $O_2$. Let $O_1, O_2$, and $O_{d, b}$ be three propositional variables and let $\langle E \rangle \psi$ be a shorthand for $\langle N \rangle \psi \lor \langle N \rangle (\langle E \rangle \psi \lor \langle E \rangle \psi)$. The constraint “there exist two objects $O_1$ and $O_2$ such that $O_2(d, b)O_2$” can be expressed by the following WSpPNL formula:

\[
\langle E \rangle \text{wn}(O_1) \lor \langle E \rangle \text{wn}(O_2) \lor \langle E \rangle \text{wn}(O_{d, b}) \land [\text{WU}](O_1 \rightarrow \langle N \rangle (O_{d, b}) \land [\text{WU}](O_2 \rightarrow \langle E \rangle O_{d, b}) \land [\text{WU}](O_{d, b} \rightarrow \langle E \rangle (O_1 \rightarrow \langle N \rangle (O_1 \rightarrow \langle N \rangle (O_2)) \land [\text{WU}](O_{d, b} \rightarrow \langle E \rangle O_{d, b}) \land [\text{WU}](O_2 \rightarrow \langle E \rangle (O_1 \rightarrow \langle N \rangle (O_1 \rightarrow \langle N \rangle (O_2)) \land [\text{WU}](O_{d, b} \rightarrow \langle E \rangle O_{d, b}) \land [\text{WU}](O_2 \rightarrow \langle E \rangle O_{d, b}) \land [\text{WU}](O_{d, b} \rightarrow \langle E \rangle (O_1 \rightarrow \langle N \rangle (O_1 \rightarrow \langle N \rangle (O_2))))
\]

4.4 WSpPNL Decidability and Complexity

In this section we prove some basic results which are instrumental to the development of a sound and complete (terminating) tableau system for WSpPNL. Let $\varphi$ be an WSpPNL-formula to be checked for satisfiability and let $AP$ be the set of its propositional variables.

\(^1\)It worth pointing out that the restriction to frames based on natural numbers and the exclusion of objects with left bottom corner equal to (0,0) or (0,0) do not change the status of the network (consistent/inconsistent).
Definition 4.4.1. The closure $\text{CL}(\varphi)$ of $\varphi$ is the set of all sub-formulas of $\varphi$ and of their negations (we identify $\lnot \psi$ with $\psi$). The set of horizontal (resp., vertical) spatial requests of $\varphi$ is the set $\text{HR}(\varphi)$ (resp., $\text{VR}(\varphi)$) of all horizontal (resp., vertical) spatial formulas in $\text{CL}(\varphi)$, that is, $\text{HR}(\varphi) = \{(E)\psi, (E)\lnot \psi \in \text{CL}(\varphi)\}$ (resp., $\text{VR}(\varphi) = \{(N)\psi, (N)\lnot \psi \in \text{CL}(\varphi)\}$).

Let $|\varphi|$ (the size of $\varphi$) be the number of symbols of $\varphi$. By induction on the structure of $\varphi$, we can easily prove the following proposition.

Proposition 4.4.2. For every formula $\varphi$, $|\text{CL}(\varphi)|$ is less than or equal to $2 \cdot |\varphi|$, while $|\text{HR}(\varphi)|$ and $|\text{VR}(\varphi)|$ are less than or equal to $2 \cdot |\varphi| - 2$.

Definition 4.4.3. A $\varphi$-atom is a set $A \subseteq \text{CL}(\varphi)$ such that:

- for every $\psi \in \text{CL}(\varphi)$, $\psi \in A$ iff $\lnot \psi \notin A$;
- for every $\psi_1 \lor \psi_2 \in \text{CL}(\varphi)$, $\psi_1 \lor \psi_2 \in A$ iff $\psi_1 \in A$ or $\psi_2 \in A$.

We denote the set of all $\varphi$-atoms by $A_{\varphi}$. We have that $|A_{\varphi}| \leq 2^{|\varphi|}$. Atoms are connected by the following binary relations.

Definition 4.4.4. Let $R^h_{\varphi}$ (resp., $R^v_{\varphi}$) be a binary relation over $A_{\varphi}$ such that, for every pair of atoms $A, A' \in A_{\varphi}$, $A R^h_{\varphi} A'$ (resp., $A R^v_{\varphi} A'$) if and only if, for every $E\psi \in \text{CL}(\varphi)$ (resp., $(N)\psi \in \text{CL}(\varphi)$), if $E\psi \in A$ (resp., $(N)\psi \in A$), then $\psi \in A'$.

We now introduce a suitable labelling of spatial structures based on $\varphi$-atoms.

Definition 4.4.5. We define a $\varphi$-labelled spatial structure (LSS for short) as a pair $L = (\{F, O(F)\}, \text{L})$, where $\{F, O(F)\}$ is a spatial structure and $\text{L} : O(F) \rightarrow A_{\varphi}$ is a labelling function such that, for every pair of objects $((h_a, v_b), (h_c, v_d))$ and $((h_e, v_e), (h_r, v_g))$, $L((h_a, v_b), (h_e, v_e)) \in R^h_{\varphi} L(((h_c, v_d), (h_r, v_g)))$, and for every pair of objects $((h_a, v_b), (h_e, v_e))$ and $((h_c, v_d), (h_r, v_g))$, $L(((h_a, v_b), (h_c, v_d))) \in R^v_{\varphi} L(((h_e, v_e), (h_r, v_g)))$.

An LSS $L$ is said to be horizontally (resp., vertically) fulfilling if and only if, for every horizontal (resp., vertical) formula of the type $(E)\psi \in \text{CL}(\varphi)$ (resp., $(N)\psi \in \text{CL}(\varphi)$), if for every horizontal (resp., vertical) object $((h_a, v_b), (h_e, v_e))$, $((h_c, v_d), (h_r, v_g))$, $(h_c, v_d))$ (resp., $(h_e, v_e), (h_r, v_g))$) such that $\psi \in L(((h_a, v_b), (h_c, v_d)))$, then there exists an object $((h_c, v_e), (h_r, v_g))$ (resp., $((h_e, v_e), (h_r, v_g))$) such that $\psi \in L(((h_a, v_b), (h_c, v_d)))$.

A formula $\varphi$ is satisfiable if and only if there exists a fulfilling LSS such that $\varphi$ belongs to the labelling of the initial object, as stated by the following theorem.

Theorem 4.4.6. A formula $\varphi$ is satisfiable if and only if there exists a fulfilling LSS $L = (\{F, O(F)\}, \text{L})$, with $\varphi \in L(((0, 0), (1, 1)))$.

Proof. (of Theorem 4.4.6) First, we suppose that $\varphi$ is satisfiable and we show that there exists a fulfilling LSS such that $\varphi \in L(((0, 0), (1, 1)))$. Let $M = (\{F, O(F)\}, V)$ be a model for $\varphi$. We build a fulfilling LSS $L_M = (\{F, O(F)\}, \text{LM})$, based on the same spatial structure as the model $M$, whose labelling function is defined in such a way that for every object $((h_a, v_b), (h_c, v_d))$, $\text{LM}(((h_a, v_b), (h_c, v_d))) = \{\psi \in \text{CL}(\varphi) : M, ((h_a, v_b), (h_c, v_d)) \models \psi\}$). It is immediate to show that $L$ is a fulfilling LSS and that $\varphi \in L(((0, 0), (1, 1)))$. Now, we suppose that $L = (\{F, O(F)\}, \text{L})$ is a fulfilling LSS such that $\varphi \in L(((0, 0), (1, 1)))$. We have to show that $\varphi$ is satisfiable. To this end, we build a model $M_L$ satisfying $\varphi$ as follows. Let $M_L = (\{F, O(F)\}, V_L)$, where $V_L$ is defined in such a way that, for each $p \in A^\varphi$ and each object $((h_a, v_b), (h_c, v_d))$, $p \in V_L, ((h_a, v_b), (h_c, v_d)))$ if and only if $p \in L(((h_a, v_b), (h_c, v_d)))$. It is not difficult to prove by induction on the complexity of the formula that for any $\psi \in \text{CL}(\varphi)$ we have that $M, ((h_a, v_b), (h_c, v_d)) \models \psi$ if and only if $\psi \in L(((h_a, v_b), (h_c, v_d)))$. $\square$
The above theorem reduces the satisfiability problem for $\varphi$ to the problem of finding a fulfilling LSS with the initial object labeled by $\varphi$. From now on, we say that a fulfilling LSS $L$ satisfies $\varphi$ if and only if $\varphi \in \mathcal{L}(((0, 0), (1, 1)))$.

Since fulfilling LSSs satisfying $\varphi$ may be arbitrarily large or even infinite, we must find a way to finitely establish their existence. In the following, we first give a bound on the size of finite fulfilling LSSs and then we show that in the infinite case we can safely restrict ourselves to infinite fulfilling LSSs with a finite bounded representation. To prove these results, we take advantage of the following two fundamental properties of LSSs: (i) the labellings of all objects that share the rightmost horizontal (resp., topmost vertical) coordinate must agree on horizontal (resp., vertical) spatial formulas, that is, for every $v_a, v_b, v_c, v_d, v_r, v_q$, $(E)\psi, (E)\psi \in \mathcal{L}(((v_a, v_b),(v_c, v_d)))$ if and only if $(E)\psi, (E)\psi \in \mathcal{L}(((v_r, v_q),(v_c, v_q)))$ (resp., for every $v_a, v_b, v_c, v_d, v_r, v_q, \psi, [N]\psi, [N]\psi \in \mathcal{L}(((v_a, v_b),(v_c, v_d)))$ if and only if $[N]\psi, [N]\psi \in \mathcal{L}(((v_r, v_q),(v_c, v_q)))$); (ii) $|HF(\varphi)|$ different objects of the type $(v_a, v_b), (v_c, v_q)$ are sufficient to fulfill the existential horizontal formulas belonging to the labeling of an object $(v_a, v_b), (v_c, v_q)$ (and symmetrically for the vertical axis).

**Definition 4.4.7.** Given an LSS $L = (F, \mathcal{O}(F), \mathcal{L})$ and $h_c \in D_h$ (resp., $v_d \in D_v$), we denote by $\text{REQ}_h(h_c)$ (resp., $\text{REQ}_v(v_d)$) the set of all and only the horizontal (resp., vertical) formulas belonging to the labellings of the objects of the type $(v_a, v_b), (v_c, v_q)$). The set $\text{REQ}_h(\varphi)$ (resp., $\text{REQ}_v(\varphi)$) is the set of all possible sets of horizontal (resp., vertical) requests for the formula $\varphi$.

In order to bound the size of finite LSSs that we must take into consideration when checking the satisfiability of a given formula $\varphi$, we determine the maximum number of times that any set in $\text{REQ}_h(\varphi)$ (resp., $\text{REQ}_v(\varphi)$) may appear in a given LSS.

**Definition 4.4.8.** Given an LSS $L = (F, \mathcal{O}(F), \mathcal{L})$, a set of points $D_h \subseteq D_h$ (resp., $D_v \subseteq D_v$), and a set of horizontal (resp., vertical) formulas $\mathcal{R} \subseteq HF(\varphi)$ (resp., $VF(\varphi)$), we say that $\mathcal{R}$ occurs $n$ times in $D_h$ (resp., $D_v$) if and only if there exist exactly $n$ distinct points $h_{i1}, \ldots, h_{in} \in D_h$ (resp., $D_v$) such that $\text{REQ}_h(h_{i1})$ (resp., $\text{REQ}_v(h_{i1})$) = $\mathcal{R}$, for all $1 \leq i \leq n$.

The main technical ingredient of the proof is given by the following lemmas that, given a fulfilling LSS, show when and how it is possible to remove a point from it in such a way that the resulting LSS is still fulfilling. From now on, let $m_h = \frac{|HF(\varphi)|}{2}$ and $m_v = \frac{|VF(\varphi)|}{2}$.

**Lemma 4.4.9.** Let $L = (F, \mathcal{O}(F), \mathcal{L})$ be a fulfilling LSS that satisfies $\varphi$. If there exists a point $h_{i1} \in D_h$, with $h_{i1} > 0$, such that there are at least $m_v \cdot m_h + m_v$ points $0 < h_{i1} < h_{i1}$ and at least $m_h + m_v$ points $h_{i1} < h_{i1}$ such that, for every $j$, $\text{REQ}_h(h_{i1})$ = $\text{REQ}_h(h_{i1})$, then there exists a fulfilling LSS $L = (F, \mathcal{O}(F), \mathcal{L})$ that satisfies $\varphi$, with $D_h = D_h \setminus \{h_{i1}\}$ and $D_v = D_v$.

**Proof.** Let us fix a point $h_{i1} \in D_h$ with set of requests $\mathcal{R}$, and let $h_{i1}, h_{i2}, \ldots, (h_{i1} > 0)$ be points in $D_h$ such that for all $j$, $\text{REQ}_h(h_{i1}) = \mathcal{R}$. Suppose that $|\{h_{i1} \mid j < l\}| \geq m_v \cdot m_h + m_v$ and that
We define $D'_h = (D \setminus \{h_i\}, <)$, and, accordingly, the spatial frame $F'_i$ and set of objects $O(F'_i)$, and $L'_i = L_i|_{O(F'_i)}$ (the restriction of $L$ to the objects in $O(F'_i)$). The tuple $L'_i = (F'_i, O(F'_i), L'_i)$ is obviously an LSS, but it is not necessarily a fulfilling one. The removal of objects such that their leftmost/rightmost horizontal coordinate is $h_i$, can be critical since it may generate three types of defects:

1. there exists a horizontal coordinate $h_a < h_i$ such that $\text{REQ}_{h}(h_a)$ contains some $(E)\psi$ which is not fulfilled anymore because of the removal of some object $\langle (h_a, v_b), (h_i, v_c) \rangle$;
2. there exists a vertical coordinate $v_b$ such that $\text{REQ}_{v}(v_b)$ contains some $(N)\psi$ which is not fulfilled anymore because of the removal of some object $\langle (h_a, v_b), (h_i, v_c) \rangle$;
3. there exists a vertical coordinate $v_c$ such that $\text{REQ}_{v}(v_c)$ contains some $(N)\psi$, which is not fulfilled anymore because of the removal of some object $\langle (h_i, v_a), (h_b, v_c) \rangle$.

As a matter of fact, the removal of an object may sometimes generate more than one type of defect. However, we can deal with each of them separately. In the following, we show how to repair each type of defect.

**Fixing defects of type 1.** Consider some horizontal coordinate $h_a < h_i$ such that the formula $\langle (E)\psi \in \text{REQ}_{h}(h_a) \rangle$ and the removed object $\langle (h_a, v_b), (h_i, v_c) \rangle$ was the only one with leftmost horizontal coordinate $h_a$ containing $\psi$ in its labeling (in $L$). Now consider the sets $\text{REQ}_{h}(h_a)$ and $\text{REQ}_{v}(v_b)$: at most $m_v + m_h - 1$ different objects of the type $\langle (h_a, v_b), (h_i, v_c) \rangle$ are needed to satisfy all horizontal and vertical requests in $\text{REQ}_{h}(h_a)$ and $\text{REQ}_{v}(v_b)$ different from $(E)\psi$. Since there exist $m_v + m_h$ points $h_i$ to the right of $h_a$ such that $\text{REQ}_{h}(h_i) = \text{REQ}_{h}(h_i) = \emptyset$, we have that at least one point $h_i$, with $j > 1$, is “useless”, in the sense that the atom $L_i^f(\langle (h_a, v_b), (h_i, v_c) \rangle)$ either fulfills no horizontal and no vertical request belonging to $\text{REQ}_{h}(h_a)$ or $\text{REQ}_{v}(v_b)$ or it fulfills only requests that are also fulfilled by some other object of the same type. Hence, we can redefine $L_i^f(\langle (h_a, v_b), (h_i, v_c) \rangle)$ by putting $L_i^f(\langle (h_a, v_b), (h_i, v_c) \rangle) = L_i^f(\langle (h_a, v_b), (h_i, v_c) \rangle)$, thus fixing the problem with the horizontal request $(E)\psi$. Notice that, since $\text{REQ}_{h}(h_i) = \text{REQ}_{h}(h_i) = \emptyset$, such a change has no impact on the objects to the right of $h_i$; moreover, since we have used the same bottommost vertical coordinate $v_b$, the change has not an impact on the satisfaction of the vertical requests either.

**Fixing defects of type 2.** Proceed as in the case of defects of type 1.

**Fixing defects of type 3.** Consider some vertical coordinate $v_a$ such that the formula $(N)\psi \in \text{REQ}_{v}(v_a)$, and that the removed object $\langle (h_i, v_a), (h_b, v_c) \rangle$ was the only one with bottommost vertical coordinate $v_a$ containing $\psi$ in its labeling (in $L$). Recall that, by hypothesis, there are at least $m_v \cdot m_h + m_v$ distinct points $h_i$, $h_i > h_i$, with horizontal requests $\emptyset$. Let us consider the horizontal requests $\langle (E)\tau_1, \ldots, (E)\tau_p \rangle \subseteq \text{REQ}_{h}(h_i)$ in $h_i$, where $p \leq m_h$. For each $(E)\tau_k$ we take an object of the type $\langle (h_i, v_{\tau_k}), (h_i, v_{\tau_k}) \rangle$ containing $\tau_k$ in its labeling (in $L$). Each $v_{\tau_k}$ has at most $m_v$ vertical requests, which are satisfied, in the worst case, by using objects of the type $\langle (h_i, v_{\tau_k}), (h_i, v_{\tau_k}) \rangle$, where $h_i < h_i$. Then, at most $m_v \cdot m_h$ horizontal coordinates are needed to satisfy the vertical requests of the vertical coordinates of the type $v_{\tau_k}$. Let us consider now the vertical coordinate $v_a$. Again, the vertical requests in $\text{REQ}_{v}(v_{\tau_k})$ different from $(N)\psi$ are at most $m_v - 1$, which are satisfied, in the worst case, by using other $m_v - 1$ different horizontal coordinates of the type $h_i < h_i$. Thus, at least one of the horizontal coordinates of the type $h_i$ is “useless” in the sense that it is not needed to satisfy any of the vertical requests in $\text{REQ}_v(v_a)$. We can redefine the labeling $L_i^f(\langle (h_i, v_a), (h_b, v_c) \rangle)$ by putting $L_i^f(\langle (h_i, v_a), (h_b, v_c) \rangle) = L_i^f(\langle (h_i, v_a), (h_b, v_c) \rangle)$, thus fixing the defect. However, in general, such a substitution can introduce a new defect, since there can be some $(E)\theta \in \text{REQ}_{h}(h_k)$ which was satisfied by $\langle (h_k, v_a), (h_b, v_c) \rangle$ and it is not satisfied anymore. Now, since $\text{REQ}_{h}(h_k) = \text{REQ}_{h}(h_k) = \emptyset$, $\theta = \tau_k$ for some $k$. Hence, we can fix this new defect by putting $L_i^f(\langle (h_i, v_{\tau_k}), (h_i, v_{\tau_k}) \rangle) = L_i^f(\langle (h_i, v_{\tau_k}), (h_i, v_{\tau_k}) \rangle)$. By repeating this last substitution in a suitable way at most $m_h$ times, we can fix all new defects that can be possibly introduced. \qed
Lemma 4.4.9 can be intuitively explained as follows. When we remove a “horizontal” point $h_{i_1}$, that is, all points with horizontal coordinate equal to $h_{i_1}$ (in fact, removing $h_{i_1}$ means removing all objects having $h_{i_1}$ as their leftmost or rightmost horizontal coordinate), we can introduce one or more defects in $L$. Such defects can be of different types, depending on which kind of existential formulas are no more satisfied as an effect of the removal of $h_{i_1}$. A defect of type 1 is generated by an $(E)\psi$ formula belonging to the set of requests of a point to the west of $h_{i_1}$. Such a defect can be immediately fixed by taking advantage of the copies of $h_{i_1}$ to the east of it. A defect of type 2 is generated by an $(N)\psi$ formula belonging to the set $\text{REQ}_v(v_a)$, where $v_a$ is the bottommost vertical coordinate of an object with rightmost horizontal coordinate $h_{i_1}$. As in the previous case, such a defect can be immediately fixed by using the copies of $h_{i_1}$ to the east of it. A defect of type 3 is generated by an $(N)\psi$ formula belonging to the set $\text{REQ}_h(v_a)$, where $v_a$ is the bottommost vertical coordinate of an object with leftmost horizontal coordinate $h_{i_1}$. As shown in Fig. 4.2, to fix a defect of this type, we take advantage of the copies of $h_{i_1}$ to the west of it. Replacing $\theta$ by $\psi$ in the labeling of the object $(h_{i_1},v_a,(h_{i_1},v_b))$ may possibly introduce a new defect $(E)\theta$. Thanks to the availability of a sufficient number of copies of $h_{i_1}$ to the west of it, we can guarantee that such a new defect may involve a horizontal request only and it can be solved by forcing $h_{i_1}$ to behave as $h_{i_1}$ behaved.

**Lemma 4.4.10.** Let $L = (F, \mathcal{O}(F), L)$ be a fulfilling LSS that satisfies $\varphi$. If there exists a point $v_i \in D_v$ such that there are at least $m_v \cdot m_h + m_v$ points $v_i < v_i$, and at least $m_h \cdot m_v$ points $v_i < v_i$ such that, for every $j$, $\text{REQ}_v(v_i) = \text{REQ}_h(v_i)$, then there exists a fulfilling LSS $\Gamma = (F, \mathcal{O}(F), \Gamma)$ that satisfies $\varphi$, with $D_h = D_h \setminus \{v_i\}$ and $D_v = D_v \setminus \{v_i\}$.

The above lemmas can be directly exploited to give a bound on finite LSSs.

**Theorem 4.4.11.** Let $L = (F, \mathcal{O}(F), L)$ be a finite fulfilling LSS that satisfies $\varphi$. Then, there exists a finite fulfilling LSS $\Gamma = (F, \mathcal{O}(F), \Gamma)$ that satisfies $\varphi$ such that, for every $h_i \in D_h$ (resp., $v_i \in D_v$), $\text{REQ}_h(h_i)$ occurs at most $m_v \cdot m_h + 2 \cdot m_v + m_h$ times in $D_h \setminus \{0\}$ (resp., $\text{REQ}_v(v_i)$ occurs at most $m_v \cdot m_h + 2 \cdot m_h + m_v$ times in $D_v \setminus \{0\}$).

**Proof.** Let $L = (F, \mathcal{O}(F), L)$ be a finite LSS satisfying $\varphi$. If it already meets the conditions of the theorem, then we are done. Otherwise, we have to show how it is possible to remove exceeding points from $L$ preserving satisfiability. We suppose that $L$ does not meet the horizontal condition and we show how to repair this situation. A similar technique can be applied, if needed, to the vertical component, thus obtaining $\Gamma$ as claimed.

Let $L_0 = L$ and let $\{R_1, R_2, \ldots, R_k\} \subseteq \text{REQ}_h(\varphi)$ be the (arbitrarily ordered) finite set of all and only the sets of requests that occur more than $m = m_v \cdot m_h + 2 \cdot m_v + m_h$ times in $D_h \setminus \{0\}$. We show how to turn $L_0$ into a fulfilling LSS $L_1 = (F_1, O(F_1), L_1)$ satisfying $\varphi$, which, unlike $L_0$, contains exactly $m$ points $h \in D_h \setminus \{0\}$ such that $\text{REQ}_h(h) = R_1$. By iterating such a transformation $k - 1$ times, we can turn $L_1$ into a fulfilling LSS, devoid of exceeding horizontal points, that satisfies $\varphi$. $L_1$ can be obtained as follows. Let $h_{i_1}, h_{i_2}, \ldots, h_{i_n}$, with $h_1 < h_{i_1} < h_{i_2} < \ldots < h_{i_n}$, be the $n$ points in $D_h$, with $n > m$, such that, for each $i$, $\text{REQ}_h(h_{i_1}) = R_1$. Let $l = m_v \cdot m_h + m_v + 1$: we consider the point $h_{i_1}$ and we apply Lemma 4.4.9 to safely eliminate it. By iterating this procedure $n - m$ times, the final LSS $L'$ meets the horizontal condition.

If necessary, we apply the same technique to the vertical component, by taking advantage of Lemma 4.4.10, to obtain $\Gamma$ as claimed.

Infinite structures can be dealt with as follows. First of all, we must distinguish among three types of infinite LSSs, depending on whether only one domain is infinite (and which one) or both. For each of them, we introduce an appropriate representation.

**Definition 4.4.12.** An infinite LSS $L = (F, \mathcal{O}(F), L)$ is horizontally ultimately periodic, with prefix $l_0$ and period $p_h > 0$, if and only if for all $l > l_0$, $\text{REQ}_h(h_l) = \text{REQ}_h(h_{l+p_h})$; it is vertically ultimately periodic, with prefix $l_0$ and period $p_v > 0$, if and only if for all $j > l_0$, $\text{REQ}_v(v_j) = \text{REQ}_v(v_{j+p_v})$; it is simply ultimately periodic if it is (i) both horizontally and vertically ultimately periodic, or (ii) horizontally ultimately periodic and vertically finite, or (iii) horizontally finite and vertically ultimately periodic.
The proof for the infinite case essentially reduces to show that for any infinite fulfilling LSS there exists an equivalent ultimately periodic fulfilling LSS whose horizontal and/or vertical prefixes and periods satisfy suitable bounds. In case of structures which are infinite in one dimension only, say, the horizontal one, the search for an ultimately periodic characterization of this component can be paired with the application of the argument of Theorem 4.4.11 to the other component, say, the vertical one (the case in which the vertical component is infinite and the horizontal one is finite is fully symmetric). Let us assume the finite vertical component to be bounded, that is, for each $\mathbb{N} \setminus \{0\} \ni q \in \mathbb{N} \setminus \{0\}$ occurs at most $m_v \cdot m_h + 2 \cdot m_v + m_v$ times in $\mathbb{N} \setminus \{0\}$. The following theorem holds.

**Theorem 4.4.13.** Let $L = (F, \mathcal{O}(F), \mathcal{L})$ be a horizontally infinite fulfilling LSS that satisfies $\varphi$. Then, there exists a horizontally ultimately periodic fulfilling LSS $\mathcal{L} = (F, \mathcal{O}(F), \mathcal{L})$, with prefix $a_h$ and period $p_h$, that satisfies $\varphi$ such that:

1. for every set of requests $\mathcal{R}$ that occurs only finitely often in $\mathcal{L}$, $\mathcal{R}$ appears at most $m_v \cdot m_h + 2 \cdot m_v + m_h$ times in the set $\{h | j < a_h\}$;
2. for every set of requests $\mathcal{R}$ that occurs infinitely often in $\mathcal{L}$, $\mathcal{R}$ appears at most $m_v \cdot m_h + m_v$ times in the set $\{h | j < a_h\}$;
3. for every pair of points $\mathcal{R}_a, \mathcal{R}_b \in \mathcal{D}_h$, with $\mathcal{R}_a \prec \mathcal{R}_b$, $\mathcal{R}_b \prec \mathcal{R}_{a + p_h}$, if $a \neq b$, then $\mathcal{R}_a(\mathcal{R}_b) \neq \mathcal{R}_b(\mathcal{R}_b)$.

**Proof.** Let $\varphi$ be a satisfiable formula and let $L = (F, \mathcal{O}(F), \mathcal{L})$ be a horizontally infinite fulfilling LSS that satisfies $\varphi$. We define the following sets: $\text{Fin}_h(L) = \{\text{req}_h(h) : \text{there exists a finite number of points } h \in \mathbb{N}_h \text{ such that } \text{req}_h(h) = \text{req}_h(h)\}$ and $\text{Inf}_h(L) = \{\text{req}_h(h) : \text{there exists an infinite number of points } h \in \mathbb{N}_h \text{ such that } \text{req}_h(h) = \text{req}_h(h)\}$. Moreover, let $l$ and $p$ be the following pair of natural numbers:

$$l = |\text{Fin}_h(L)| \cdot (m_v \cdot m_h + 2 \cdot m_v + m_h) + |\text{Inf}_h(L)| \cdot (m_v \cdot m_h + m_v),$$

$$p = |\text{Inf}_h(L)|.$$

We build a horizontally ultimately periodic fulfilling LSS $\mathcal{L}$, with prefix $a_h \leq l$ and period $p_h = p$, that satisfies $\varphi$ (and respects Conditions 1 and 2) as follows.

Let $h_q$ such that $h_q$ is the greatest point in $\mathbb{N}_h$ such that $\text{req}_h(h_q) \in \text{Inf}_h(L)$. For each $h_q$, with $\text{req}_h(h_q) \in \text{Fin}_h(L)$, we can guarantee that there exist at most $m_v \cdot m_h + 2 \cdot m_v + m_h$ occurrences by possibly removing exceeding ones by iteratively applying Lemma 4.4.19. By applying the same lemma, we can guarantee that there exist at most $m_v \cdot m_h + m_v$ occurrences of $h_q$, with $\text{req}_h(h_q) \in \text{Inf}_h(L)$, less than $h_q$. Let $L'$ be the LSS obtained in such a way. It is apparent that $L'$ meets Condition 1 and 2. We now focus our attention on the points greater than $h_q$. By definition of $h_q$, for all $h > h_q$, $\text{req}_h(h) \in \text{Inf}_h(L)$. Consider an arbitrary permutation of the set $\text{Inf}_h(L) = \{a_0, a_1, \ldots, a_{p_h-1}\}$. We construct an LSS $\mathcal{L} = (F, \mathcal{O}(F), \mathcal{L})$ in such a way that, for all $k > 0$, the set of horizontal requests of $h_q + k \equiv k \pmod{p_h}$ be the sequence of the horizontal (resp., vertical) requests in $\mathcal{L}$ in an arbitrary order.

**Horizontal existential requests associated with a point to the left of $h_q$.** Let $h_a \leq h_q$. Since $L'$ is fulfilling, for each $(E)\psi_1 \in \text{req}_h(h_a)$, there exist $v_b, v_d \in \mathbb{D}_v$ and $c, d \in \mathbb{C}$ such that $\psi_1 \in L' \cap \{((h_a, v_b), (h_c, v_d))\}$. If $h_a \leq h_q$, we put $\mathcal{L}'(\{((h_a, v_b), (h_c, v_d))\}) = L' \cap \{((h_a, v_b), (h_c, v_d))\}$. Otherwise, $\text{req}_h(h_a) \in \text{Inf}_h(L)$ and thus there exists $j$ such that $\text{req}_h(h_c') = \mathcal{R}_j$. We set $\mathcal{L}'(\{((h_a, v_b), (h_c, v_d))\}) = L' \cap \{((h_a, v_b), (h_c, v_d))\}$, where $h_c = h_q + j \cdot p_h + j$. Notice that $h_c$ depends on the position of $\mathcal{R}_j$ in the ordered sequence of sets of horizontal requests and on the position of the formula $(E)\psi_1$ in the ordered sequence of existential horizontal requests.

**Vertical existential requests.** Let $\mathbb{D}_v \supseteq \mathbb{D}_v$. Since $L'$ is fulfilling, for each $(N)\psi_1 \in \text{req}_v(v_b)$, there exist $v_a \in \mathbb{D}_v$ and $h_c, h_d \in \mathbb{D}_h$ such that $\psi_1 \in L' \cap \{((h_a, v_b), (h_c, v_d))\}$. Three cases may arise:

- If $h_c \leq h_q$, we put $\mathcal{L}'(\{((h_a, v_b), (h_c, v_d))\}) = L' \cap \{((h_a, v_b), (h_c, v_d))\}$. Three cases may arise:


If $h_a \leq h_q$ and $h_c > h_q$, then $\text{REQ}_h(h_c) \in \text{Inf}_h(L)$ and there exists $j$ such that $\text{REQ}_h(h_c) = \mathcal{R}_j$. We set $\mathcal{L}(\langle (h_a, v_b), (h_c, v_d) \rangle) = \mathcal{L}'((h_a, v_b), (h_c, v_d)), \text{where } \mathcal{L} = h_q + (m_h + 1) \cdot p_h + i \cdot p_n + j$.

If $h_a > h_q$, then $\text{REQ}_h(h_a), \text{REQ}_h(h_c) \subseteq \text{Inf}_h(L)$ and there exist $j', j$ for which $\text{REQ}_h(h_a) = \mathcal{R}_{j'}$ and $\text{REQ}_h(h_c) = \mathcal{R}_j$. We set $\mathcal{L}(\langle (h_a', v_b'), (h_c, v_d') \rangle) = \mathcal{L}'((h_a, v_b), (h_c, v_d)), \text{where } \mathcal{L} = h_q + (m_h + 1) \cdot p_h + i \cdot p_n + j$.

**Horizontal existential requests associated with a point to the right of $h_q$.** Let $h_a > h_q$. We distinguish two cases: (i) $h_a \leq h_q + (m_h + m_v + 1) \cdot p_h$ and (ii) $h_a > h_q + (m_h + m_v + 1) \cdot p_h$.

Such a distinction is necessary to avoid any change in the labeling of objects whose left horizontal coordinate lies in between $h_q + m_h \cdot p_h$ and $h_a + (m_h + 1) \cdot p_h$ (these are the objects that have been previously used to deal with a specific class of vertical requests). Let us consider case (i). By construction, we have that $\text{REQ}_h(h_a) \in \mathcal{L}$ is equal to $\mathcal{R}_{j'}$, with $j' = (h_a - h_q) \mod p_n$. Since $\mathcal{R}_{j'} \in \text{Inf}_h(L)$, there exist infinitely many $h > h_q$ such that $\text{REQ}_h(h) = \mathcal{R}_{j'}$ in $\mathcal{L}'$. Let $h'$ be the first one. Since $\mathcal{L}'$ is fulfilling, for each $\langle \psi_i \rangle \in \text{REQ}_h(h')$, there exist $v_b, v_d \in D_v$ and $h_c \in D_h$ such that $\psi_i \in \mathcal{L}'((h_b', v_b), (h_c, v_d)))$, with $\text{REQ}_h(h_c) = \mathcal{R}_j$ for some $\mathcal{R}_j \in \text{Inf}_h(L)$. We satisfy the formula $\langle \psi_i \rangle \text{REQ}_v(h_a)$ in $\mathcal{L}$ by letting $\mathcal{L}(\langle (h_a, v_b), (h_c, v_d) \rangle) = \mathcal{L}'((h_b', v_b), (h_c, v_d)))$, where $\mathcal{L} = h_q + (m_h + m_v + 1) \cdot p_h + i \cdot p_n + j$. As for case (ii), it suffices to put $\mathcal{L} = h_q + (p_n - j') + i \cdot p_n + j$.

Consider the case of an LSS that is both horizontal and vertical infinite. Let $\text{Fin}_v(L) = \langle \text{REQ}_v(v_1) \rangle$ there exist a finite number of points $v \in D_v$ such that $\text{REQ}_v(v) = \text{REQ}_v(v) \in \text{Inf}_v(L)$ and $\text{Fin}_v(L) = \langle \text{REQ}_v(v) \rangle$. There exist an infinite number of points $v \in D_v$ such that $\text{REQ}_v(v) = \langle \text{REQ}_v(v) \rangle$ we define $l_h = \langle (\text{Fin}_v(L) \cup \text{Fin}_v(L)) \rangle(m_v, m_h + 2 \cdot m_h + m_v), l_v = \langle (\text{Fin}_v(L) \cup \text{Fin}_v(L)) \rangle(m_v, m_h + 2 \cdot m_h + m_v), v_p = \langle (\text{Fin}_v(L) \cup \text{Fin}_v(L)) \rangle$. As before, by applying the Lemma 4.4.9, we can build a fulfilling LSS $\mathcal{L}$ that respect the following properties:

- for every $h_a, h_c \in D_h$ with $h_a < h_c < h_q$ and every $v_b, v_d \in D_v$ with $v_b < v_d \leq v_p$ we have that $\mathcal{L}(\langle (h_a, v_b), (h_c, v_d) \rangle) = \mathcal{L}'((h_a, v_b), (h_c, v_d)))$;
- for every $h_a > h_q$ $\text{REV}_h(h_a) = \mathcal{R}^h_{(h_a - h_q)} \mod p_n + 1$;
- for every $v_b > v_p$ $\text{REQ}_v(v_b) = \mathcal{R}^v_{(v_b - v_p) + p_n}$.

Let $h_a \leq h_q$ consider a formula $\langle \psi_i \rangle \in \text{REQ}_h(h_a)$, since $L'$ is fulfilling exists $v_b, v_d \in D_v$ and $h_c \in D_h$ for which $\psi_i \in \mathcal{L}(\langle (h_a, v_b), (h_c, v_d) \rangle)$ we define suitably $h', v', v''$ and we set $\mathcal{L}(\langle (h_a, v_b), (h_c, v_d) \rangle) = \mathcal{L}'((h_a, v_b), (h_c, v_d)))$.

If $h_c < h_q$ we set $\mathcal{L}(\langle (h_a, v_b), (h_c, v_d) \rangle) = \mathcal{L}'((h_a, v_b), (h_c, v_d)))$ and exists $j$ for which $\text{REQ}_v(v_b) = \mathcal{R}_j^v$ let $v' = v_q + p_n \cdot i \cdot j'$. If $v_b \leq v_q$ set $v'' = v', v'' = v_p + p \cdot i \cdot j'$. Analogously for the vertical case we let $v_q \leq v_p$ consider a formula $\langle \psi_i \rangle \in \text{REQ}_h(v_b)$, since $L'$ is fulfilling exists $v_b, v_d \in D_v$ and $h_c \in D_h$ for which $\psi_i \in \mathcal{L}(\langle (h_a, v_b), (h_c, v_d) \rangle)$ we define suitably $h', v', v''$ and we set $\mathcal{L}(\langle (h_b', v_b), (h', v'', v''') \rangle) = \mathcal{L}'((h_a, v_b), (h_c, v_d)))$. If $h_a \leq h_q$ we set $h' = h_q$ otherwise $\text{REQ}_h(h_a) \in \text{Inf}_h(L)$ and exists $j$ for which $\text{REQ}_h(h_a) = \mathcal{R}_j$ let $h'' = h_q + p_n \cdot i \cdot j'$. If $v_d \leq v_q$ set $v'' = v_d$ otherwise $\text{REQ}_v(v_d) \in \text{Inf}_v(L)$ and exists $j''$ for which $\text{REQ}_v(v_d) = \mathcal{R}_j^v$ let $v'' = v_p + p \cdot i \cdot j''$. Now consider $h_a > h_q$ by construction we have that $\text{REQ}_h(h) = \mathcal{R}_j^h(h - h_q)$, since $L'$ is fulfilling and $\text{REQ}_h(h_a) \in \text{Inf}_h(L')$ exists $v_b', v_d' \in D_v$ and $h_a', h_c' \in D_h$ for which $\psi_i \in \mathcal{L}(\langle (h_a', v_b'), (h_c', v_d') \rangle)$ with $\text{REQ}_h(h_a) = \text{REQ}_h(h_a')$ and $\text{REQ}_h(h_c') \in \text{Inf}_h(L')$ let $\text{REQ}_h(h_c') = \mathcal{R}_j^h$ we define suitably
h', v', v'' and we set \( \overline{\sigma}(\langle h_a, v' \rangle (h', v'')) \langle \langle h'_a, v'_b \rangle, (h'_c, v'_d) \rangle \). Remember that \( h_a \) belongs to the \( u_a = \left\lceil \frac{v_a - h_a}{p_a} \right\rceil \) unfold of the horizontal period. We define \( h' = h_a + p_r \cdot u_a + p_r \cdot (1 + m_r) + p_r \cdot i + j \). If \( v_b < v_p \) set \( v'' = v_p \) otherwise \( \text{REQ}_v(v_b) \in \text{Inf}_v(L) \) and exists \( j' \) for which \( \text{REQ}_v(v_b) = R_{j'} \), let \( v' = h_a + j' \). If \( v_d < v_p \) set \( v'' = v_d \) otherwise \( \text{REQ}_v(v_d) \in \text{Inf}_v(L) \) and exists \( j'' \) for which \( \text{REQ}_v(v_d) = R_{j''} \), let \( v'' = v_d + p_r \cdot i + j'' \). Analogously for the vertical case if \( v_b > v_p \) by construction we have that \( \text{REQ}_v(v_b) = R_{j''} ^{v''} \), since \( L' \) is fulfilling and \( \text{REQ}_v(v_b) \in \text{Inf}_v(L') \) exists \( v''_b, v''_d \in D_v \) and \( h'_a, h'_c \in D_h \) for which \( v_i \in \text{Inf}((\langle h'_a, v'_b \rangle, (h'_c, v'_d) \rangle) \) with \( \text{REQ}_v(v_b) = \text{REQ}_v(v'_b) \) and \( \text{REQ}_v(v'_d) \in \text{Inf}_v(L') \) let \( \text{REQ}_v(v'_d) = R_{j''} \) we define suitably \( v', h', h'' \) and we set \( \overline{\sigma}((\langle h'_a, v'_b \rangle (h', v'')) \langle \langle h'_a, v'_b \rangle, (h'_c, v'_d) \rangle \). It is worth to notice that \( v_b \) belongs to the \( u_b = \left\lceil \frac{v_b - v_a}{p_a} \right\rceil \) unfold of the vertical period. We define \( v' = v_p + p_v \cdot u_b + p_v \cdot (1 + m_v) + p_v \cdot i + j \). If \( h_a \leq h_q \) we set \( h' = h_q \) otherwise \( \text{REQ}_h(h_a) \in \text{Inf}_h(L) \) and exists \( j' \) for which \( \text{REQ}_h(h_a) = R_{j'} \) let \( h' = h_q + j' \). If \( h_c \leq h_q \) we set \( h'' = h_q \) otherwise \( \text{REQ}_h(h_c) \in \text{Inf}_h(L) \) and exists \( j'' \) for which \( \text{REQ}_h(h_c) = R_{j''} \), let \( v'' = h_q + p_h + (p_h \cdot i) + j'' \). Finally we have to define the labeling \( \overline{\sigma} \) for the tuples \( (\langle h_a, v_b \rangle, (h_c, v_d) \rangle) \) that had not been defined above. It is worth to notice that by construction or \( h_c > h_q \) or \( v_d > v_p \). We define the points \( h', v'' \in D_h \) and \( v', v'' \in D_v \) in \( L' \) as follows:

- \( h_a \leq h_q \) then \( h' = h_q \) otherwise \( h' = h_a \) such that \( \text{REQ}_h(h_a) = \text{REQ}_h(h_a) \);
- \( h_c \leq h_q \) then \( h'' = h_q \) otherwise \( h'' = h'_c \) such that \( \text{REQ}_h(h'_c) = \text{REQ}_h(h'_c) \) and \( v_r > h' \);
- \( v_b \leq v_p \) then \( v'' = v_p \) otherwise \( v'' = v'_b \) such that \( \text{REQ}_v(v'_b) = \text{REQ}_v(v'_b) \);
- \( v_d \leq v_p \) then \( v'' = v_d \) otherwise \( v'' = v'_d \) such that \( \text{REQ}_v(v'_d) = \text{REQ}_v(v'_d) \) and \( v_d > v' \).

We define \( \overline{\sigma}((\langle h_a, v_b \rangle, (h_c, v_d) \rangle) = \text{Inf}((\langle v', v'' \rangle, (h'', v''')) \rangle) \). At the end of this infinite procedure we obtain a fulfilling LSS with the desired properties. \( \square \)

By applying a similar process to the vertical component, it is possible to get an ultimately periodic counterpart to any vertically infinite fulfilling LSS (the proof is fully symmetric and thus omitted). Moreover, the process can be generalized to deal with LSS which are infinite in both directions.

**Theorem 4.4.14.** Let \( L = (F, \emptyset(F), L) \) be a horizontally and vertically infinite fulfilling LSS that satisfies \( \varphi \). Then, there exists a horizontally and vertically ultimately periodic fulfilling LSS \( \overline{L} = (\overline{F}, \emptyset(\overline{F}), \overline{L}) \), with prefix \( \overline{a} \) and period \( p_h \), that satisfies \( \varphi \) such that:

1. for every set of requests \( \mathcal{R} \) that occurs only finitely often in \( L \), \( \mathcal{R} \) appears at most \( m_v \cdot m_h + 2 \cdot m_v + m_h \) times in the set \( \{ \overline{a} : a \leq \overline{a} \} \);
2. for every set of requests \( \mathcal{R} \) that occurs infinitely often in \( L \), \( \mathcal{R} \) appears at most \( m_v \cdot m_h + m_v \) times in the set \( \{ \overline{a} : a \leq \overline{a} \} \);
3. for every pair of points \( \overline{a}, \overline{b} \in D_h \), with \( \overline{a} < \overline{a}, \overline{b} \leq \overline{a} + p_h \), if \( a \neq b \), then \( \text{REQ}_h(\overline{a}) \neq \text{REQ}_h(\overline{b}) \).

Hence, the search for LSS (models) satisfying a given WSpPNL-formula can be confined to the structures of Definition 4.4.12.

Taking advantage of Theorem 4.4.11 and Theorem 4.4.13, we can devise a simple decision procedure for WSpPNL, that restricts the search for a model satisfying \( \varphi \) to finite exponential (pseudo-)models (such a decision procedure can be viewed as a generalization of the one for RPNL given in [20]). It immediately follows that the satisfiability problem for WSpPNL is in NEXPTIME. To prove NEXPTIME-hardness, we will reduce the satisfiability problem for RPNL over natural numbers (which has been shown to be NEXPTIME-hard in [20]) to it. As we show in Chapter 3 RPNL is the future fragment of the interval logic of temporal neighborhood. Formulas of RPNL are built on by using propositional variables, logical connectives, and the neighborhood modality (A) according to the grammar:
f := p | ¬f | f ∨ g | ⟨A⟩f

RPNL interpreted over models of the form \( M = ⟨Δ, I([Δ]), V⟩ \), where \( Δ = ⟨D, <⟩ \) is \( N \) or a prefix of it, \( I([Δ]) = [d_1, d_i] \) is the set of all intervals over \( D \), and \( V \) is the evaluation function. The semantics of \( ⟨A⟩ \) is that \( M, [d_1, d_i] \models ⟨A⟩f \) iff \( \exists d_k \in D \), with \( d_k > d_i \), such that \( M, [d_i, d_k] \models f \).

Let us consider now an encoding \( η \) of RPNL formulas into WSpPNL that makes no change to the original formula except for the replacement of \( ⟨A⟩ \) with \( ⟨E⟩ \). It is not difficult to prove the next lemma.

Lemma 4.4.15. Let \( ϕ \) be an RPNL-formula. We have that \( ϕ \) is satisfiable if and only if the WSpPNL-formula \( η(ϕ) ∧ [E][N]⊥ \) is satisfiable.

Hence, we have the following theorem.

Theorem 4.4.16. The satisfiability problem for WSpPNL is \( \text{NEXPTIME-complete} \).

4.5 The Tableau Method

In this section, we present a sound and complete (terminating) tableau method for WSpPNL based on the model-theoretic results of the previous section. We assume the reader to be familiar with the standard notions of decorated tree, node in a tree, leaf, branch, and height of a tree.

Definition 4.5.1. Given any WSpPNL-formula \( ϕ \) to be checked for satisfiability, a tableau for \( ϕ \) is a suitable decorated tree \( T_ϕ \). Each node of \( T_ϕ \) is labelled with a tuple of the type \( (ψ, ([h_a, v_b), (h_c, v_d)]), D_h, D_v) \) where \( ψ \in \text{CL}(ϕ) \), \( D_h \) and \( D_v \) are finite linearly-ordered sets, and \( ([h_a, v_b), (h_c, v_d)] \in \mathcal{O}(F) \), where \( F \) is the spatial frame obtained from \( D_h \) and \( D_v \).

Given a tableau \( T_ϕ \) and a branch \( B \) of it, we denote with \( D^B_h \) (resp., \( D^B_v \)) the linear order \( D_h \) (resp., \( D_v \)) associated with the leaf of \( B \). Moreover, we denote by \( Γ^B_ϕ(([h_a, v_b), (h_c, v_d)])) \) the set \( \{ψ | (ψ, ([h_a, v_b), (h_c, v_d)]), D_h, D_v) \in B \} \). Let \( N = \{n_1, ..., n_k\} \) be a finite set of nodes. We denote by \( B \cdot N \) the expansion \( B \cdot n_1 | ... | n_k \) obtained by adding \( k \) immediate successors to the leaf of \( B \). Given a finite linear order \( D = \{d_1, ..., d_m\} \) and a point \( d \not\in D \), we denote by \( D \cup \{d < d < d_{i+1}\} \) (resp., \( D \cup \{d < d < d_m\} \)) the linear order obtained from \( D \) adding \( d \) in between \( d_i \) and \( d_{i+1} \) (resp., after \( d_m \)).

Given a tableau \( T_ϕ \) for a WSpPNL-formula \( ϕ \) and one of its branches \( B \), for every horizontal (resp., vertical) coordinate \( h \in D^B_h \) (resp., \( v \in D^B_v \)), we define the set of its horizontal (resp., vertical) requests \( \text{REQ}^B_h(h) \) (resp., \( \text{REQ}^B_v(v) \)) as the smallest set satisfying the following properties:

- if exists \( n = ⟨⟨E⟩ψ⟩, ([h_a, v_b), (h_c, v_d)], D^B_h, D^B_v) \) in \( B \), then \( ⟨E⟩ψ \in \text{REQ}^B_h(h) \);
- if exists \( n = ⟨⟨N⟩ψ⟩, ([h_a, v_b), (h_c, v)], D^B_h, D^B_v) \) in \( B \), then \( ⟨N⟩ψ \in \text{REQ}^B_v(v) \);
- if exists \( n = ⟨⟨E⟩ψ⟩, ([h_a, v_b), (h_c, v)], D^B_h, D^B_v) \) in \( B \), then \( ⟨E⟩ψ \in \text{REQ}^B_h(h) \);
- if exists \( n = ⟨⟨N⟩ψ⟩, ([h_a, v_b), (h_c, v)], D^B_h, D^B_v) \) in \( B \), then \( ⟨N⟩ψ \in \text{REQ}^B_v(v) \);
- if exists \( n = ϕ, ([h_a, v_b), (h_c, v)], D^B_h, D^B_v) \) in \( B \) and \( ⟨E⟩ψ \in \text{CL}(ϕ) \), then \( ⟨E⟩ψ \in \text{REQ}^B_h(h) \);
- if exists \( n = ϕ, ([h_a, v_b), (h_c, v)], D^B_h, D^B_v) \) in \( B \) and \( ⟨N⟩ψ \in \text{CL}(ϕ) \), then \( ⟨N⟩ψ \in \text{REQ}^B_v(v) \).

Rules. Let \( T_ϕ \) be a tableau for a WSpPNL-formula \( ϕ \) and let \( B \) be a branch of it. The following rules can be applied to \( B \):

- **Not-rule**: if there exists a node labelled with \( ⟨¬ψ⟩, ([h_a, v_b), (h_c, v)], D_h, D_v) \) and \( ψ \not\in Γ^B_ϕ(([h_a, v_b), (h_c, v)]) \), then expand \( B \) to \( B \cdot n \), where \( n = ⟨ψ, ([h_a, v_b), (h_c, v)], D^B_h, D^B_v) \);

\(^2\text{Hereafter, for the sake of simplicity, we will denote both cases as } D \cup \{d_1 < d < d_{i+1}\} \text{ with the implicit assumption that } d_{i+1} \text{ is missing whenever } d_i = d_m. \)
• **And-rule:** if there exists a node labelled with \((\neg \psi_1 \lor \psi_2), \langle \{h_a, v_b\}, \{h_c, v_d\}\rangle, D_h, D_v) \in B\) and \(\{\psi_1, \neg \psi_2\} \nsubseteq \Gamma_B(\langle \{h_a, v_b\}, \{h_c, v_d\}\rangle)\), then expand \(B\) to \(B \cdot n_1 \cdot n_2\), where \(n_1 = (\neg \psi_1, \langle \{h_a, v_b\}, \{h_c, v_d\}\rangle, D_h^B, D_v^B)\) and \(n_2 = (\neg \psi_2, \langle \{h_a, v_b\}, \{h_c, v_d\}\rangle, D_h^B, D_v^B)\) (if one between \(\neg \psi_1\) and \(\neg \psi_2\) already belongs to \(\Gamma_B(\langle \{h_a, v_b\}, \{h_c, v_d\}\rangle)\), we can avoid to add the corresponding node);

• **Or-rule:** if exists \((\psi_1 \lor \psi_2), \langle \{h_a, v_b\}, \{h_c, v_d\}\rangle, D_h, D_v) \in B\), and \(\{\psi_1, \psi_2\} \cap \Gamma_B(\langle \{h_a, v_b\}, \{h_c, v_d\}\rangle) = \emptyset\), then expand \(B\) to \(B \cdot n_1 \cdot n_2\), where \(n_1 = (\psi_1, \langle \{h_a, v_b\}, \{h_c, v_d\}\rangle, D_h^B, D_v^B)\) and \(n_2 = (\psi_2, \langle \{h_a, v_b\}, \{h_c, v_d\}\rangle, D_h^B, D_v^B)\);

• **DiamondE-rule:** if for some point \(h_a \in D_h^B\) it holds that \((E)\psi \in \text{REQ}^B(h_a)\) and, for every \(h_c \in D_h^B\) with \(h_c \succ h_a\), and every \(v_b, v_d \in D_v^B\), \(\psi \not\in \Gamma_B(\langle \{h_a, v_b\}, \{h_c, v_d\}\rangle)\), then we expand \(B\) as follows. Let \(\vec{h}, v, \vec{v}\) three fresh points. We define the following classes of nodes:

\[
\begin{align*}
\text{n}_1^{(i,j)} &= (\langle \psi, \{h_a, v_1\}, \{h_i, v_j\}\rangle, D_h, D_v); \\
\text{m}_1^{(i,j)} &= (\langle \psi, \{h_a, v_i\}, \{h_i, v_j\}\rangle, D_h, D_v \cup \{v_1 < v' < v_{i+1}\}); \\
\text{m}_i^{(i,j)} &= (\langle \psi, \{h_a, v_i\}, \{h_i, v_j\}\rangle, D_h \cup \{h_i < h < h_{i+1}\}, D_v); \\
\text{m}_i^{(i,j)} &= (\langle \psi, \{h_a, v_i\}, \{h_i, v_j\}\rangle, D_h \cup \{h_i < h < h_{i+1}\}, D_v \cup \{v_1 < v' < v_{i+1}\}); \\
\text{n}_i^{(i,j)} &= (\langle \psi, \{h_a, v_i\}, \{h_i, v_j\}\rangle, D_h \cup \{v_1 < v < v_{j+1}\}); \\
\text{m}_1^{(i,j)} &= (\langle \psi, \{h_a, v_i\}, \{h_i, v_j\}\rangle, D_h \cup \{v_1 < v' < v_{i+1}, v < v_{j+1}\}); \\
\text{n}_1^{(i,j)} &= (\langle \psi, \{h_a, v_i\}, \{h_i, v_j\}\rangle, D_h \cup \{h_i < h < h_{i+1}\}, D_v \cup \{v_1 < v' < v_{i+1}\}); \\
\text{m}_1^{(i,j)} &= (\langle \psi, \{h_a, v_i\}, \{h_i, v_j\}\rangle, D_h \cup \{h_i < h < h_{i+1}\}, D_v \cup \{v_1 < v' < v_{i+1}, v < v_{j+1}\}).
\end{align*}
\]

Let \(N = \{n_1^{(i,j)} | a + 1 \leq i \leq |D_h^B| - 1 \land 0 \leq j \leq 1 \land 1 \leq j \} \cup \ldots \cup \{m_1^{(i,j)} | a + 1 \leq i \leq |D_h^B| - 1 \land 0 \leq j \leq |D_v^B| - 1 \land 1 \leq j\}\). We expand \(B\) to \(B \cdot N\);

• **DiamondN-rule:** analogous to the previous case;

• **BoxE-rule:** if for some point \(h_a \in D_h^B\) we have that \((E)\psi \in \text{REQ}^B(h_a)\) and there exist three points \(h_c \in D_h^B, v_b, v_d \in D_v^B\), such that \(\psi \not\in \Gamma_B(\langle \{h_a, v_b\}, \{h_c, v_d\}\rangle)\), then we expand \(B\) to \(B \cdot n\) where \((\psi, \langle \{h_a, v_b\}, \{h_c, v_d\}\rangle, D_h^B, D_v^B)\);

• **BoxN-rule:** analogous to the previous case.

The behavior of the **DiamondE-rule** can be explained as follows. Suppose that we are trying to build a model for the formula \(\varphi\) and, at a certain stage of the construction, we find an object labeled by \((E)\psi\). We have to foresee all possible ways of satisfying the request \((E)\psi\), namely, \(\psi\) can be satisfied on an object that has been already introduced in the model (node class \(n_1^{(i,j)}\)) or on a new one. In the latter case, the new object can be created by adding at most one point in the horizontal component and at most two points in the vertical one, in all possible ways with respect to the existing points. This forces us to consider seven distinct classes of new nodes.

**Expansion Strategy.** We introduce the notions of fulfilled branch, closed (and open) branch, and blocked (and non-blocked) branch, and describe how the expansion rules must be applied in order to guarantee the completeness of the method. We say that \((E)\psi \in \text{HF}(\varphi)\) (resp., \((N)\psi \in \text{VF}(\varphi)\)) is fulfilled for \(h\) by \(h'\) (resp., fulfilled for \(v\) by \(v'\)) if there exists a node \(n = (\psi, \langle \{h, v_b\}, \{h', v_d\}\rangle, D_h^B, D_v^B)\) (resp., \(n = (\psi, \langle \{h_a, v\}, \{h_c, v'\}\rangle, D_h^B, D_v^B)\)) in \(B\).

**Definition 4.5.2.** Let \(T_\psi\) be a tableau for a WSpPNL-formula \(\varphi\) and \(B\) be one of its branches. We say that \(B\) is horizontally (resp., vertically) fulfilled if there exist two points \(h_p \prec h_q \in D_h^B\) (resp., \(v_p < v_q \in D_v^B\)) such that the following conditions are respected:

1. For every \(h \prec h_q\) (resp., \(v \prec v_q\), every formula \((E)\psi \in \text{REQ}^B(h)\) (resp., \((N)\psi \in \text{REQ}^V(v)\)) is fulfilled in \(B\).
2. For every point $h' \geq h_p$ (resp., $v' \geq v_p$), there exists a point $h'' < h_p$ (resp., $v'' < v_p$) such that $\text{REQ}_h^k(h') = \text{REQ}_h^k(h'')$ (resp., $\text{REQ}_v^k(v') = \text{REQ}_v^k(v'')$);

3. For every point $h' \geq h_q$ (resp., $v' \geq v_q$), there exists a point $h'' \leq h_q$ (resp., $v_p \leq v'' \leq v_q$) such that $\text{REQ}_h^k(h') = \text{REQ}_h^k(h'')$ (resp., $\text{REQ}_v^k(v') = \text{REQ}_v^k(v'')$).

The notion of horizontally (resp., vertically) fulfilled branch can be explained as follows. If each existential formula in $B$ is explicitly fulfilled, we can choose the greatest element of $D_h^B$ (resp., $D_v^B$) as $h_q$ (resp., $v_q$) and any other (distinct) element of $D_h^B$ (resp., $D_v^B$) as $h_p$ (resp., $v_p$). This deals with the case of finite models. If there exist some existential formulas in $B$ which are not explicitly fulfilled, it is possible to show that the satisfaction of the conditions of Definition 4.5.2 guarantees the existence of an infinite model for $\varphi$ (in fact, it allows us to produce a finite representation of an ultimately periodic model for $\varphi$).

**Definition 4.5.3.** Let $T_\varphi$ be a tableau for a WSplNL-formula $\varphi$ and let $B$ be one of its branches. We say that $B$ is closed if and only if there exist four points $h_a, h_c \in D_h^B$ and $v_b, v_d \in D_v^B$ such that $\{\psi, \neg\psi\} \subseteq \Gamma_B((h_a, v_b), (h_c, v_d))$; otherwise, we say that it is open.

**Definition 4.5.4.** Let $T_\varphi$ be a tableau for a WSplNL-formula $\varphi$ and let $B$ be one of its branches. We say that $B$ is blocked if and only if one of the following conditions hold:

1. There exists a point $h \in D_h^B$ such that $\text{REQ}_h^k(h)$ occurs $m_v \cdot m_h + 2 \cdot m_v + m_h + 1$ times in $D_h^B$;

2. There exists a point $v \in D_v^B$ such that $\text{REQ}_v^k(v)$ occurs $m_v \cdot m_h + 2 \cdot m_h + m_v + 1$ times in $D_v^B$.

Given a WSplNL-formula $\varphi$, the initial tableau $T_\varphi$ for $\varphi$ is a single-node tree labelled by $\langle \varphi, \langle (h_0, v_0), (h_1, v_1) \rangle, \{h_0 < h_1\}, \{v_0 < v_1\} \rangle$. We expand the tableau by applying to its open branches $B$ the following rules (in the given order):

1. apply the Not/And/Or-rules until they generate no new nodes in $T_\varphi$;

2. apply the BoxE/BoxN-rules until they generate no new nodes in $T_\varphi$;

3. if $B$ is not blocked and it is not horizontally (resp., vertically) fulfilled, apply the DiamondE-rule (resp., DiamondN-rule) to it.

**Definition 4.5.5.** Given a WSplNL-formula $\varphi$ and a tableau $T_\varphi$ for it, we say that $T_\varphi$ is final if and only if the application of the expansion strategy to every open branch of $T_\varphi$ does not generate new nodes. A final $T_\varphi$ is said to be open if there exists a vertically and horizontally fulfilled open branch $B$ in it.

**Soundness and Completeness.** To prove that the method is sound, we take a open branch $B$, that is both horizontally and vertically fulfilled, and show how to obtain a model for the formula $\varphi$ from it.

**Theorem 4.5.6.** If $\varphi$ is a WSplNL-formula and $T_\varphi$ is an open final tableau for it, then $\varphi$ is satisfiable.

**Proof.** Let $T_\varphi$ be the tableau for $\varphi$. By definition, if $T_\varphi$ is open, then there exists an open branch $B$ in $T_\varphi$ that is both vertical and horizontal fulfilled. We define an LSS $L = (F, \odot(F), L)$ as follows:

- $D_h = (D_h^B, <)$ and $D_v = (D_v^B, <)$;

- for every $\psi \in \text{CL}(\varphi)$ and every object $\langle (h_a, v_b), (h_c, v_d) \rangle$, if $\psi \in \Gamma_B(\langle (h_a, v_b), (h_c, v_d) \rangle)$, then we put $\psi \in L(\langle (h_a, v_b), (h_c, v_d) \rangle)$.
Once this construction is finalized, we have two cases. If each existential formula in \( L \) is explicitly fulfilled, then we are done. Otherwise, let \( h' \in D_h \) be the smallest horizontal point such that there exists a formula \((E)\psi \in REQ_h(h')\) that is not fulfilled for \( h' \), and complete the construction of \( L \) as follows. By definition of horizontally fulfilled we have that there exists \( h_q < h_q \in D_h \) such that \( h' > h_q \) (by Condition 1 of Definition 4.5.2) and a point \( h_p \leq h'' \leq h_q \in D_h \) with \( \text{REQ}_h(h'') = \text{REQ}_h(h') \) such that \((E)\psi \) is fulfilled in \( h'' \) by some point \( \overline{h} > h_p \) (Condition 3 of Definition 4.5.2). We add a fresh point \( \overline{h}' \) in \( D_h \) as the greatest point of the horizontal domain, and we define the labeling as follows:

- \( \mathcal{L}(((a, v_b), (\overline{h}', v_d)) = \mathcal{L}(((a, v_b), (\overline{h}, v_d))) \) for every \( v_b, v_d \in D_v \) and for every \( h_a < \overline{h}' \) with \( h_q \neq h' \);
- \( \mathcal{L}(((h', v_b), (\overline{h}', v_d)) = \mathcal{L}(((h', v_b), (\overline{h}, v_d))) \) for every \( v_b, v_d \in D_v \);
- Since \( \overline{h} > h_p \), then, by Condition 2 of Definition 4.5.2 it follows that for every \( h_a \geq \overline{h} \) there exists \( h_a < h_p \) with \( \text{REQ}_h(h_a) = \text{REQ}_h(h_p) \). Thus, we define the labeling \( \mathcal{L}(((a, v_b), (\overline{h}, v_d)) = \mathcal{L}(((a, v_b), (\overline{h}, v_d))) \) for every \( v_b, v_d \in D_v \) and every \( h_a \geq \overline{h} \) with \( h_q \neq h' \).

The case of \((N)\psi\) formulas requires a symmetric procedure. At the end of this (possibly infinite) procedure, we have that all the \((E)\psi\) and \((N)\psi\) are satisfied for each point in the horizontal and vertical domains. The resulting structure \( L \) is not necessarily an LSS. Indeed, it could be the case that for some object \((h, v_b, (h_c, v_d))\) and some formula \( \psi \in \text{CL}(\phi) \), neither \( \psi \in \mathcal{L}(((h, v_b), (h_c, v_d))) \) nor \( \neg \psi \in \mathcal{L}(((h, v_b), (h_c, v_d))) \). However, \( L \) can be extended to a complete LSS as follows:

- if \( \psi = \phi \) or \( \psi = \neg \phi \), we set \( \neg \phi \in \mathcal{L}(((h, v_b), (h_c, v_d))) \);
- if \( \psi = \neg \phi_1 \), we set \( \phi_1 \in \mathcal{L}(((h, v_b), (h_c, v_d))) \) if and only if \( \psi_1 \notin \mathcal{L}(((h, v_b), (h_c, v_d))) \);
- if \( \psi = \phi_1 \lor \phi_2 \), we set \( \phi_2 \in \mathcal{L}(((h, v_b), (h_c, v_d))) \) if and only if \( \phi_1 \notin \mathcal{L}(((h, v_b), (h_c, v_d))) \) or \( \phi_2 \notin \mathcal{L}(((h, v_b), (h_c, v_d))) \);
- if \( \psi = (E)\phi_1 \), we set \( \phi_1 \in \mathcal{L}(((h, v_b), (h_c, v_d))) \) if and only if there exist \( h_c \in D_h \) and \( v_f, v_g \in D_v \) such that \( \phi_1 \in \mathcal{L}(((h_c, v_f), (h_c, v_g))) \);
- if \( \psi = (N)\phi_1 \), we set \( \phi_1 \in \mathcal{L}(((h, v_b), (h_c, v_d))) \) if and only if there exist \( h_c, h_f \in D_h \)

To prove that the method is complete it suffices to show that, for each LSS satisfying either the conditions of Theorem 4.4.11 or those of Theorem 4.4.13, there exists a corresponding horizontally and vertically fulfilled open branch in the (generated) final tableau \( T_\phi \) for \( \phi \).

**Theorem 4.5.7.** If \( \phi \) is a satisfiable WSpPNL-formula, then there exists a final tableau \( T_\phi \) for it.

**Proof.** Let \( \phi \) be a satisfiable formula and let \( L = (F, \mathcal{O}(F), \mathcal{L}) \) be LSS for \( \phi \). Suppose that \( L \) respects the conditions of Theorem 4.4.11, if it is finite, and of Theorem 4.4.13, otherwise. We show how an open tableau \( T_\phi \) for \( \phi \) can be obtained from \( L \). Since \( L \) is fulfilling, then \( \phi \in \mathcal{L}(((h_0, v_0), (h_1, v_1))) \). We start the construction of \( T_\phi \) with one node initial tableau \((\phi, ((h_0, v_0), (h_1, v_1)), (h_3 < h_1), (v_0 < v_1))) \), and then we proceed in accordance with the expansion strategy. We prove by induction on the number of steps of the tableau construction that the current tableau \( T_\phi \) includes a branch \( B \) which satisfies the following invariant: for every object \(((h, v_b), (h_c, v_d)))\), and every \( \psi \in \text{CL}(\phi) \), if \( \psi \in \Gamma_\phi(((h, v_b), (h_c, v_d))) \) then \( \psi \in \mathcal{L}(((h, v_b), (h_c, v_d))) \). By construction, the initial tableau satisfies the invariant. As for the inductive step, let \( T_\phi \) be the current tableau and let \( B \) be the branch of \( T_\phi \) that satisfies the invariant. The following cases may arise:
The Not-rule is applied to B, that is, \( \neg \psi \in \Gamma_B((\{a_0, v_b\}, \{c, v_d\})) \) and \( \psi \notin \Gamma_B((\{a_0, v_b\}, \{c, v_d\})) \). By the inductive hypothesis, \( \neg \psi \in L((\{a_0, v_b\}, \{c, v_d\})) \). Thus \( \psi \in L(((a_0, v_b), (c, v_d))) \), and the expanded branch \( B \cdot n_1 \) with \( n_1 = (\psi, ((a_0, v_b), (c, v_d)), D^B_{n_1}, D^B_{n_1}) \) satisfies the invariant;

The And-rule is applied to B, that is, \( -(\psi_1 \land \psi_2) \in \Gamma_B((\{a_0, v_b\}, (c, v_d))) \) and \( \neg \psi_1, \neg \psi_2 \notin \Gamma_B(((a_0, v_b), (c, v_d))) \). By definition of \( L \), both \( \neg \psi_1 \in L((\{a_0, v_b\}, (c, v_d))) \) and \( \neg \psi_2 \in L(((\{a_0, v_b\}, (c, v_d))) \). It immediately follows that the expanded branch \( B \cdot n_1 \cdot n_2 \) with \( n_1 = (\neg \psi_1, ((a_0, v_b), (c, v_d))), D^B_{n_1}, D^B_{n_1} \) and \( n_2 = (\neg \psi_2, ((a_0, v_b), (c, v_d)), D^B_{n_2}, D^B_{n_2}) \), satisfies the invariant;

The Or-rule is applied to B, that is, \( \psi_1 \lor \psi_2 \in \Gamma_B((\{a_0, v_b\}, (c, v_d))) \) and \( \psi_1, \psi_2 \notin \Gamma_B((\{a_0, v_b\}, (c, v_d))) \). The expanded branch \( B \cdot n_k \), with \( n_k = (\psi_k, ((a_0, v_b), (c, v_d)), D^B_{n_k}, D^B_{n_k}) \), satisfies the invariant;

The DiamondE-rule is applied to B, that is, there exists \( h_d \in D^B_h \) such that \( (E\psi) \in \text{REQ}_B(h_d) \) and that for every \( h_c \in D^B_h \) with \( h_c > h_a \) and \( v_b, v_d \in D^B_h \) the condition \( \psi \notin \Gamma_B((\{a_0, v_b\}, (c, v_d))) \) holds. Since \( L \) is fulfilling, there exists \( h_c \succ h_a \) and \( v_b < v_d \) such that \( \psi \in L(((a_0, v_b), (c, v_d))) \). The expanded branch B \( n \), with \( n = (\psi, ((a_0, v_b), (c, v_d)), D^B_n, D^B_n) \) (where \( D^B_n \) is \( D^B_h \) suitably modified as explained in Section 4.5, and similarly for \( D^B_n \)), preserves the invariant;

The DiamondN-rule is applied to B. Similar to the above case;

The BoxE-rule is applied to B, that is, \( [\psi] \in \Gamma_B((\{a_0, v_b\}, (c, v_d))) \), and there exists \( h_c \in D^B_h \) with \( v_b < v_d \in D^B_h \) for which \( \psi \notin \Gamma_B((\{a_0, v_b\}, (c, v_d))) \). By inductive hypothesis \( [\psi] \in L(((a_0, v_b), (c, v_d))) \), and, by definition of \( L \), \( \psi \in L(((a_0, v_b), (c, v_d))) \). Then, the expanded branch \( B \cdot n \), with \( n = (\psi, ((a_0, v_b), (c, v_d)), D^B_n, D^B_n) \), preserves the invariant;

The BoxN-rule is applied to B. Similar to the above case.

The branch B defined by the above inductive procedure is, obviously, an open branch of \( T_\varphi \). To conclude the proof, we have to show that it is both vertically and horizontally fulfilling. If L is a finite LSS, then, by Theorem 4.4.11, we can assume that every set of horizontal requests \( \mathcal{R}_n \) occurs at most \( m_v \cdot m_h + 2 \cdot m_v + m_h \) times in \( D^B_h \) and that every set of vertical requests \( \mathcal{R}_n \) occurs at most \( m_v \cdot m_h + 2 \cdot m_h \) times in \( D^B_h \). Hence, B is not blocked and it is both vertically and horizontally fulfilling.

If L is a horizontally infinite LSS, by Theorem 4.4.13 we can assume that L is horizontally ultimately periodic with prefix \( p_1 \) and period \( p_2 \). Moreover, B can be expanded up to the \( (m_v + m_h) \)-th occurrence of the period without being declared blocked. If we consider the first and last points of the second occurrence of the period, that is \( h_p = h_{p_1 + p_2} + 1 \) and \( h_n = h_{p_1 + 2 \cdot p_2} \), we have that they respect Definition 4.5.2, and thus B is horizontally fulfilling and non-blocked. A similar argument can be used to prove that B is vertically fulfilling and non-blocked even when L is a vertically infinite LSS.

In this chapter, we introduced and studied a decidable modal logic for spatial reasoning about directional relations. In particular, we proved its NEXPTIME-completeness and we provided it with an optimal tableau-based decision procedure. The achieved results can be generalized in several directions. First, the logic can be extended to the three-dimensional case, and beyond. The restriction to two dimensions only can be removed as well. In both cases, the resulting logic preserves decidability. Moreover, we believe it is possible to extend it to dense domains without loosing decidability. Other extensions seem to be more problematic from the decidability point of view. We are currently thinking of the possibility of constraining adjacent rectangles to overlap with respect to one dimension or of restricting the east (resp., north) of a rectangle to the area to the north-east of it only and to redefine the semantics of the modal operators accordingly (both restrictions can be easily lifted to the case of higher-dimensional structures).
4. WSpPNL: a decidable spatial extension of PNL
This chapter deals with interval logics of subinterval structures over dense linear orderings. There are three natural definitions of the subinterval relation: reflexive $\subseteq$ (the current interval is a subinterval of itself), proper $\subset$ (subintervals can share one endpoint with the current interval), and strict $\subsetneq$ (both endpoints of the subintervals are strictly inside the current interval). The logic $D_{\subseteq}$ of reflexive subintervals has been studied first by van Benthem in [58], where it is proved that this logic, if interpreted over dense linear orderings, is equivalent to the standard modal logic $\text{S4}$. The connections between the logic of strict subintervals $D_{\subsetneq}$ and the logic of Minkowski spacetime have been explored by Shapirovsky and Shehtman in [56]. The authors proved that the following axiomatic system is sound and complete for $D_{\subsetneq}$ over the class of dense orderings:

- the K axiom
- transitivity: $([D]p \rightarrow [D][D]p)$
- seriality: $(D)\top$
- 2-density: $([D]p_1 \land [D]p_2 \rightarrow (D)((D)p_1 \land (D)p_2))$

By means of a suitable filtration technique, they also proved $D_{\subseteq}$ decidability and 

By means of a suitable filtration technique, they also proved $D_{\subseteq}$ decidability and PSPACE completeness [55, 56].

In this chapter, we develop a sound, complete, and terminating tableau system for $D_{\subseteq}$. In order to prove soundness and completeness, we introduce a kind of finite pseudo-models for $D_{\subseteq}$, called $D_{\subseteq}$-structures, and we show that every formula satisfiable in $D_{\subseteq}$ is satisfiable in such pseudo-models with a bound on their dimension which depends on the size of the formula to be checked for satisfiability, thereby proving small-model property and decidability in PSPACE of $D_{\subseteq}$ (the result established earlier by Shapirovsky and Shehtman by means of filtration). Inter alia, we show that $D_{\subseteq}$ is also the logic of subinterval structures over the interval $[0, 1]$ of the rational line.

Then, we extend our results to the case of the interval logic $D_{\subseteq}$ interpreted in dense interval structures with proper (irreflexive) subinterval relation. $D_{\subseteq}$ differs substantially from $D_{\subseteq}$ and is much more difficult to analyze. The presence of the special families of beginning subintervals and ending subintervals of a given interval in a structure with proper subinterval relation leads to considerable complications in the constructions of both pseudo-models and tableaux with respect to the case of $D_{\subseteq}$. For instance, the formula $(([D]p \land (D)q) \rightarrow (D)((D)p \land (D)q))$ is valid in $D_{\subseteq}$, but not in $D_{\subseteq}$ (in $D_{\subseteq}$, $p$ and $q$ could be satisfied in respectively beginning and ending subintervals only). Furthermore, the formula

$$(D)(p \land [D]q) \land (D)(p \land [D]\neg q) \land [D]\neg((D)(p \land [D]q) \land (D)(p \land [D]\neg q))$$

can only be satisfied in a $D_{\subseteq}$-structure, as it forces $p$ to be true at some beginning and at some ending subintervals, a requirement which cannot be imposed in $D_{\subseteq}$. Note, however, that $D_{\subseteq}$ can refer to beginning or ending subintervals, but it cannot differentiate between them. This is a subtle but crucial detail: as shown by Lodaya [39], the interval logic BE with modalities respectively for beginning and ending subintervals is undecidable over the class of dense orderings.

The chapter is structured as follows. In Section 5.1 we provide some basic definitions, including syntax and semantics of the considered logics. In Section 5.2 we introduce pseudo-models for $D_{\subseteq}$ and we use them to prove its decidability and PSPACE completeness. Moreover, we develop an
optimal tableau-based decision procedure for $D_{\subseteq}$ formulas, which, from a practical point of view, turned out to be much more efficient than a brute enumeration method directly based on the small-model property. In Section 5.3 we deal with the more difficult case of $D_{\subset}$ by providing a non-trivial generalization of the notion of pseudo-model that allows us to prove its decidability and PSPACE completeness. Moreover, we provide an optimal tableau method for it. In Section 5.4 we implement an evaluation of the two tableau methods in Lotrec.

5.1 Preliminaries

Let $D = (D, <)$ be a dense linear order. An interval over $D$ is an ordered pair $[b, c]$, where $b < c$. We denote the set of all intervals over $D$ by $I(D)$. We consider three subinterval relations: the reflexive subinterval relation (denoted by $\sqsubseteq$), defined by $[d_k, d_l] \sqsubseteq [d_i, d_j]$ if $d_k \leq d_l$ and $d_i \leq d_j$, the proper (or irreflexive) subinterval relation (denoted by $\sqsubset$), defined by $[d_k, d_l] \sqsubset [d_i, d_j]$ iff $[d_k, d_j] \sqsubseteq [d_i, d_j]$ and $[d_k, d_l] \neq [d_i, d_j]$, and the strict subinterval relation (denoted by $\subset$), defined by $[d_k, d_l] \subset [d_i, d_j]$ iff $d_i < d_k$ and $d_i < d_j$.

The three modal logics $D_{\subseteq}$, $D_{\subset}$, and $D_{\subset}$ share the same language, consisting of a set $AP$ of propositional letters, the propositional connectives $\neg$ and $\lor$, and the modal operator $(D)$. The other propositional connectives, as well as the logical constants $\top$ (true) and $\bot$ (false) and the dual modal operator $(D)$, are defined as usual. Formulae are defined by the following grammar:

$$\varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid (D)\varphi.$$  

The semantics of $D_{\subseteq}$, $D_{\subset}$, and $D_{\subset}$ only differ in the interpretation of the $(D)$ operator. For the sake of brevity, we use $\sim \in \{\subseteq, \subset, \subset\}$ as a shorthand for either one of the three subinterval relations. The semantics of a subinterval logic $D_{\sim}$ is based on interval models $M = (I(D), \sim, \mathcal{V})$. The valuation function $\mathcal{V}: AP \rightarrow 2^{I(D)}$ assigns to every propositional variable $p$ the set of intervals $\mathcal{V}(p)$ over which it holds. The satisfiability relation $\models$ is recursively defined as follows:

- for every propositional variable $p \in AP$, $M, [d_1, d_j] \models p$ iff $[d_1, d_j] \in \mathcal{V}(p)$;
- $M, [d_1, d_j] \models \neg \psi$ iff $M, [d_1, d_j] \not\models \psi$;
- $M, [d_1, d_j] \models \psi_1 \lor \psi_2$ iff $M, [d_1, d_j] \models \psi_1$ or $M, [d_1, d_j] \models \psi_2$;
- $M, [d_1, d_j] \models (D)\psi$ iff $\exists [d_k, d_i] \in I(D)$ such that $[d_k, d_i] \sim [d_1, d_j]$ and $M, [d_k, d_i] \models \psi$.

A $D_{\sim}$-formula is $D_{\sim}$-satisfiable if it is true in some interval in some interval model and it is $D_{\sim}$-valid if it is true in every interval in every interval model.

As already pointed out in the introduction, the logics $D_{\subseteq}$ turn out to be equivalent to the standard modal logic S4. Hence, hereafter we will concentrate our attention on the two logics $D_{\subseteq}$ and $D_{\subset}$.

We introduce now some basic definitions and notation that will be extensively used in the following. Given a $D_{\sim}$-formula $\varphi$, the definitions of $CL(\varphi), \varphi$-atom and $A_{\varphi}$ are identical to the corresponding ones introduced in Section 2.2.

Atoms are connected by the following binary relation $D_{\varphi}$.

**Definition 5.1.1.** Let $D_{\varphi}$ be a binary relation over $A_{\varphi}$ such that, for every pair of atoms $A, A' \in A_{\varphi}$, $D_{\varphi} A A'$ holds if and only if for every formula $[D]\psi \in A$, both $\psi \in A'$ and $[D]\psi \in A'$.

Let $A$ be a $\varphi$-atom. We denote the set $\langle [D]\psi \in CL(\varphi) : (D)\psi \in A \rangle$ of temporal requests of $A$ by $\text{REQ}(A)$. We have that if $(D)\psi \notin \text{REQ}(A)$, then $[D]\neg \psi \in A$ (by definition of $\varphi$-atom) and thus $\text{REQ}(A)$ identifies all temporal formulae in $A$. We denote by $\text{REQ}_{\varphi}$ the set of all $(D)$-formulae in $CL(\varphi)$.

Both logics $D_{\subseteq}$ and $D_{\subset}$ will be interpreted over special classes of directed graphs. A directed graph is a pair $G = (V, E)$, where $V$ is a set of vertices and $E \subseteq V \times V$ is a set of edges. We say that
a vertex $v \in V$ is reflexive if the edge $(v, v)$ belongs to $E$, otherwise we say that $v$ is an irreflexive vertex. Given two vertices $v, v' \in V$, we say that $v'$ is a successor of $v$ if the edge $(v, v')$ belongs to the graph and we say that $v'$ is a descendant of $v$ if $v'$ is reachable from $v$ via a path of successors.

The size of a graph $G = \langle V, E \rangle$ depends on two parameters: (i) the breadth of the graph, which is the maximum number of outgoing edges of a vertex, and (ii) the depth of the graph, which is the maximum length of a repetition-free path of vertices. Both parameters will come into play in the decidability proofs for both $D_{\subset}$ and $D_{\subset^1}$: we will show that a formula is satisfiable if and only if it is satisfiable in a graph of bounded breadth and depth.

### 5.2 The logic $D_{\subset}$ of strict subinterval structures

#### 5.2.1 Structures for $D_{\subset}$ formulas

To devise a decision procedure for $D_{\subset}$, we interpret it over a special class of graphs, that we call $D_{\subset}$-graphs. We will prove that a $D_{\subset}$ formula is satisfiable in a dense interval structure if and only if it is "satisfiable" in a $D_{\subset}$-graph. As matter of fact, it will turn out that this is equivalent to satisfiability in an interval structure over the interval $[0, 1]$ of the rational line.

**Definition 5.2.1.** A finite directed graph $G = \langle V, E \rangle$ is a $D_{\subset}$-graph if (and only if) the following conditions hold:

1. there exists an irreflexive vertex $v_0 \in V$, called the root of $G$, such that any other vertex $v \in V$ is reachable from it;
2. every irreflexive vertex $v \in V$ has a unique successor $v_D$, which is reflexive;
3. every successor of a reflexive vertex $v$, different from it, is irreflexive.

An example of $D_{\subset}$-graph is depicted in Figure 5.1. $D_{\subset}$-graphs are finite by definition, but they may include loops involving irreflexive vertices.

![Figure 5.1: An example of $D_{\subset}$-graph.](image)

A $D_{\subset}$-structure is a $D_{\subset}$-graph paired with a labeling function that associates an $A_\varphi$ atom with every vertex in the graph. It is formally defined as follows.

**Definition 5.2.2.** A $D_{\subset}$-structure is a pair $S = \langle \langle V, E \rangle, \mathcal{L} \rangle$, where $\langle V, E \rangle$ is a $D_{\subset}$-graph and $\mathcal{L} : V \rightarrow A_\varphi$ is a labeling function that assigns to every vertex $v \in V$ an atom $\mathcal{L}(v)$ such that, for every edge $(v, v') \in E$, $\mathcal{L}(v) D_\varphi \mathcal{L}(v')$. Let $v_0$ be the root of $\langle V, E \rangle$. If $\varphi \in \mathcal{L}(v_0)$, we say that $S$ is a $D_{\subset}$-structure for $\varphi$.

$D_{\subset}$-structures can be viewed as tentative ‘pseudo-models’ for $D_{\subset}$. Formulas devoid of temporal operators are satisfied by definition of $\varphi$-atom; moreover, $[D]$ formulas are satisfied by definition of $D_\varphi$. To guarantee the satisfiability of $[D]$ formulas, we introduce the notion of fulfilling $D_{\subset}$-structures.

**Definition 5.2.3.** A $D_{\subset}$-structure $S = \langle \langle V, E \rangle, \mathcal{L} \rangle$ is fulfilling if and only if for every vertex $v \in V$ and every formula $[D] \psi \in \mathcal{L}(v)$, there exists a descendant $v'$ of $v$ such that $\psi \in \mathcal{L}(v')$. 
A fulfilling $D_\Sigma$-structure for the formula $\varphi = (D)p \land (D)\neg p \land (D)q$ is shown in Figure 5.2, where we label each node with the associated atom.

![Diagram of a structure](image)

**Figure 5.2:** A fulfilling $D_\Sigma$-structure for the formula $\varphi = (D)p \land (D)\neg p \land (D)q$.

**Theorem 5.2.4.** Let $\varphi$ be a $D_\Sigma$ formula which is satisfied by a strict interval model. Then, there exists a fulfilling $D_\Sigma$-structure $S = \langle (V,E),\mathcal{L} \rangle$ such that $\varphi \in \mathcal{L}(v_0)$, where $v_0$ is the root of $(V,E)$.

**Proof.** Let $M = \langle I(V),\subset,\land,\lor,\neg,\vDash \rangle$ be a strict interval model and let $[b_0, e_0] \in I(V)$ be an interval such that $M,[b_0, e_0] \models \varphi$. We recursively build a fulfilling $D_\Sigma$-structure $S = \langle (V,E),\mathcal{L} \rangle$ for $\varphi$ as follows.

We start with the one-node graph $\langle (V_0),\emptyset \rangle$ and the labeling function $\mathcal{L}$ such that $\mathcal{L}(v_0) = \{ \varphi \in CL(\varphi) : M,[b_0, e_0] \models \varphi \}$.

Next, for every formula $(D)\psi \in \mathcal{L}(v_0)$, we pick up an interval $[b_1, e_1] \subset [b_0, e_0]$ and $M,[b_1, e_1] \models \psi$. Since $D$ is a dense ordering and $CL(\varphi)$ is a finite set of formulas, there exist two intervals $[b_1, e_1]$ and $[b_2, e_2]$ such that:

- $[b_2, e_2] \subset [b_1, e_1] \subset [b_0, e_0]$;
- for every interval $[b_0, e_0],[b_1, e_1],[b_2, e_2]$ satisfies the same formulas of $CL(\varphi)$.

Since $M$ is a model and $[b_0, e_0] \subset [b_2, e_2]$ for every interval $[b_0, e_0],[b_2, e_2]$ satisfies $(D)\psi$ for every $(D)\psi \in \mathcal{L}(v_0)$. Moreover, since $[b_1, e_1]$ and $[b_2, e_2]$ satisfy the same formulas of $CL(\varphi)$ and $[b_2, e_2] \subset [b_1, e_1]$, for every $(D)\psi \in CL(\varphi)$, if $[b_2, e_2]$ satisfies $(D)\psi$, then it satisfies $\psi$ as well.

Accordingly, we add a new (reflexive) vertex $v_D$ and the edges $\langle v_0,v_D \rangle$ and $\langle v_D,v_D \rangle$ to the graph and we label $v_D$ by $\mathcal{L}(v_D) = \{ \xi \in CL(\varphi) : M,[b_2, e_2] \vDash \xi \}$. Furthermore, for every interval $[b_2, e_2]$, we add a new (irreflexive) vertex $v_{\psi}$, together with the edge $\langle v_D,v_{\psi} \rangle$, and we label it by $\mathcal{L}(v_{\psi}) = \{ \xi \in CL(\varphi) : M,[b_2, e_2] \vDash \xi \}$. Then, to obtain a $D_\Sigma$-structure for $\varphi$, we recursively apply the above construction to the vertices $v_{\psi_1},\ldots,v_{\psi_n}$.

To keep the construction finite, whenever the above procedure requests us to introduce a successor $v'$ of a reflexive (resp., irreflexive) node $v \in V$, but there exists an irreflexive (resp., reflexive) node $v' \in V$ such that $\mathcal{L}(v') = \mathcal{L}(v')$, we replace the addition of the node $v'$ with the addition of an edge from $v$ to $v'$. Since the set of atoms is finite, this guarantees the termination of the construction process.

Let $S$ be a fulfilling $D_\Sigma$-structure for a formula $\varphi$. We will prove that $\varphi$ is satisfiable in a strict interval structure. Moreover, we will show that such a structure can be constructed on the interval
[0, 1] of the rational line. To begin with, we define a function \( \mathcal{S} \) connecting intervals in \( \mathbb{I}([0, 1]) \) to vertices in \( S \). Such a function will allow us to define a model for \( \varphi \).

**Definition 5.2.5.** Let \( S = \langle (V, E), \mathcal{L} \rangle \) be a \( D_\mathbb{C} \)-structure. The function \( \mathcal{S} : \mathbb{I}([0, 1]) \rightarrow V \) is recursively defined as follows:

- \( \mathcal{S}([0, 1]) = v_0 \);
- let \([b, e] \) be an interval such that \( \mathcal{S}([b, e]) = v \) and \( \mathcal{S} \) has not been yet defined over any of its subinterval. We distinguish two cases. If \( v \) is irreflexive, let \( v_D \) be its unique reflexive successor. If \( v \) is reflexive, let \( v_D = v \). Two alternatives must be taken into consideration.
  1. \( v_D \) has no successors different from itself. In such a case, we put \( \mathcal{S}([b', e']) = v_D \) for every proper subinterval \([b', e'] \) of \([b, e] \).
  2. \( v_D \) has at least one successor different from itself. Let \( v^1_D, \ldots, v^k_D \) be the successors of \( v_D \) different from \( v_D \). We consider the intervals defined by the points \( b, b + p, b + 2p, \ldots, b + 2kp, b + (2k + 1)p = e \), with \( p = \frac{e - b}{2k + 1} \). The function \( \mathcal{S} \) over such intervals is defined as follows:
     - for every \( i = 1, \ldots, k \), we put \( \mathcal{S}([b + (2i - 1)p, b + 2ip]) = v^i_D \).
     - for every \( i = 0, \ldots, k \), we put \( \mathcal{S}([b + 2ip, b + (2i + 1)p]) = v_D \).

We complete the construction by putting \( \mathcal{S}([b', e']) = v_D \) for every subinterval \([b', e'] \) of \([b, e] \) which is not a subinterval of any of the intervals \([b + ip, b + (1 + 1)p] \).

The structure induced by Definition 5.2.5 is depicted below.

\[
\begin{array}{c}
\vdots \\
v_D \\
v_B \\
v_{b'} \\
v_B \\
v_{b''} \\
\vdots \\
v_D \\
v_e \\
\end{array}
\]

It is easy to show that \( \mathcal{S} \) satisfies the following properties (proof omitted).

**Lemma 5.2.6.**

1. For every pair of intervals \([b, e], [b', e'] \) \( \in \mathbb{I}([0, 1]) \) such that \([b', e'] \subseteq [b, e] \), \( \mathcal{S}([b', e']) \) is reachable from \( \mathcal{S}([b, e]) \).

2. For every interval \([b, e] \) \( \in \mathbb{I}([0, 1]) \), if \( \mathcal{S}([b, e]) = v \) and \( v' \) is reachable from \( v \), then there exists \([b', e'] \subseteq [b, e] \) such that \( \mathcal{S}([b', e']) = v' \).

Given a fulfilling \( D_\mathbb{C} \)-structure \( S \) for \( \varphi \), let \( M_\varphi \) be the triplet \( \langle \mathbb{I}([0, 1]), \mathcal{E}, V \rangle \), where \( V(p) = \{[b, e] : p \in \mathcal{L}(\mathcal{S}([b, e])) \} \) for every \( p \in \mathcal{A}_\varphi \). It turns out that \( M_\varphi \) is a model for \( \varphi \).

**Theorem 5.2.7.** Let \( S \) be a fulfilling \( D_\mathbb{C} \)-structure for \( \varphi \). Then, \( M_\varphi \models [0, 1] \vdash \varphi \).

**Proof.** We prove that for every interval \([b, e] \in \mathbb{I}([0, 1]) \) and every formula \( \psi \in \mathcal{L}(\mathcal{S}([b, e])) \), \( M_\varphi \models [b, e] \vdash \psi \) if and only if \( \psi \in \mathcal{L}(\mathcal{S}([b, e])) \). The proof is by induction on the structure of the formula.

- the case of propositional letters as well as those of Boolean connectives are straightforward and thus omitted;
- let \( \psi = (D)\xi \), and suppose that \( \psi \in \mathcal{L}(\mathcal{S}([b, e])) \). Since \( S \) is fulfilling, there exists a vertex \( v' \), which is reachable from \( \mathcal{S}([b, e]) \), such that \( \xi \in \mathcal{L}(v') \). By Lemma 5.2.6, there exists \([b', e'] \subseteq [b, e] \) such that \( \mathcal{S}([b', e']) = v' \). By inductive hypothesis, \( M_\varphi \models [b', e'] \vdash \xi \) and thus \( M_\varphi \models [b, e] \vdash (D)\xi \).

To prove the opposite implication, suppose by reductio ad absurdum that \( M_\varphi \models (D)\xi \rightarrow (D)\xi \), but \( (D)\xi \not\in \mathcal{L}(\mathcal{S}([b, e])) \). By definition of \( \varphi \)-atom, this implies that \( \mathcal{S}(\xi) \in \mathcal{L}(\mathcal{S}([b, e])) \). Thus, by Lemma 5.2.6, we have that, for every \([b', e'] \subseteq [b, e] \), \( (D)\xi \notin \mathcal{L}(\mathcal{S}([b', e'])) \). By inductive hypothesis, this implies that \( M_\varphi \models [b', e'] \vdash \neg \xi \) for every \([b', e'] \subseteq [b, e] \), which is a contradiction.
Let $v_0$ be the root of $S$. Since $\varphi \in \mathcal{L}(v_0)$ and $\mathcal{G}([0, 1]) = v_0$, it immediately follows that $M_\varphi, [0, 1] \models \varphi$. \hfill $\square$

In Figure 5.3, we show how to turn the fulfilling $D_\subseteq$-structure for the formula $\varphi = (D)p \land (D)\neg p \land (D)q$ depicted in Figure 5.2 into a model for $\varphi$.

### 5.2.2 A small-model theorem for $D_\subseteq$-structures

In this section we prove a small-model theorem for $D_\subseteq$, that is, we show that a $D_\subseteq$ formula is satisfiable if and only if there exists a fulfilling $D_\subseteq$-structure of bounded size. In particular, we demonstrate that the breadth and the depth of such a fulfilling $D_\subseteq$-structure are linear in the size of the formula.

**Theorem 5.2.8.** For every satisfiable $D_\subseteq$ formula $\varphi$, there exists a fulfilling $D_\subseteq$-structure whose breadth and depth are bounded by $2 \cdot |\varphi|$.

**Proof.** Let $S = (\langle V, E \rangle, \mathcal{L})$ be a fulfilling $D_\subseteq$-structure for $\varphi$. The following algorithm builds a fulfilling $D_\subseteq$-structure $S' = (\langle V', E' \rangle, \mathcal{L}')$ for $\varphi$ with the requested property.

**Initialization.** Initialize $S'$ as the one-vertex $D_\subseteq$-structure $(\langle \{v_0\}, \emptyset \rangle, \mathcal{L}')$, where $v_0$ is the root of $S$ and $\mathcal{L}'(v_0) = \mathcal{L}(v_0)$. Call the procedure $\text{Expansion}$ on $v_0$.

**Expansion($v$).** If $v$ is irreflexive, execute Step 1; otherwise, execute Step 2.

**Step 1.** Let $v'$ be unique reflexive successor of $v$ in $S$. Add $v'$ to $V'$ and $(v, v'), (v', v')$ to $E'$; moreover, put $\mathcal{L}'(v') = \mathcal{L}(v')$. Call the procedure $\text{Expansion}$ on $v'$.

**Step 2.** Let $\text{REQ}(\mathcal{L}'(v)) = \{\langle D \rangle \psi_1, \ldots, \langle D \rangle \psi_k\}$. Since $S$ is fulfilling, for every formula $\langle D \rangle \psi_i \in \text{REQ}(\mathcal{L}'(v))$, there exists a descendant $v_i$ of $v$ in $S$ such that $\psi_i \in \mathcal{L}(v_i)$. For $i = 1 \ldots, k$, add $v_i$ to $V'$ and $(v, v_i)$ to $E'$; moreover, put $\mathcal{L}'(v_i) = \mathcal{L}(v_i)$. Next, for every $v_i$ such that $\text{REQ}(\mathcal{L}'(v_i)) = \text{REQ}(\mathcal{L}'(v))$, add an edge $(v_i, v)$ to $E'$. For the remaining vertices $v_i$, it holds that $\text{REQ}(\mathcal{L}'(v_i)) \subset \text{REQ}(\mathcal{L}'(v))$, because every $[D]$ formula in $\mathcal{L}'(v)$ also belongs to $\mathcal{L}'(v_i)$ and there exists at least one $\psi_i$ such that $\langle D \rangle \psi_i \in \text{REQ}(\mathcal{L}'(v))$ and $[D]\neg \psi_i \in [\mathcal{L}][\mathcal{L}'](v_i)$). For every vertex $v_i$ in this latter set, call the procedure $\text{Expansion}$ on $v_i$.

It is easy to show that the algorithm terminates and that it produces a fulfilling $D_\subseteq$-structure $S'$ for $\varphi$. To prove that both the breadth and the depth of $S'$ are less than or equal to $2 \cdot |\varphi|$, it suffices to observe that:

- every irreflexive vertex has exactly one outgoing edge;
- the number of outgoing edges of reflexive vertices is bounded by the number of $[D]$ formulas in $\text{CL}(\varphi)$, not exceeding the size of the formula;
in Step 2, the procedure \( \text{Expansion} \) is called only on those vertices \( v_i \) such that \( \text{REQ}(L'(v_i)) \subset \text{REQ}(L'(v)) \). It follows that at every step the number of \( (D) \) formulas strictly decreases. As a consequence, we have that every path in \( S' \) devoid of repetitions includes at most \(|\varphi|\) different irreflexive vertices. Since in every repetition-free path reflexive and irreflexive vertices alternate, the depth of the \( D_\subset \)-structure is bounded by \( 2 \cdot |\varphi| \).

\[ \square \]

Given a formula \( \varphi \), let \( n \) be the number of \( (D) \) formulas in \( \text{CL}(\varphi) \). From Theorem 5.2.8, it follows that there exists a fulfilling \( D_\subset \)-structure for \( \varphi \) whose repetition-free vertex paths \( v_0, ..., v_k \), starting from the root, have length at most \( 2n \).

A PSPACE non-deterministic algorithm for satisfiability checking of \( D_\subset \) formulas can be defined as follows.

**procedure** \( D_\subset \)-sat(\( \varphi \))

\[ A = \text{a non-deterministically generated atom containing } \varphi; \]

\( D_\subset \)-step(\( A \));

**procedure** \( D_\subset \)-step(\( A \))

\[ A' = \text{a non-deterministically generated reflexive atom such that } A \models_\varphi A' \text{ and } \text{REQ}(A') \subseteq \text{REQ}(A); \]

\[ \forall(D)\psi \in \text{REQ}(A') \text{ do} \]

\[ A'' = \text{a non-deterministically generated atom such that } A' \models_\varphi A'' \text{ and } \psi \in A''; \]

\[ \text{if } \text{REQ}(A'') \neq \text{REQ}(A') \text{ then } D_\subset \text{-step}(A''); \]

The above procedure fails whenever an atom with the requested properties cannot be generated. It is easy to show that it has a successful (terminating) computation if and only if there exists a fulfilling \( D_\subset \)-structure for \( \varphi \). The procedure does not exceed polynomial space because the number of nested calls of the procedure \( D_\subset \)-step is bounded by \( O(n) \) (the maximum length of a repetition-free path) and every call needs \( O(n) \) memory space for local operations.

### 5.2.3 A tableau method for \( D_\subset \)

A tableau for a \( D_\subset \) formula \( \varphi \) is a finite, tree-like graph, in which every node is a subset of \( \text{CL}(\varphi) \). Nodes are grouped into **macronodes**, that is, finite subtrees of the tableau, dealt with by the expansion rules. Branching inside a macronode corresponds to disjunctions. Macronodes and edges connecting them represent vertices and edges in the \( D_\subset \)-structure for \( \varphi \). We distinguish two types of rules: **local rules**, that generate new nodes belonging to the same macronode, and **global rules**, that generate new nodes belonging to new macronodes.

Given two nodes \( n \) and \( n' \) such that \( n' \) is a descendant of \( n \), we say that \( n' \) is a **local descendant** of \( n \) (or, equivalently, that \( n \) is a **local ancestor** of \( n' \)) if \( n \) and \( n' \) belong to the same macronode and that \( n' \) is a **global descendant** of \( n \) (\( n \) is a **global ancestor** of \( n' \)) if \( n \) and \( n' \) belong to different macronodes.

#### Local Rules.

\[ \begin{align*}
\text{(NOT)} & \quad \frac{\neg \psi_1, F}{\psi_1, F} \quad \text{(OR)} \quad \frac{\psi_1 \lor \psi_2, F}{\psi_1, F \mid \psi_2, F} \quad \text{(AND)} \quad \frac{\neg(\psi_1 \lor \psi_2), F}{\neg \psi_1, \neg \psi_2, F}
\end{align*} \]

\[ \text{(REFL)} \quad \frac{[D]\psi, F}{\psi, [D]\psi, F} \quad \text{where } \psi \text{ does not occur in any local ancestor of the node} \]
Global Rules.

\[(2\text{-DENS}) \frac{[D]\psi_1, \ldots, [D]\psi_m, (D)\varphi_1, \ldots, (D)\varphi_n, F}{\psi_1, \ldots, \psi_m, [D]\psi_1, \ldots, [D]\psi_m, (D)\varphi_1, \ldots, (D)\varphi_n} \]
where \(m, n \geq 0\) and \(F\) contains no temporal formulas;

\[(\text{STEP}) \frac{[D]\psi_1, \ldots, [D]\psi_m, (D)\varphi_1, \ldots, (D)\varphi_n, F}{G_1 | \ldots | G_n} \]
where \(m \geq 0\), \(n > 0\), \(F\) contains no temporal formulas and, for
every \(i = 1, \ldots, n\), \(G_i = [\varphi_i, \psi_1, \ldots, \psi_m, [D]\psi_1, \ldots, [D]\psi_m] \)

It goes without saying that the meaning of the vertical dashes in the \text{STEP} rule is different from that in the \text{OR} rule.

Reflexive and irreflexive macronodes.

Like the vertices of a \(D_{\mathbb{C}}\)-graph, macronodes can be either reflexive or irreflexive. A macronode is \emph{irreflexive} if (i) the macronode contains the initial node of the tableau or (ii) the macronode has been created by an application of the \text{STEP} rule. In the other cases, viz., when the macronode has been created by an application of the \(2\text{-DENS}\) rule, the macronode is \emph{reflexive}. A node of the tableau is reflexive (resp., irreflexive) if it belongs to a reflexive (resp., irreflexive) macronode.

Expansion strategy.

Given a formula \(\varphi\), the tableau for \(\varphi\) is obtained from the one-node initial tableau \(\langle\{\varphi\}\rangle, \emptyset\) by recursively applying the following expansion strategy, until it cannot be applied anymore:

1. do not apply the expansion rules to nodes of the tableau containing both \(\psi\) and \(\neg\psi\), for some \(\psi \in \text{CL}(\varphi)\);
2. apply the \text{NOT}, \text{OR}, and \text{AND} rules to both reflexive and irreflexive nodes;
3. apply the \(2\text{-DENS}\) rule to irreflexive nodes;
4. apply the \text{REFL} and \text{STEP} rules to reflexive nodes;

To keep the construction finite, when the application of the \(2\text{-DENS}\) (resp., \text{STEP}) rule to a node \(n\) would generate a new reflexive (resp., irreflexive) node such that there exists another reflexive (resp., irreflexive) node \(n'\) in the tableau with the same set of formulas, add an edge from \(n\) to \(n'\) instead of generating such a new node.

Open and closed tableau.

A node \(n\) in a tableau \(\mathcal{T}\) is \emph{closed} if one of the following conditions holds:

1. there exists a formula \(\psi \in \text{CL}(\varphi)\) such that \(\psi, \neg\psi \in n\);
2. in the tableau construction, the \text{NOT}, \text{OR}, \text{AND}, \(2\text{-DENS}\), or \text{REFL} rules have been applied to \(n\) and all the immediate successors of \(n\) are closed;
3. in the tableau construction, the \text{STEP} rule has been applied to \(n\) and at least one of the immediate successors of \(n\) is closed.

A tableau \(\mathcal{T}\) is \emph{closed} if its initial node is closed; otherwise, it is \emph{open}. 

Example of application.

The construction of the tableau is exemplified in Figure 5.4, where a closed tableau for the un-satisfiable formula \((D)(D)p \land (D)q \land [D][D]\neg p \lor [D]\neg q\) is reported. For the sake of readability, macronodes are represented by boxes and formulas to which local rules are applied are underlined.

Figure 5.4: The tableau for \((D)(D)p \land (D)q \land [D][D]\neg p \lor [D]\neg q\).

### 5.2.4 Termination, complexity, soundness, and completeness

The following theorem ensures the termination of the proposed tableau method:

**Theorem 5.2.9.** Given a \(D_{\subseteq}\) formula \(\varphi\), every tableau \(\mathcal{T}\) for \(\varphi\) is finite.

Theorem 5.2.9 easily follows from an analysis of the construction rules:

- the local rules can be applied only finitely many times to the nodes of a macronode (the application of the NOT, AND, and OR rules to a node generates one or two nodes where one of the formulas in the original node has been replaced by one or two formulas of strictly lower size; moreover, for any branch in the subtree associated with a macronode and any \(D\) formula in \(CL(\varphi)\), the REFL rule can be applied at most one time);

- the 2-DENS (resp., STEP) rule generates a new node only when the set of temporal formulas of such a node is different from that of the other reflexive (resp. irreflexive) nodes in the tableau, and thus it can be applied only finitely many times.

As for the complexity, we have shown that a formula \(\varphi\) is satisfiable if and only if there exists a \(D_{\subseteq}\)-structure for it whose breadth and depth are linear in \(|\varphi|\). The same property holds for the tableau method as formally stated by the following theorem.

**Theorem 5.2.10.** Let \(\mathcal{T}\) be a tableau for a \(D_{\subseteq}\) formula \(\varphi\). Then, the depth and the breadth of \(\mathcal{T}\) are linear in \(|\varphi|\).

It is easy to prove that such a bound holds for any tableau \(\mathcal{T}\) for \(\varphi\). Let \(n\) be the number of \((D)\) formulas in \(CL(\varphi)\). The number of outgoing edges of a node is bounded by \(n\). Moreover, as in the case of \(D_{\subseteq}\)-structures, every repetition-free path of macronodes starting from the root is of length at most \(2 \cdot n\). Hence, both the breadth and the depth of \(\mathcal{T}\) are linear in \(|\varphi|\).

It immediately follows that any tableau \(\mathcal{T}\) for \(\varphi\) can be non-deterministically generated and explored by using a polynomial amount of space. Thus, we obtain the following corollary.
Corollary 5.2.11. The proposed tableau method for $D_{\leq}$ has a PSPACE complexity.

We conclude the section by proving soundness and completeness of the proposed tableau-based decision procedure.

Theorem 5.2.12. (Soundness) Let $\varphi$ be a $D_{\leq}$ formula and $T$ be a tableau for it. If $T$ is open, then $\varphi$ is satisfiable.

Proof. Let $T$ be an open tableau for $\varphi$. We build a fulfilling $D_{\leq}$-structure $S = \langle (V, E), \mathcal{L} \rangle$ for $\varphi$ in an incremental way, starting from the root $n_0$ of $T$. Since $T$ is open, then $n_0$ is not closed. We generate a one-node $D_{\leq}$-graph $\langle \{v_0\}, \emptyset \rangle$ and we put the formulas that belong to $n_0$ in $\mathcal{L}(v_0)$. Now, let $n$ be a non-closed node in $T$ and let $v$ be the corresponding vertex in the $D_{\leq}$-graph. We distinguish four possible cases, depending on the expansion rule that has been applied to $n$ in the tableau construction.

1. One among the NOT, AND, and REFL rules has been applied to $n$. Since $n$ is not closed, its unique successor $n'$ is not closed either. We add the formulas contained in $n'$ to $\mathcal{L}(v)$ and then we proceed by taking into consideration the node $n'$ and the corresponding vertex $v$ (notice that different nodes can be associated with the same vertex of the $D_{\leq}$-structure).

2. The OR rule has been applied to $n$. Since $n$ is not closed, it has at least one successor $n'$ that is not closed either. We add the formulas contained in $n'$ to $\mathcal{L}(v)$ and then we proceed by taking into consideration the node $n'$ and the corresponding vertex $v$.

3. The 2-DENS rule has been applied to $n$. Since $n$ is not closed, its unique successor $n'$ is not closed either. We must distinguish two cases: either $n'$ has been already taken into account during the construction of the $D_{\leq}$-structure or not. In the former case, we simply add an edge $(v, v')$ to $E$, where $v'$ is the vertex corresponding to $n'$ in the $D_{\leq}$-graph. In the latter case, we add a new reflexive vertex $v'$ to $V$, we add the edges $(v, v')$ and $(v', v')$ to $E$, and we put the formulas belonging to $n'$ in $\mathcal{L}(v')$. Then, we proceed by taking into consideration the node $n'$ and the corresponding vertex $v'$.

4. The STEP rule has been applied to $n$. Since $n$ is not closed, all its successors $n_1, \ldots, n_k$ are not closed either. For every $n_i$, we must distinguish two cases: either $n_i$ has been already taken into account during the construction of the $D_{\leq}$-structure or not. In the former case, there exists a vertex $v_i$ that corresponds to $n_i$ in the $D_{\leq}$-graph and we simply add an edge $(v, v_i)$ to $E$. In the latter case, we add an irreflexive vertex $v_i$ to $V$, we add the edge $(v, v_i)$ to $E$, and we put the formulas belonging to $n_i$ in $\mathcal{L}(v_i)$. Then, we proceed by taking into consideration the node $n_i$ and the corresponding vertex $v_i$.

Since any tableau for $\varphi$ is finite, such a construction process terminates. However, the resulting pair $\langle (V, E), \mathcal{L} \rangle$ is not necessarily a $D_{\leq}$-structure: while $(V, E)$ respects the definition of $D_{\leq}$-graph, the function $\mathcal{L}$ is not necessarily a labeling function. Since in the tableau construction we add new formulas only when necessary, there may exist a vertex $v \in V$ and a formula $\psi \in \text{CL}(\varphi)$ such that neither $\psi$ nor $\neg \psi$ belongs to $\mathcal{L}(v)$. Let $v \in V$ and $\psi \in \text{CL}(\varphi)$ such that neither $\psi$ nor $\neg \psi$ belongs to $\mathcal{L}(v)$. We can complete the definition of $\mathcal{L}$ as follows (by induction on the complexity of $\psi$):

- if $\psi = p$, with $p \in A \mathcal{P}$, we put $\neg p \in \mathcal{L}(v)$;
- if $\psi = \neg \xi$, we put $\psi \in \mathcal{L}(v)$ if and only if $\xi \not\in \mathcal{L}(v)$;
- if $\psi = \psi_1 \lor \psi_2$, we put $\psi_1 \lor \psi_2 \in \mathcal{L}(v)$ if and only if $\psi_1 \in \mathcal{L}(v)$ or $\psi_2 \in \mathcal{L}(v)$;
- if $\psi = \langle D \rangle \xi$, we put $\psi \in \mathcal{L}(v)$ if and only if there exists a descendant $v'$ of $v$ such that $\xi \in \mathcal{L}(v')$.

The resulting $D_{\leq}$-structure $\langle (V, E), \mathcal{L} \rangle$ is a fulfilling $D_{\leq}$-structure for $\varphi$. From Theorem 5.2.7, it follows that $\varphi$ is satisfiable. \qed
Theorem 5.2.13. (Completeness) Let \( \varphi \) be a \( \mathcal{D}_\mathcal{E} \) formula and \( \mathcal{T} \) be a tableau for it. If \( \mathcal{T} \) is closed, then \( \varphi \) is unsatisfiable.

Proof. We prove a stronger claim. Given a node \( n \) in a tableau, we say that (the set of formulas belonging to) \( n \) is consistent if there exists a fulfilling \( \mathcal{D}_\mathcal{E} \)-structure \( S = \langle \langle V, E \rangle, \mathcal{L} \rangle \) such that one of the following conditions holds:

- \( n \) belongs to an irreflexive macronode and there exists an irreflexive vertex \( v \in V \) such that \( \mathcal{L}(v) \) contains all formulas in \( n \);
- \( n \) belongs to a reflexive macronode and there exists a reflexive vertex \( v \in V \) such that \( \mathcal{L}(v) \) contains all formulas in \( n \).

We prove that for any node \( n \) in a tableau \( \mathcal{T} \), if \( n \) is closed, then \( n \) is inconsistent.

If there exists a formula \( \psi \in \text{CL}(\varphi) \) such that \( n \) contains both \( \psi \) and \( \neg \psi \), then \( n \) is obviously inconsistent. In the other cases, we proceed by induction, from the leaves to the root, on the expansion rule that has been applied to the node \( n \) in the construction of the tableau. Since any tableau is finite, we eventually reach the initial node of \( \mathcal{T} \), thus concluding that \( \varphi \) is an inconsistent formula.

- **The NOT rule has been applied to \( n \).** Then \( n \) is of the form \( \neg \psi, F \) and it has a unique successor \( n' = \psi, F \). If \( n' \) is closed then, by inductive hypothesis, \( \psi, F \) is an inconsistent set. Hence, \( \neg \psi, F \) is inconsistent.

- **The OR rule has been applied to \( n \).** Then \( n \) is of the form \( \psi_1 \lor \psi_2, F \) and it has two immediate successors \( n_1 = \psi_1, F \) and \( n_2 = \psi_2, F \). If both \( n_1 \) and \( n_2 \) are closed then, by inductive hypothesis, both \( \psi_1, F \) and \( \psi_2, F \) are inconsistent sets. Hence, \( \psi_1 \lor \psi_2, F \) is inconsistent.

- **The AND rule has been applied to \( n \).** Then \( n \) is of the form \( \neg(\psi_1 \lor \psi_2), F \) and it has a unique successor \( n' = \neg \psi_1, \neg \psi_2, F \). If \( n' \) is closed then, by inductive hypothesis, \( \neg \psi_1, \neg \psi_2, F \) is an inconsistent set. Hence, \( \neg \psi_1 \land \neg \psi_2, F \) is inconsistent.

- **The REFLEX rule has been applied to \( n \).** In this case, \( n \) belongs to a reflexive macronode and it is of the form \( [D]\psi, F \). Suppose, for contradiction, that \( n \) is closed, but consistent. This implies that there exists a fulfilling \( \mathcal{D}_\mathcal{E} \)-structure \( S = \langle \langle V, E \rangle, \mathcal{L} \rangle \) and a reflexive vertex \( v \in V \) such that \( n \subseteq \mathcal{L}(v) \). Since \( v \) is reflexive, we have that \( \langle v, v \rangle \in E \) and thus \( \mathcal{L}(v) \models \varphi \). Hence, we have that \( \psi \in \mathcal{L}(v) \) and thus the set of formulas \( \psi, [D] \psi, F \) is consistent. By inductive hypothesis, this implies that the successor \( n' \) of \( n \) is not closed, which contradicts the hypothesis that \( n \) is closed.

- **The 2-CONSISTENCY rule has been applied to \( n \).** In this case, \( n \) belongs to an irreflexive macronode and it is of the form \( [D]\psi_1, \ldots, [D]\psi_m, [D] \varphi_1, \ldots, [D] \varphi_h, F \). Suppose, for contradiction, that \( n \) is closed, but consistent. Hence, there exists a fulfilling \( \mathcal{D}_\mathcal{E} \)-structure \( S = \langle \langle V, E \rangle, \mathcal{L} \rangle \) and an irreflexive vertex \( v \in V \) such that \( n \subseteq \mathcal{L}(v) \). By the definition of \( \mathcal{D}_\mathcal{E} \)-structure, \( v \) has a reflexive successor \( v' \) such that \( \mathcal{L}(v) \models \varphi \). Hence, \( v' \) is such that \( \psi_1, \ldots, \psi_m, [D]\psi_1, \ldots, [D]\psi_m, [D] \varphi_1, \ldots, [D] \varphi_h \) \( \subseteq \mathcal{L}(v') \). This proves that the successor \( n' \) of \( n \) in the tableau is consistent. By inductive hypothesis, \( n' \) is not closed, which contradicts the hypothesis that \( n \) is closed.

- **The STEP rule has been applied to \( n \).** In this case, \( n \) belongs to a reflexive macronode and it is of the form \( [D] \varphi_1, \ldots, [D] \psi_m, [D] \varphi_1, \ldots, [D] \varphi_h, F \). Suppose, for contradiction, that \( n \) is closed, but consistent. Hence, there exists a fulfilling \( \mathcal{D}_\mathcal{E} \)-structure \( S = \langle \langle V, E \rangle, \mathcal{L} \rangle \) and a reflexive vertex \( v \in V \) such that \( n \subseteq \mathcal{L}(v) \). Since \( S \) is fulfilling, for every formula \( \langle D \rangle \varphi_i \), there exists a descendant \( v_i \) of \( v \) such that \( \varphi_i \in \mathcal{L}(v_i) \). This implies that, for every \( i = 1, \ldots, h \), the set of formulas \( G_i = \{ \varphi_i, \psi_1, \ldots, \psi_m, [D] \varphi_1, \ldots, [D] \psi_m \} \) is consistent. By inductive hypothesis, this implies that every immediate successor of \( n \) is not closed, which contradicts the hypothesis that \( n \) is closed.

\( \square \)
5.3 The logic $D_{\prec}$ of proper subinterval structures

5.3.1 Structures for $D_{\prec}$ formulas

In this section we extend the notion of $D_{\prec}$-graph and $D_{\prec}$-structures given in Section 5.2.1 to the case of the logic $D_{\prec}$.

The presence of the special families of beginning subintervals and ending subintervals complicates the construction of pseudo-models for $D_{\prec}$. Given an interval $[b, e]$, a beginning subinterval of $[b, e]$ is an interval $[b, e']$, with $e' < e$, an ending subinterval of $[b, e]$ is an interval $[b', e]$, with $b < b'$, and an internal subinterval of $[b, e]$ is an interval $[b', e']$, with $b < b'$ and $e' < e$. To represent infinite chains of beginning (resp., ending) subintervals of a given interval, we need to introduce the notion of cluster of reflexive nodes. Given a graph $G = (V, E)$, we define a cluster as a maximally strongly connected subgraph $C$ which includes reflexive vertices only. By abuse of notation, we say that a cluster $C$ is a successor of a vertex $v$ if $v$ does not belong to $C$ and there exists a successor $v'$ of $v$ in $C$. Conversely, a vertex $v$ is a successor of $C$ if $v$ does not belong to $C$ and there exists a predecessor $v'$ of $v$ in $C$. The definition of $D_{\prec}$-graph can be extended to $D_{\prec}$ as follows.

**Definition 5.3.1.** A finite directed graph $G = (V, E)$ is a $D_{\prec}$-graph if:

1. there exists an irreflexive vertex $v_0 \in V$, called the root of $G$, such that any other vertex $v \in V$ is reachable from it;
2. every irreflexive vertex $v \in V$ has exactly two clusters as successors: a beginning successor cluster $C_b$ and an ending successor cluster $C_e$;
3. $C_b$ and $C_e$ have a unique common successor $v_c$, which is a reflexive vertex;
4. every successor of $v_c$, different from $v_c$ itself, is irreflexive;
5. there exists at most one edge exiting the clusters $C_b$ and $C_e$ and reaching an irreflexive node;
6. apart from the edge leading to $v_c$, there are no edges exiting from $C_b$ (resp. $C_e$) that reach a reflexive vertex.

Figure 5.5 depicts a portion of a $D_{\prec}$-graph. The root $v_0$ has two successor clusters $C_b$ and $C_e$ of four vertices each. Both $C_b$ and $C_e$ have exactly one irreflexive successor. Their common reflexive successor $v_c$ has two irreflexive successors.
Let \( \varphi \) be a \( D_{\mathcal{C}} \) formula. \( D_{\mathcal{C}} \)-structures are defined by pairing a \( D_{\mathcal{C}} \)-graph with a labeling function that associates an \( A_{\varphi} \) atom with each vertex of the graph.

**Definition 5.3.2.** A \( D_{\mathcal{C}} \)-structure is a quadruple \( S = \langle \mathcal{V}, \mathcal{E}, \mathcal{B}, \mathcal{E} \rangle \), where:

1. \((\mathcal{V}, \mathcal{E})\) is a \( D_{\mathcal{C}} \)-graph;
2. \( \mathcal{L} : \mathcal{V} \rightarrow A_{\varphi} \) is a labeling function that assigns to every vertex \( v \in \mathcal{V} \) an atom \( \mathcal{L}(v) \) such that for every edge \((v, v') \in \mathcal{E}, \mathcal{L}(v) \cup \mathcal{L}(v') \);
3. \( \mathcal{B} : \mathcal{V} \rightarrow 2^{REQ} \) and \( \mathcal{E} : \mathcal{V} \rightarrow 2^{REQ} \) are mappings that assign to every vertex the sets of its beginning and ending requests, respectively;
4. for every irreflexive vertex \( v \in \mathcal{V} \), with successor clusters \( \mathcal{C}_b \) and \( \mathcal{C}_e \), we have that:
   - \( v_c \), the common reflexive successor of \( \mathcal{C}_b \) and \( \mathcal{C}_e \), is such that \( \mathcal{L}(v_c) = \mathcal{B}(v_c) = \emptyset \) and \( \text{REQ}(\mathcal{L}(v_c)) = \mathcal{E}(v_c) = \emptyset \), and \( \text{REQ}(\mathcal{L}(v_c)) \cup B(v) \),
   - every reflexive vertex \( v' \in \mathcal{C}_b \) is such that \( \mathcal{B}(v') = \mathcal{B}(v), \mathcal{E}(v') = \emptyset \), and \( \text{REQ}(\mathcal{L}(v')) = \text{REQ}(\mathcal{L}(v_c)) \cup \mathcal{B}(v) \),
   - the unique irreflexive successor \( v'' \) of \( \mathcal{C}_b \) (if any) is such that \( \mathcal{B}(v) \cap \mathcal{L}(v'') \subseteq \mathcal{B}(v'') \) (requests which have been classified as initial in a given vertex cannot be reclassified in its descendants),
   - every reflexive vertex \( v' \in \mathcal{C}_e \) is such that \( \mathcal{E}(v') = \emptyset(v), \mathcal{B}(v') = \emptyset \), and \( \text{REQ}(\mathcal{L}(v')) = \text{REQ}(\mathcal{L}(v_c)) \cup \mathcal{E}(v) \),
   - the unique irreflexive successor \( v'' \) of \( \mathcal{C}_e \) (if any) is such that \( \mathcal{E}(v) \cap \mathcal{L}(v'') \subseteq \mathcal{E}(v'') \) (requests which have been classified as ending in a given vertex cannot be reclassified in its descendants).

Let \( v_0 \) be the root of \((\mathcal{V}, \mathcal{E})\). If \( \varphi \in \mathcal{L}(v_0) \), we say that \( S \) is a \( D_{\mathcal{C}} \)-structure for \( \varphi \).

Beginning and ending requests associated with a vertex \( v \) can be viewed as requests that must be satisfied over respectively beginning and ending subintervals of any interval corresponding to \( v \) (possibly over both of them), but not over its internal subintervals.

Every \( D_{\mathcal{C}} \)-structure can be regarded as a Kripke model for \( D_{\mathcal{C}} \), where the valuation is determined by the labeling. As in the case of \( D_{\mathcal{C}} \)-structures, we restrict our attention to fulfilling structures.

**Definition 5.3.3.** A \( D_{\mathcal{C}} \)-structure \( S = \langle \mathcal{V}, \mathcal{E}, \mathcal{B}, \mathcal{E} \rangle \) is fulfilling if for every \( v \in \mathcal{V} \) and every \( (\mathcal{D})\psi \in \mathcal{L}(v) \), there exists \( v' \in \mathcal{V} \) such that \( v' \) is a descendant of \( v \) and \( \psi \in \mathcal{L}(v') \).

**Theorem 5.3.4.** Let \( \varphi \) be a \( D_{\mathcal{C}} \) formula which is satisfied in a proper interval model. Then, there exists a fulfilling \( D_{\mathcal{C}} \)-structure \( S = \langle \mathcal{V}, \mathcal{E}, \mathcal{B}, \mathcal{E} \rangle \) for \( \varphi \).

**Proof.** Let \( M = \langle [I(D), \subset, \mathcal{V}] \rangle \) be a proper interval model and let \( [b_0, c_0] \in I(D) \) be an interval such that \( M, [b_0, c_0] \models \varphi \). We recursively build a fulfilling \( D_{\mathcal{C}} \)-structure \( S = \langle \mathcal{V}, \mathcal{E}, \mathcal{B}, \mathcal{E} \rangle \) for \( \varphi \) as follows.

We start with the one-node graph \( \langle [v_0], \emptyset \rangle \) and a labeling function \( \mathcal{L} \) such that \( \mathcal{L}(v_0) = \{ \psi \in \text{CL}(\varphi) : M, [b_0, c_0] \models \psi \} \). Then, we partition the set \( \text{REQ}(\mathcal{L}(v_0)) \) into the following three sets of formulas:

**Beginning requests:** \( B_{v_0} \) contains all \( (\mathcal{D})\xi \in \text{REQ}(\mathcal{L}(v_0)) \) such that \( \xi \) is satisfied over beginning subintervals of \([b_0, c_0]\), but not over internal subintervals of \([b_0, c_0]\);

**Ending requests:** \( E_{v_0} \) contains all \( (\mathcal{D})\xi \in \text{REQ}(\mathcal{L}(v_0)) \) such that \( \xi \) is satisfied over ending subintervals of \([b_0, c_0]\), but not over internal subintervals of \([b_0, c_0]\);

**Internal requests:** \( I_{v_0} = (\text{REQ}(\mathcal{L}(v_0)) \setminus B_{v_0}) \setminus E_{v_0} \), that is, the set of all \( (\mathcal{D})\xi \in \text{REQ}(\mathcal{L}(v_0)) \) such that \( \xi \) is satisfied over internal subintervals of \([b_0, c_0]\).
We put $B(v_0) = B_{v_0}$ and $E(v_0) = E_{v_0}$. Then, for every formula $(D)\psi \in \mathcal{L}(v_0)$, we choose an interval $[b_\psi, e_\psi]$, with $[b_\psi, e_\psi] \subset [b_0, e_0]$, such that $M, [b_\psi, e_\psi] \models \psi$. If $(D)\psi \in I_{v_0}$, then $b_0 < b_\psi < e_\psi < e_0$, else if $(D)\psi \in B_{v_0}$, then $b_0 = b_\psi < e_\psi < e_0$, else $(D)\psi \in E_{v_0}$ $b_0 < b_\psi < e_\psi = e_0$.

Since $\mathbb{D}$ is a dense ordering and $\mathcal{CL}(\varphi)$ is a finite set of formulas, there exist two beginning intervals $[b_0, e_1]$ and $[b_0, e_2]$ such that:

- for every interval $[b_\psi, e_\psi]$, with $(D)\psi \in B_{v_0} \cup I_{v_0}$, $[b_\psi, e_\psi] \subset [b_0, e_2] \subset [b_0, e_1]$;
- $[b_0, e_1]$ and $[b_0, e_2]$ satisfy the same formulas of $\mathcal{CL}(\varphi)$.

We start the construction of the beginning successor cluster $\mathcal{C}_b$ of $v_0$ by adding a new vertex $v_b$ and a pair of edges $(v_b, v_0)$ and $(v_b, v_b)$, and by putting $\mathcal{L}(v_b) = \{ \xi \in \mathcal{CL}(\varphi) : M, [b_0, e_1] \models \xi \}$, $B(v_b) = B_{v_0}$ and $E(v_b) = \emptyset$. Next, for every $(D)\psi \in B(v_b)$ we establish whether or not we must add a vertex $v_\psi$ in $\mathcal{C}_b$ as follows. Let $[b_\psi, e_\psi]$ be a beginning subinterval such that $M, [b_0, e_\psi] \models \psi$. We add a reflexive vertex $v_\psi$ to $\mathcal{C}_b$ if $[b_0, e_\psi]$ satisfies the same temporal formulas $[b_0, e_1]$ satisfies. Moreover, we put $\mathcal{L}(v_\psi) = \{ \xi \in \mathcal{CL}(\varphi) : M, [b_0, e_\psi] \models \xi \}$, $B(v_\psi) = B(v_b)$, and $E(v_\psi) = \emptyset$. Let $\{v_1, \ldots, v_k\}$ be the resulting set of vertices added to $\mathcal{C}_b$. For $i = 1, \ldots, k - 1$, we add an edge $(v_i, v_{i+1})$ to $E$; furthermore, we add the edges $(v_b, v_1)$ and $(v_b, v_\psi)$ to $E$. If for all formulas $(D)\psi \in B(v_b)$ there exists a corresponding vertex $v_\psi$ in $\mathcal{C}_b$, we are done. Otherwise, let $\Gamma_b$ be the set of the remaining formulas $(D)\psi \in B(v_b)$ and let $[b_0, e_\max] \subset B(v_b)$ be a beginning subinterval such that, for every formula $(D)\psi \in \Gamma_b$, we have that $M, [b_0, e_\max] \models \psi$ or $M, [b_0, e_\max] \models (D)\psi$. We add a new reflexive vertex $v_\max$ and an edge connecting an arbitrary vertex in $\mathcal{C}_b$ to it, say $v_\psi$, and we define its labeling as $\mathcal{L}(v_\max) = \{ \xi \in \mathcal{CL}(\varphi) : M, [b_0, e_\max] \models \xi \}$.

The ending successor cluster $\mathcal{C}_e$ of $v_0$ is built in the very same way.

To complete the first phase of the construction, we must introduce the common reflexive successor $v_c$ of $\mathcal{C}_b$ and $\mathcal{C}_e$. Since $\mathbb{D}$ is a dense ordering and $\mathcal{CL}(\varphi)$ is a finite set of formulas, there exist two intervals $[b_3, e_3]$ and $[b_4, e_4]$ such that:

- for every interval $[b_\psi, e_\psi]$, with $(D)\psi \in I_{v_0}$, $[b_\psi, e_\psi] \subset [b_4, e_4] \subset [b_3, e_3]$;
- $[b_3, e_3]$ and $[b_4, e_4]$ satisfy the same formulas of $\mathcal{CL}(\varphi)$.

We add a new vertex $v_c$, together with the edges $(v_b, v_c)$, $(v_e, v_c)$, and $(v_c, v_e)$, and we put $\mathcal{L}(v_c) = \{ \xi \in \mathcal{CL}(\varphi) : M, [b_3, e_3] \models \xi \}$, $B(v_c) = E(v_c) = \emptyset$.

For every formula $(D)\psi \in I_{v_0}$, we add a new vertex $v_\psi$ and an edge $(v_c, v_\psi)$, and we define its labeling as $\mathcal{L}(v_\psi) = \{ \xi \in \mathcal{CL}(\varphi) : M, [b_0, e_\max] \models \xi \}$.

Then, we recursively apply the above procedure to the irrereflexive vertices we have introduced. To keep the construction finite, whenever there exists an irrereflexive vertex $v' \in V$ such that $\mathcal{L}(v_\psi) = \mathcal{L}(v')$ for some $v_\psi$, we simply add an edge to $v'$ instead of creating a new vertex $v_\psi$ and an edge entering it. Since the set of atoms is finite, the construction is guaranteed to terminate.

Let $S$ be a fulfilling $D_\mathbb{D}$-structure for a formula $\varphi$. To build a model for $\varphi$, we consider the interval $[0, 1]$ of the rational line and define a function $\mathcal{G}$ mapping intervals in $I([0, 1])$ to vertices in $S$.

**Definition 5.3.5.** Let $S = \langle V, E \rangle, \mathcal{L}, \mathbb{B}, \mathcal{E} \rangle$ be a $D_\mathbb{D}$-structure. The function $\mathcal{G} : I([0, 1]) \mapsto V$ is defined recursively as follows. First, $\mathcal{G}([0, 1]) = v_0$. Now, let $[b, c]$ be an interval such that $\mathcal{G}([b, c]) = v$ and $\mathcal{G}$ has not been yet defined over any of its subintervals. We distinguish two cases.

**Case 1:** $v$ is an irrereflexive vertex. Let $\mathcal{C}_b$ and $\mathcal{C}_e$ be the reflexive successor beginning and ending clusters of $v$, respectively, and $v_c$ be their common reflexive successor. Let $v_\max$ be the irrereflexive successor of $\mathcal{C}_b$ (if any), $v_\max$ be the irrereflexive successor of $\mathcal{C}_e$ (if any), and $v_1, \ldots, v_k$ be the $k$ irrereflexive successors of $v_c$ (if any). Let $p = \frac{c - b}{2^k + 3}$. The function $\mathcal{G}$ is defined as follows (the definition is very close to that for $D_\mathbb{D}$-structures given in Section 5.2):

1. we put $\mathcal{G}([b, b + p]) = v_\max$ and $\mathcal{G}([c - p, c]) = v_\max$;
2. for every $i = 1, \ldots, k$, we put $\mathcal{G}([b + 2ip, b + (2i + 1)p]) = v_i$;
3. for every $i = 1, \ldots, k + 1$, we put $S([b + (2i - 1)p, b + 2ip]) = v_c$;
4. for every strict subinterval $[b', e']$ of $[b, e]$ which is not a subinterval of any of the intervals $[b + ip, b + (i + 1)p]$, we put $S([b', e']) = v_c$.

To complete the construction, we need to define $S$ over the beginning subintervals $[b, e']$ such that $b + p < e' < e$ and the ending subintervals $[b', e]$ such that $b < b' < e - p$. We map such beginning (resp., ending) subintervals to vertices in $C_b$ (resp., $C_e$) in such a way that for any beginning subinterval $[b, e']$ (resp., ending subinterval $[b', e]$) and any $v_b \in C_b$ (resp., $v_e \in C_e$), there exists a beginning subinterval $[b, e'']$ with $[b, b + p] \sqsubset [b, e''] \sqsubset [b, e']$ (resp., ending subinterval $[b'', e']$, with $[e - p, e] \sqsubset [b'', e] \sqsubset [b', e]$) such that $S([b, e'']) = v_b$ (resp., $S([b'', e]) = v_e$).

**Case 2:** $v$ is a reflexive vertex. The case in which $v$ belongs to $C_b$ or $C_e$ has been already dealt with. Thus, we only need to consider the case of vertices $v_c$ with irreflexive successors only (apart from themselves). We distinguish two cases:

1. $v_c$ has no successors apart from itself. In such a case, we put $S([b', e']) = v_c$ for every subinterval $[b', e']$ of $[b, e]$.
2. $v_c$ has at least one successor different from itself. Let $v^1_c, \ldots, v^k_c$ be the $k$ successors of $v_c$ different from $v_c$. We consider the intervals defined by the points $b, b + p, b + 2p, \ldots, b + 2kp, b + (2k + 1)p = e$, with $p = \frac{e - b}{2k + 1}$. The function $S$ over such intervals is defined as follows:
   - for every $i = 1, \ldots, k$, we put $S([b + (2i - 1)p, b + 2ip]) = v^i_c$.
   - for every $i = 0, \ldots, k$, we put $S([b + 2ip, b + (2i + 1)p]) = v_c$.

We complete the construction by putting $S([b', e']) = v_c$ for every subinterval $[b', e']$ of $[b, e]$ which is not a subinterval of any of the intervals $[b + ip, b + (i + 1)p]$.

As in the case of $D_{\mathbb{Z}}$-structures, the function $S$ satisfies some basic properties.

**Lemma 5.3.6.**
1. For every interval $[b, e] \in \mathbb{I}([0, 1])$, if $S([b, e]) = v$ and $v'$ is reachable from $v$, then there exists an interval $[b', e']$ such that $S([b', e']) = v'$ and $[b', e'] \sqsubset [b, e]$.
2. For every pair of intervals $[b, e]$ and $[b', e']$ in $\mathbb{I}([0, 1])$ such that $[b', e'] \sqsubset [b, e]$, we have that for every formula $\square \psi \in \mathcal{L}(S([b, e]))$, both $\psi$ and $\square \psi$ belong to $\mathcal{L}(S([b', e']))$.

**Proof.** Condition 1 can be easily proved by observing that it trivially holds for all successors of $v$ by definition of $S$ and then extending the result to every descendant $v'$ of $v$ by induction on the length of the shortest path from $v$ to $v'$.

As for condition 2, let $[b, e]$ and $[b', e']$ be two intervals in $\mathbb{I}([0, 1])$ such that $[b', e'] \sqsubset [b, e]$, $v = S([b, e])$, and $v' = S([b', e'])$. If $v'$ is a descendant of $v$ in the $D_{\mathbb{Z}}$-graph, then condition 2 holds by definition of $D_{\mathbb{Z}}$. When we apply the construction step defined by Case 1, Point 4, of Definition 5.3.5, it may happen that $[b', e'] \sqsubset [b, e]$ but $v'$ is not reachable from $v$ in the $D_{\mathbb{Z}}$-graph. In such a case, both $[b, e]$ and $[b', e']$ are internal subintervals, and thus, by definition of the labeling functions $\mathbb{B}$ and $\mathbb{E}$, Condition 2 is satisfied.

**Theorem 5.3.7.** Given any fulfilling $D_{\mathbb{Z}}$-structure $S$ for $\varphi$, there exists an interval model $M_{\mathbb{S}} = (\mathbb{I}([0, 1]), \sqsubset, \mathbb{V})$ over the rational interval $[0, 1]$ such that $M_{\mathbb{S}}, [0, 1] \models \varphi$.

**Proof.** For every $p \in \mathbb{A}$, let $\mathbb{V}(p) = \{[b, e] : p \in \mathcal{L}(S([b, e]))\}$. We can prove by induction on the structure of formulas $\psi \in \mathcal{CL}(\varphi)$ that for every interval $[b, e] \in \mathbb{I}([0, 1])$:

$$M_{\mathbb{S}}, [b, e] \models \psi \iff \psi \in \mathcal{L}(S([b, e])).$$

The atomic case immediately follows from definition of $\mathbb{V}$. The Boolean cases follow from the definition of atom. Finally, the case of temporal formulas follows from Lemma 5.3.6. This allows us to conclude that $M_{\mathbb{S}}, [0, 1] \models \varphi$. 

\footnote{Notice that the density of the rational interval $[0, 1]$ plays here an essential role.}
5.3.2 A small-model theorem for $D_\infty$-structures

Given a fulfilling $D_\infty$-structure, we can remove from it those vertices which are not necessary to fulfill any $\langle D \rangle$ formula to obtain a smaller $D_\infty$-structure of bounded size, as proved by the following theorem.

**Theorem 5.3.8.** For every satisfiable $D_\infty$ formula $\varphi$, there exists a fulfilling $D_\infty$-structure with breadth and depth bounded by $2 \cdot |\varphi|$.

**Proof.** Consider a fulfilling $D_\infty$-structure $S$. The size of the structure can be safely reduced as follows:

- we remove from every cluster $\mathcal{C}$ all vertices that either do not fulfill any $\langle D \rangle$ formula or fulfill only formulas that are fulfilled by some descendant of it. Let $\mathcal{C}'$ be the resulting cluster. We select a minimal subset $\mathcal{C}' \subseteq \mathcal{C}$ that fulfills all formulas that are fulfilled only inside $\mathcal{C}$ and we replace $\mathcal{C}$ with $\mathcal{C}'$ (if $\mathcal{C}'$ is empty, we replace $\mathcal{C}$ with one of its vertices);
- for every common reflexive successor $v_c$ of a pair of clusters, we select a minimal subset of its irreflexive successors whose vertices satisfy all $\langle D \rangle$ formulas in $v_c$.

The execution of the first removal process produces a $D_\infty$-structure where the size of every cluster is at most $|\varphi|$ and every vertex in a cluster of size at least 2 fulfills some $\Psi$ formulas which are not fulfilled elsewhere, while the execution of the second removal process produces a $D_\infty$-structure where every vertex has at most $|\varphi|$ immediate successors. Since whenever we exit from a cluster or we move from a reflexive node to an irreflexive one the number of requests strictly decreases, we can conclude that the length of every loop-free path is at most $2 \cdot |\varphi|$.

As a direct consequence of Theorem 5.3.8, we have that a fulfilling $D_\infty$-structure for a formula $\varphi$ (if any) can be generated and explored by a non-deterministic procedure that uses only a polynomial amount of space. This gives the following complexity bound to the decision problem for $D_\infty$.

**Theorem 5.3.9.** The decision problem for $D_\infty$ is in PSPACE.

**Proof.** A PSPACE non-deterministic algorithm to check the satisfiability of a $D_\infty$ formula $\varphi$ can be obtained as follows. We non-deterministically generate an atom containing $\varphi$, we associate it with the root $v$ of a $D_\infty$-structure, and we guess the subsets $\mathcal{B}(v)$ and $\mathcal{E}(v)$ of $\text{REQ}(\mathcal{L}(v))$. Then, we apply the following recursive procedure on $v$:

- If the vertex $v$ is irreflexive, we generate three new vertices $v_b, v_c$ and $v_e$. Then, we non-deterministically generate three labelings for them such that (if such labelings do no exist, the procedure fails):
  (i) $v_b, v_c$, and $v_e$ are successors of $v$, that is, $\mathcal{L}(v) D_\varphi \mathcal{L}(v_c)$ for every $x \in \{b, c, e\}$;
  (ii) $v_b, v_c$, and $v_e$ are reflexive vertices, that is, $\mathcal{L}(v_c) D_\varphi \mathcal{L}(v_x)$ for every $x \in \{b, c, e\}$;
  (iii) $v_e$ is a successor of both $v_b$ and $v_e$, that is, $\mathcal{L}(v_b) D_\varphi \mathcal{L}(v_e)$ and $\mathcal{L}(v_e) D_\varphi \mathcal{L}(v_c)$;
  (iv) $\mathcal{B}(v) \subseteq \mathcal{L}(v_b)$, $\mathcal{E}(v) \subseteq \mathcal{L}(v_c)$, and $\{[D] \dashv \vdash \psi \mid \langle D \rangle \psi \in \mathcal{B}(v) \cup \mathcal{E}(v) \} \subseteq \mathcal{L}(v_c)$.

The vertex $v_b$ is the first vertex of the beginning cluster $\mathcal{C}_b$ associated with $v$, the vertex $v_e$ is the first vertex of the ending cluster $\mathcal{C}_e$, and the vertex $v_c$ is the common reflexive successor of $\mathcal{C}_b$ and $\mathcal{C}_c$. We recursively call the procedure separately on $v_b$, $v_c$, and $v_e$. If one of these three calls returns fail, the procedure ends with failure; otherwise, it returns success.

- If the vertex $v$ is the common reflexive successor of a pair of clusters, we generate an irreflexive successor $v_\emptyset$ of it for every $\langle D \rangle \psi \in \text{REQ}(\mathcal{L}(v))$ and we guess a labeling $\mathcal{L}(v_\emptyset)$ such $\psi \in \mathcal{L}(v_\emptyset)$ and $\mathcal{L}(v) D_\varphi \mathcal{L}(v_\emptyset)$ (if no such a labeling exists, the procedure fails). If $\text{REQ}(\mathcal{L}(v_\emptyset)) \neq \mathcal{L}(v)$, we recursively call the procedure on $v_\emptyset$; otherwise, we add an edge from $v_\emptyset$ to $v$ in the $D_\infty$-structure. If one of these calls fails, then we return fail; otherwise, we return success.
5.3. The logic $D_\prec$ of proper subinterval structures

- If the node $v$ belongs to a beginning cluster $\mathcal{C}_B$ (the ending case is symmetric), we guess a formula $\langle D \rangle \psi \in \mathcal{B}(v)$ such that $\langle D \rangle \psi \in \mathcal{L}(v)$ and no vertex in $\mathcal{C}_B$ satisfies $\psi$ (if any). If such a formula does not exist, the procedure ends with success; otherwise, the procedure non-deterministically chooses to satisfy $\langle D \rangle \psi$ either in $\mathcal{C}_B$ or outside it. In the former case, it adds a new reflexive node $v'$ to $\mathcal{C}_B$ and it guesses a labeling $\mathcal{L}(v')$ such that $\psi \in \mathcal{L}(v')$, $\mathcal{L}(v') \cup \mathcal{D} \phi \cup \mathcal{L}(v')$, and $\text{REQ}(\mathcal{L}(v')) = \text{REQ}(\mathcal{L}(v))$ (if no such a labeling exists, the procedure fails). In the latter case, it adds an irreflexive successor $v'$ to $\mathcal{C}_B$ and it guesses a labeling $\mathcal{L}(v')$ such that $\psi \in \mathcal{L}(v')$, $\mathcal{L}(v') \cup \mathcal{D} \phi \cup \mathcal{L}(v')$ and $\text{REQ}(\mathcal{L}(v')) \subset \text{REQ}(\mathcal{L}(v))$ (if no such a labeling exists, the procedure fails). Finally, we recursively call the procedure on $v'$.

As for the computational complexity, every call to the procedure needs to store only the path from the root to the current vertex, and it generates at most $|\varphi|$ distinct recursive calls. Moreover, every recursive call either satisfies a $\langle D \rangle \psi$ formula (and thus it strictly decreases the number of remaining $\langle D \rangle$ formulas) or it generates a beginning, an ending, or a ‘middle’ reflexive vertex. Since every step at which a new beginning, ending, or middle reflexive vertex is generated is followed by a step at which a vertex that satisfies a $\langle D \rangle$ formula is generated, the maximum number of nested calls to the procedure is bounded by $2 \cdot |\varphi|$. This allows us to conclude that the procedure is of PSPACE complexity.

The very same reduction that has been used to prove $D_\prec$ PSPACE hardness can be applied to $D_\prec$, thus proving the PSPACE completeness of the satisfiability problem for $D_\prec$.

### 5.3.3 The tableau method for $D_\prec$

In this section we present a tableau system for $D_\prec$. From the model-theoretic results in the previous section, we have that a $D_\prec$ formula $\varphi$ is satisfiable if and only if there exists a fulfilling $D_\prec$-structure for it. The tableau method attempts systematically to build such a structure if there is any, returning “satisfiable” if it succeeds and “unsatisfiable” otherwise.

The nodes of the tableaux are sets of locally consistent formulas (i.e., parts of atoms). At the root of the tableau, we place a set containing only the formula $\varphi$ the satisfiability of which is being tested. We then proceed recursively to expand the tableau, following the expansion rules described below. Every disjunctive branch of the tableau describes an attempt to construct a fulfilling $D_\prec$-structure for the atom at the root. Going down the branch roughly corresponds to going deeper into subintervals of the interval corresponding to the root. The applicability of an expansion rule at a given node depends on the formulas in the node and on the parts of $D_\prec$-structure we are building. The expansion of the tableau proceeds as follows.

1. We start with the current vertex (at the beginning, the root) $v_0$ of the $D_\prec$-structure that is being constructed and we apply the usual Boolean rules to decompose Boolean operators.

2. Then, we impose a suitable marking on $\langle D \rangle$ formulas to partition them into four sets: the set of formulas that are satisfied only on beginning subintervals, that of formulas that are satisfied only on ending subintervals, that of formulas that are satisfied both on beginning and ending subintervals, and that of formulas that are satisfied on internal subintervals.

3. The third phase of the procedure is the construction of the first vertex $v_b$, the first vertex $v_e$ of the ending successor cluster $\mathcal{C}_e$, and their common successor $v_c$.

4. Next, we proceed in parallel with the construction of the clusters $\mathcal{C}_b$ and $\mathcal{C}_e$ by guessing the $\langle D \rangle$ formulas from the set $\text{REQ}(\mathcal{L}(v_0))$ that should be satisfied inside each of them.

5. Then, we build the irreflexive successor $v_b^{\text{irr max}}$ of $\mathcal{C}_b$, the irreflexive successor $v_e^{\text{irr max}}$ of $\mathcal{C}_e$, and the irreflexive successors of $v_c$, if needed, and proceed recursively with their expansion from Step 1 above.
During the expansion of the tableau, we restrict our search to models with the property stated in Theorem 5.3.8. In particular, during the construction of a cluster we explicitly satisfy only those \( \langle D \rangle \) formulas that should be satisfied inside the cluster and can never be satisfied outside it. In this way we have the following advantages:

\( i \) we consider a \( \langle D \rangle \) formula only once on a given branch of the tableau.

\( ii \) when we exit a cluster, we can add the negation of every \( \langle D \rangle \) formula that has been explicitly satisfied inside that cluster, thus reducing the search space of the successive expansion steps.

The rules of the tableau.

Before describing the tableau rules in details, we need to introduce some preliminary notation. A formula of the form \( \langle D \rangle \psi \in \text{CL}(\varphi) \) can be possibly marked as follows:

\[
\langle D \rangle^M \psi, \langle D \rangle^B \psi, \langle D \rangle^{BC} \psi, \langle D \rangle^{BNC} \psi, \langle D \rangle^E \psi, \langle D \rangle^{EC} \psi, \langle D \rangle^{ENC} \psi, \langle D \rangle^{BE} \psi.
\]

This notation has the following intuitive meaning. The markings \( \langle D \rangle^M \psi \), \( \langle D \rangle^B \psi \), \( \langle D \rangle^E \psi \), and \( \langle D \rangle^{BE} \) appear when we try to construct an irreflexive interval node and we guess that the formula \( \langle D \rangle \psi \) should be satisfied over an internal (middle) subinterval, only over a beginning subinterval, only over an ending subinterval, or both over a beginning and over an ending (but not over middle) subinterval of the current one. The markings \( \langle D \rangle^{BC} \psi \) or \( \langle D \rangle^{BNC} \psi \) (resp. \( \langle D \rangle^{EC} \psi, \langle D \rangle^{ENC} \psi \)) substitute a previously marked \( \langle D \rangle^B \psi \) (resp. \( \langle D \rangle^E \psi \)) formula when we try to construct a beginning cluster and we guess that the formula \( \psi \) should be satisfied in the current cluster ((\( D \rangle^B \psi \) marking) or not ((\( D \rangle^{BNC} \psi \) marking). The marking is only used for bookkeeping purposes, to facilitate the correct choice of the rules to be applied. It does not affect the existence of a contradiction; we say that a node is closed if once we remove the marking from every formula in it, it then contains both \( \psi \) and \( \neg \psi \) for some \( \psi \in \text{CL}(\varphi) \).

Given a set \( \Phi \) of possibly marked formulas, the set \( \text{TF}(\Phi) \) (the temporal fragment of \( \Phi \)) is the set of all the formulas in \( \Phi \) of the types \( \langle D \rangle \psi \) and \( \lnot \psi \) (ignoring the markings). Given a set of formulas \( \Gamma \), we use \( \langle D \rangle \Gamma \), where \( \langle D \rangle \in \{\langle D \rangle, \langle D \rangle^M, \langle D \rangle^B, \langle D \rangle^{BC}, \langle D \rangle^{BNC}, \langle D \rangle^{E}, \langle D \rangle^{EC}, \langle D \rangle^{ENC}, \langle D \rangle^{BE}\} \), as a shorthand for \( \{\langle D \rangle \psi \mid \psi \in \Gamma\} \). Likewise, \( \lnot \Gamma \) stands for \( \{\lnot \psi \mid \psi \in \Gamma\} \) and \( \Gamma \lor \langle D \rangle \Gamma \) for \( \{\psi \lor \langle D \rangle \psi \mid \psi \in \Gamma\} \).

We now describe the rules used to expand the tableau nodes. In order to help the reader in understanding them, they are introduced and briefly explained in the order they appear in the procedure. We start with an initial tableau consisting of only one node containing the formula \( \varphi \) that we want to check for satisfiability. We apply the following Boolean Rules to \( \varphi \) and to the newly generated nodes until these rules are no longer applicable:

\[
\begin{align*}
\frac{\Phi, \lnot \psi}{\Phi, \psi} & \quad \frac{\Phi, \psi_1 \lor \psi_2}{\Phi, \psi_1} \quad \frac{\Phi, \psi_1 \lor \psi_2}{\Phi, \psi_1} \\
\frac{\Phi, \psi_1 \lor \psi_2}{\Phi, \lnot \psi_1, \lnot \psi_2} & \quad \frac{\Phi, \lnot \psi_1 \lor \lnot \psi_2}{\Phi, \lnot \psi_1 \lor \lnot \psi_2}
\end{align*}
\]

Next, we focus on a node to which the Boolean Rules are no more applicable. At this stage the node contains only atomic formulas and a subset of the temporal fragment of an atom (there may exist a formula \( \langle D \rangle \psi \in \text{REQ}(\varphi) \) for which neither \( \langle D \rangle \psi \) nor \( \lnot \langle D \rangle \psi \) belongs to the current node). In order to obtain a complete temporal fragment, we apply the following Completion Rule to the current node and to all newly generated nodes:

\[
\frac{\Phi}{\Phi, \langle D \rangle \psi} \quad \text{where } \langle D \rangle \psi \in \text{CL}(\varphi), \langle D \rangle \psi \notin \Phi, \text{ and } \lnot \langle D \rangle \psi \notin \Phi.
\]

Given a node with a complete temporal fragment, we have to classify every formula of the form \( \langle D \rangle \psi \) belonging to it as a beginning, middle, ending, or both beginning and ending one. This is done by the following Marking Rule:
\[
\begin{array}{c}
\Phi, (D)^B \psi \\
\Phi, (D)^E \psi \mid \Phi, (D)^M \psi \mid \Phi, (D)^{BC} \psi \mid \Phi, (D)^{BNC} \psi
\end{array}
\]

where neither \( (D)^B \psi \) nor \( (D)^E \psi \) belongs to an ancestor of the current node.

The conditions for the application of this rule will be explained later.

Given an irreflexive node with a complete temporal fragment, whose \( (D) \) formulas have been classified and marked, we generate its two reflexive successors, together with their common reflexive successor. This operation is performed by applying once the following **Reflexive Step Rule**:

\[
\begin{array}{c}
\Phi, (D)^B \Gamma, (D)^M \Lambda, (D)^{BC} \Theta, (D)^{BNC} \Gamma, (D)^{M} M, [D] \Delta \\
(D)^{B} \Gamma, (D)^{B} \Theta, (D)^{M} M, [D] \neg \Lambda, [D] \Delta, \neg \Lambda, \Delta \\
(D)^{E} \Lambda, (D)^{B} \Theta, (D)^{M} M, [D] \neg \Gamma, [D] \Delta, \neg \Gamma, \Delta
\end{array}
\]

This rule splits the requests over three nodes accordingly to their classification. If a request cannot appear in a node, it introduces the corresponding negation. The generated nodes have a complete temporal fragment and are reflexive since all box arguments belong to them.

We deal with the middle node in a way similar to the case of \( D_L \) -structures (see Section 5.2). First, we apply the Boolean Rules until they are no longer applicable. Then, we apply the following **Middle Step Rule**:

\[
\begin{array}{c}
\Phi, (D)^M \mu_1, \ldots, (D)^M \mu_n, [D] \Gamma \\
\mu_1, \Gamma, [D] \Gamma \mid \mu_n, \Gamma, [D] \Gamma
\end{array}
\]

For every request in the current node, this rule creates an irreflexive successor of it. Then, we re-apply the expansion procedure from the beginning for every newly generated node.

The expansion of a beginning node takes place as follows. As usual, we first apply the Boolean Rules to it, and to the newly generated nodes, until they are applicable. Then, for any \( (D)^B \psi \) formula in the current node, we distinguish two cases: \( (D)^B \psi \) can be fulfilled in the cluster or it can be fulfilled in one of its descendants. They are dealt with the following **Build Beginning Cluster Rule**:

\[
\begin{array}{c}
\Phi, (D)^B \psi, (D)^B \Gamma B, (D)^{BNC} \Gamma, (D)^{BC} \Gamma Bc, (D)^{M} M, [D] \Delta \\
\psi, (D)^B \Gamma B, (D)^{BNC} \Gamma Bc, (D)^{M} M, [D] \Delta \\
(D)^{BNC} \Gamma Bc, (D)^{M} M, [D] \Delta, \Delta
\end{array}
\]

The former case is handled by the first branch, which marks the request as \( (D)^{BC} \psi \) (in order to avoid loops) and satisfies \( \psi \) in a new cluster node with the same temporal fragment as the current one. The latter case is handled by the second branch that simply reclassifies the request as \( (D)^{BNC} \psi \) without moving to another cluster node. Such a procedure is iterated until every \( (D)^B \psi \) is re-marked as \( (D)^{BC} \psi \) or \( (D)^{BNC} \psi \).

The case of ending nodes is dealt with in a very similar way by means of the following **Build Ending Cluster Rule**:

\[
\begin{array}{c}
\Phi, (D)^E \psi, (D)^E \Gamma E, (D)^{E} \Gamma E, (D)^{ENC} \Gamma E, (D)^{M} M, [D] \Delta \\
\psi, (D)^E \Gamma E, (D)^{E} \Gamma E, (D)^{ENC} \Gamma E, (D)^{M} M, [D] \Delta \\
(D)^{E} \Gamma E, (D)^{ENC} \Gamma E, (D)^{M} M, [D] \Delta, \Delta
\end{array}
\]

Once we reach a cluster node such that no Boolean rules are applicable and every \( (D)^B \psi \) request has been reclassified as \( (D)^{BC} \psi \) or \( (D)^{BNC} \psi \), we proceed as follows. If the node does not include any \( (D)^{BNC} \psi \) request, we are done (all requests have been satisfied in the cluster). Otherwise (there exists at least one marked formula of the form \( (D)^{BNC} \psi \)), we generate an irreflexive successor of the cluster that, for every formula \( (D)^{BNC} \psi \), satisfies either \( \psi \) or \( (D)^B \psi \). This last case is handled by the formulas \( \Gamma Bc \lor (D)^B \Gamma Bc \) introduced by the following **Exit Beginning Cluster Rule**:
5. \(D\)-logic over dense linear orderings

\[
\frac{\Phi \land (D)_{BC} \Gamma_{BC}, (D)_{BNC} \Gamma_{BNC}, (D)_{M} \Delta}{\Gamma_{BNC} \lor (D)_{BC} \Gamma_{BC}, (D)_{BNC} \Gamma_{BNC}, (D)_{M} \Delta} \quad \text{where } \Gamma_{BNC} \neq \emptyset.
\]

The case of the ending cluster is dealt with in a very similar way by means of the following Exit Ending Cluster Rule:

\[
\frac{\Phi \land (D)_{EC} \Gamma_{EC}, (D)_{ENC} \Gamma_{ENC}, (D)_{M} \Delta}{\Gamma_{ENC} \lor (D)_{EC} \Gamma_{EC}, (D)_{ENC} \Gamma_{ENC}, (D)_{M} \Delta} \quad \text{where } \Gamma_{ENC} \neq \emptyset.
\]

Then, we apply again all steps from the beginning, with only a little difference in the application of the Marking Rule. The Completion Rule may produce some requests \((D)\psi\) devoid of any markings. For all these requests, we must check whether they have been marked as \((D)^{B}\psi\) or \((D)^{E}\psi\) in an ancestor of the current node and, if this is the case, we must guarantee the downward propagation of their markings. To this end, before applying the Marking Rule, we apply the following Persistent Beginning and Persistent Ending Rules:

\[
\frac{\Phi \land (D)\psi}{\Phi, (D)^{B}\psi} \quad \frac{\Phi \land (D)\psi}{\Phi, (D)^{E}\psi}
\]

whenever \((D)^{B}\psi\) (resp., \((D)^{E}\psi\)) belongs to an ancestor of the current node.

Building the tableaux.

As in the case of \(D\)-logic, a tableau for a \(D\)-formula \(\varphi\) is a finite graph \(T = (V, E)\), whose vertices are subsets of \(\text{CL}(\varphi)\) and whose edges are generated by the application of expansion rules. The construction of the tableau starts with the initial tableau, which is the single node graph \(\langle\{\varphi\}\rangle\). To describe such a construction process, we take advantage of macronodes, which can be viewed as the counterpart of vertices of \(D\)-structures.

Given a set \(V' \subseteq V\), let \(E(V')\) be the restriction of \(E\) to vertices in \(V\). Moreover, let the Reflexive Step, Middle Step, Build Beginning/Ending Cluster and Exit Beginning/Ending Cluster rules be called Step Rules. Macronodes are defined as follows.

**Definition 5.3.10.** Let \(\langle V, E \rangle\) be a tableau for a \(D\)-formula \(\varphi\). A macronode is a set \(V' \subseteq V\) such that:

- \(\langle V', E(V') \rangle\) is a tree;
- the root of \(\langle V', E(V') \rangle\) is either the initial node of the tableau or a node generated by an application of a Step Rule;
- every edge in \(E(V')\) is generated by the application of an expansion rule which is not a Step Rule;
- the only expansion rule that can be applied to the leaves of \(\langle V', E(V') \rangle\) is a Step Rule.

A macronode \(m\) is reflexive if its root is generated by the application of the Reflexive Step Rule or of the Build Beginning/Ending Cluster Rules; otherwise, it is irreflexive.

We say that a rule is applicable to a node \(n\) if it generates at least one successor node. The construction of a tableau for a \(D\)-formula \(\varphi\) starts with the initial tableau \(\langle\{\varphi\}\rangle, \emptyset\) and proceeds by applying the following expansion strategy to the leaves of the current tableau, until it cannot be applied anymore.

Apply the first rule in the list whose condition is satisfied:

1. a Boolean Rule is applicable;
2. the Completion Rule is applicable;
3. The node belongs to an irreflexive macronode and the Persistent Beginning Rule is applicable;
4. the node belongs to an irreflexive macronode and the Persistent Ending Rule is applicable;
5. the node belongs to an irreflexive macronode and the Marking Rule is applicable;
6. the node belongs to an irreflexive macronode and the Reflexive Step Rule is applicable;
7. the node belongs to a reflexive macronode with only $M$ markings and the Middle Step Rule is applicable;
8. the node belongs to a reflexive macronode with $B$ markings or $E$ markings and the Build Beginning/Ending Cluster Rules are applicable;
9. the node belongs to a reflexive macronode with $B$ markings or $E$ markings and the Exit Beginning/Ending Cluster Rules are applicable.

Termination is ensured by the following looping conditions:

- if an application of the Reflexive Rule generates a node which is the root of an existing reflexive macronode, then add an edge from the current node to this node instead of creating the new one.
- if the Middle Step Rule is applied to a node $n$ and one of the successor nodes it generates, say $n'$, is such that $\text{TF}(n') = \text{TF}(n)$, then add the edge $(n', n)$ to the tableau. Do not apply any expansion rule to $n'$.

We say that a node $n$ in a tableau is closed if one of the following conditions holds:

- there exists $\psi$ such that both $\psi$ and $\neg\psi$ belong to $n$;
- a Middle Step Rule or a Reflexive Step Rule have been applied to $n$ and at least one of its successors is closed;
- a rule different from the Middle Step Rule and the Reflexive Step Rule has been applied to $n$ and all its successors are closed;
- $n$ is a descendant of a node $n'$ to which an Exit Beginning/Ending Cluster Rule has been applied and $\text{TF}(n') = \text{TF}(n)$.

A node in a tableau is open if it is not closed. A tableau is open if and only if its root is open. We will prove that a formula is satisfiable if and only if there exists an open tableau for it.

As for computational complexity, it is not difficult to show that the proof of Theorem 5.3.8 can be adapted to the proposed tableau method. The only difference is that at any step of the tableau construction we either expand a node or mark one of its formulas. As a consequence, any node of a $\mathcal{D}_\subseteq$-structure corresponds to a path of at most $|\phi|$ nodes in the tableau. Hence, the depth of the tableau is bounded by $2 \cdot |\phi|^2$. Since the breadth of the tableau is $2 \cdot |\phi|$, we can conclude that the proposed tableau-based decision procedure is in $\text{PSPACE}$ (and thus optimal).

**Theorem 5.3.11.** (Complexity) The proposed tableau procedure is in $\text{PSPACE}$.

**Example of application.**

Here we give an example of the above-described expansion strategy at work. Consider the formula $\varphi = (D)p \land (D)q \land \neg (D)\neg p \land (D)q$, which states that the given interval has a subinterval where $p$ holds and a subinterval where $q$ holds, but no subintervals covering both of them. It is easy to see that in any model for this formula $p$ and $q$ respectively hold in a beginning and an ending subinterval only, or vice versa. Part of the tableau for $\varphi$ is depicted in Figure 5.6. Due to space limitations, we restrict our attention to the non-closed region of the tableau and we skip.
the details about the application of Boolean Rules. We start with the root A, whose temporal fragment is complete, and we apply the Marking Rule. For the sake of conciseness, we only consider a correct marking, which inserts \( (D)^B_p \) and \( (D)^E_q \) in B. Once all \( (D) \) formulas have been marked, we apply the Reflexive Step Rule, that generates the three successors of B. The first successor is node C that contains the request \( (D)^B_p \) and the negation of the request \( (D)^E_q \), namely, \([D] \neg q\). The second one is node E that contains the request \( (D)^E_q \) and the negation of the request \( (D)^B_p \), namely, \([D] \neg p\). The third one is node D that contains the negation of the two requests (such a node represents the middle reflexive vertex of the corresponding \( D \_r \)-structure). Node D contains no \( (D) \) formulas and thus it cannot be expanded anymore. Since it does not include any contradiction, we declare it open. Consider now node C. According to the expansion strategy, we apply the Build Beginning Cluster Rule to \( (D)^B_p \) in node C, that generates nodes F and G. Node F includes \( p \) and, accordingly, replaces \( (D)^B_p \) with \( (D)^BC_p \). It does not contain \( (D)^BNC \) formulas and no expansion rules are applicable to it. Since it does not include any contradiction, we declare it open. The same argument can be applied to nodes E and H. This allows us to conclude that the tableau is open (and thus \( \varphi \) is satisfiable).

![Figure 5.6: (Part of) the tableau for \( \varphi = (D)p \land (D)q \land (D)\neg p \land (D)\neg q \).](image)

To better explain the proposed tableau method, we include in Figure 5.6 additional nodes which are not strictly necessary to conclude that the tableau is open. This is the case with node G that replaces \( (D)^B_p \) with \( (D)^BNC_p \), thus postponing the satisfaction of \( p \). According to the expansion strategy, we apply the Exit Beginning Cluster Rule to G, that generates the irreflexive node L. Such a node contains the formula \( (D)^B_p \lor p \), stating that \( p \) is satisfied either in L or in
5.3. The logic $D_{\subset}$ of proper subinterval structures

some descendant of it. The application of the Or Rule to $(D)^B p \lor p$ generates nodes $M$ and $N$. Node $M$ includes again the formula $(D)^B p$ and, since $TF(M) = TF(G)$, we declare it closed. As for node $N$, that satisfies $p$, we apply the Completion Rule (neither $(D)^B p$ nor $D\neg p$ belongs to $N$), that generates its two successors. The first successor turns out to be identical to $M$ and thus we add an edge from $N$ to $M$ instead of adding a new node; the second successor is node $O$, with $TF(O) \subset TF(G)$. Then, we apply Reflexive Step Rule to node $O$. Since it does not contain any $(D)$ formula, its three reflexive successors coincides with node $D$. Hence, we add an edge from $O$ to $D$ and we stop the expansion of (this part of) the tableau.

5.3.4 Soundness and completeness

We conclude the section by proving soundness and completeness of the proposed tableau method.

**Theorem 5.3.12. (Soundness) Let $\varphi$ be a $D_{\subset}$ formula and $T$ be a tableau for it. If $T$ is open, then $\varphi$ is satisfiable.**

**Proof.** We build a fulfilling $D_{\subset}$-structure $S = ⟨⟨V, E⟩, ℒ, B, E⟩$ for $\varphi$ step by step, starting from the root of $T$ and proceeding according to the expansion rules that have been applied in the construction of the tableau.

Let $n_0$ be the root of $T$. We generate the one-node $D_{\subset}$-graph $⟨⟨v_0⟩, 0⟩$ and we put formulas belonging to $n_0$ in $ℒ(v_0)$. Now, let $n$ be an open node in $T$ and let $v$ be the corresponding vertex in the $D_{\subset}$-graph. The way in which we develop the $D_{\subset}$-structure depends on the expansion rule that has been applied to $n$ during the construction of the tableau.

- **A Boolean Rule has been applied.** Then, at least one successor $n'$ of $n$ is open. We add formulas belonging to $n'$ to $ℒ(v)$ and we proceed by taking into consideration the tableau node $n'$ and the vertex $v$.

- **The Completion Rule has been applied.** Then, at least one successor $n'$ of $n$ is open. As in the previous case, we add formulas belonging to $n'$ to $ℒ(v)$ and we proceed by taking into consideration the tableau node $n'$ and the vertex $v$.

- **The Marking/Persistent Beginning/Persistent Ending Rule has been applied.** Let $(D)\psi$ be the formula to which the rule has been applied and let $n'$ be one of the open successors of $n$. Four cases may arise, depending on which marking has been applied to the considered formula in $n'$:
  - if $(D)^B \psi \in n'$, then we put $(D)\psi \in B(v)$;
  - if $(D)^E \psi \in n'$, then we put $(D)\psi \in E(v)$;
  - if $(D)^B \psi \in n'$, then we add $(D)\psi$ to both $B(v)$ and $E(v)$;
  - if $(D)^M \psi \in n'$, then the marking does not influence the construction of the $D_{\subset}$-structure.

In all cases, we proceed recursively by taking into consideration the tableau node $n'$ and the current vertex $v$.

- **The Reflexive Step Rule has been applied.** Since $T$ is open, all successors of $n$ are open either. Let $n_b$, $n_c$, and $n_e$ be the first, second, and third successor of $n$, respectively. We add three reflexive vertices $v_b$, $v_c$, and $v_e$ to $V$ and the edges $⟨v, v_b⟩$, $⟨v, v_c⟩$, $⟨v_b, v_c⟩$, $⟨v_b, v_e⟩$, $⟨v_c, v_e⟩$, and $⟨v_e, v_e⟩$ to $E$. The labeling of $v_b$, $v_c$, and $v_e$ is defined as follows: $ℒ(v_b) = n_b$, $ℒ(v_c) = n_c$, and $ℒ(v_e) = n_e$. We recursively apply the construction by taking into consideration the node $n_b$ with the corresponding vertex $v_b$, the node $n_c$ with the corresponding vertex $v_c$, and the node $n_e$ with the corresponding vertex $v_e$.

- **The Middle Step Rule has been applied.** Since $n$ is open, all its successors $n_1, ..., n_h$ are open either. We add $h$ new vertices $v_1, ..., v_h$ to $V$ and the edges $⟨v, v_1⟩, ..., ⟨v, v_h⟩$ to $E$, and we define their labeling in such a way that for $i = 1, ..., h$, $ℒ(v_i) = n_i$. We recursively apply the construction to every node $n_i$ paired with the corresponding vertex $v_i$. 
• **The Build Beginning/Ending Cluster Rule has been applied.** Suppose that the rule has been applied to a formula $({D})^B\psi \in n$ (the case of $({D})^E\psi$ is analogous) and let $n'$ be an open successor of $n$. Two cases may arise:

1. $({D})^BC\psi \in n'$ ($({D})\psi$ has been satisfied in the cluster). We introduce a new node $v'$ in the cluster of $v$ by adding the edges $(v, v'), (v', v')$, and $(v', v)$ to $E$. The labeling $\mathcal{L}(v')$ of $v'$ consists of the set of formulas belonging to $n'$. We proceed by taking into consideration the node $n'$ and the corresponding vertex $v'$.

2. $({D})^BNC\psi \in n'$ (satisfaction of $({D})\psi$ has been postponed). We do not add any vertex to the $D_\prec$-structure, but simply proceed by taking into consideration the node $n'$ and the current vertex $v$.

• **The Exit Beginning/Ending Cluster Rule has been applied.** Since $T$ is open, the unique successor $n'$ of $n$ is open and it is the root of an irreflexive macronode. We add a new irreflexive vertex $v'$ to $V$ and an edge $(v, v')$ to $E$. Moreover, we set the labeling of $v'$ as the set of formulas belonging to $n'$. Then, we proceed by taking into consideration the node $n'$ with the corresponding vertex $v'$.

To keep the construction finite, whenever the procedure reaches a tableau node $n'$ that has been already taken into consideration, instead of adding a new vertex to the $D_\prec$-structure, it simply adds an edge from the current vertex $v$ to the vertex $v'$ corresponding to $n'$.

Since any tableau for $\varphi$ is finite, such a construction is terminating. However, the resulting structure $\langle (V, E), \mathcal{L}, \mathcal{B}, \mathcal{E} \rangle$ is not necessarily a $D_\prec$-structure: there may exist a vertex $v \in V$ and a non-temporal formula $\psi \in \mathcal{L}(v)$ such that neither $\psi$ nor $\neg \psi$ belongs to $\mathcal{L}(v)$. To overcome this problem, we can consistently extend the labeling $\mathcal{L}(v)$ as follows:

• if $\psi = p$, with $p \in \mathcal{A}P$, we put $\neg p \in \mathcal{L}(v)$;

• if $\psi = \neg \xi$, we put $\psi \in \mathcal{L}(v)$ if and only if $\xi \notin \mathcal{L}(v)$;

• if $\psi = \psi_1 \lor \psi_2$, we put $\psi_1 \lor \psi_2 \in \mathcal{L}(v)$ if and only if $\psi_1 \in \mathcal{L}(v)$ or $\psi_2 \in \mathcal{L}(v)$.

The resulting $D_\prec$-structure $\langle (V, E), \mathcal{L}, \mathcal{B}, \mathcal{E} \rangle$ is a fulfilling $D_\prec$-structure for $\varphi$ and thus $\varphi$ is satisfiable. $\square$

**Theorem 5.3.13. (Completeness)** Let $\varphi$ be a $D_\prec$ formula. If $\varphi$ is satisfiable, then there exists an open tableau for it.

**Proof.** Let $S = \langle (V, E), \mathcal{L}, \mathcal{B}, \mathcal{E} \rangle$ be a fulfilling $D_\prec$-structure that satisfies $\varphi$. We take advantage of such a structure to show that there exists an open tableau $T$ for $\varphi$. In particular, we will define a correspondence between (some) nodes in $T$ and vertices in $S$ that satisfies the following constraints:

1. if $n$ is associated with an irreflexive vertex $v$, then $n$ belongs to an irreflexive macronode;

2. if $n$ is associated with a reflexive vertex $v$, then $n$ belongs to a reflexive macronode;

3. if $n$ is associated with a vertex $v$, then, for every formula $\psi \in n$, $\psi \in \mathcal{L}(v)$.

Let $n_0$ be the root of the tableau. We associate it with the root $v_0$ of $S$. Since $n_0$ belongs to an irreflexive macronode, $v_0$ is an irreflexive vertex, and $\varphi \in \mathcal{L}(v_0)$, all constraints are satisfied.

Let $n$ be the current node of the tableau, $v$ be the vertex of $S$ associated with it, and, by inductive hypothesis, $n$ and $v$ satisfy the constraints. We proceed by taking into consideration the rule that, according to the expansion strategy, is applicable to node $n$.

• **One of the Boolean Rules is applicable.** We consider the application of the OR Rule to a formula of the form $\psi_1 \lor \psi_2$ (the other cases are simpler and thus omitted). Since $\psi_1 \lor \psi_2 \in n$, by Constraint (3), $\psi_1 \lor \psi_2 \in \mathcal{L}(v)$ and thus $\psi_1 \in \mathcal{L}(v)$ or $\psi_2 \in \mathcal{L}(v)$. If $\psi_1 \in \mathcal{L}(v)$, then we associate the successor $n_1$ of $n$, that contains $\psi_1$, with $v$; otherwise, we associate the successor $n_2$ of $n$, that contains $\psi_2$, with $v$. In either cases, all constraints are satisfied.
5.3. The logic $D_C$ of proper subinterval structures

- **The Completion Rule is applicable.** Let us consider the application of the Completion Rule to the formula $\langle D \rangle \psi$. Since $\mathcal{L}(v)$ is an atom, either $\langle D \rangle \psi \in \mathcal{L}(v)$ or $\langle D \rangle \neg \psi \in \mathcal{L}(v)$. In the former case, we associate the successor $n_1$ of $n$, that contains $\langle D \rangle \psi$, with $v$; in the latter case, we associate the successor $n_2$ of $n$, containing $\langle D \rangle \neg \psi$, with $v$. In either cases, all constraints are satisfied.

- **The Marking Rule is applicable.** Let us consider the application of the Marking Rule to the formula $\langle D \rangle \psi$. According to the expansion strategy, $n$ belongs to an irreflexive macronode and thus, by inductive hypothesis, $v$ is an irreflexive vertex. Let $\mathcal{C}_b$ be the beginning successor cluster of $v$, $\mathcal{C}_e$ the ending successor cluster of $v$, and $\nu_e$ their common reflexive successor (see Definition 5.3.1). Four cases may arise:

  1. $\langle D \rangle \psi$ appears in $\mathcal{C}_b$, but not in $\mathcal{C}_e$ and $\nu_e$. In this case, we associate the successor $n'$ of $n$, which includes $\langle D \rangle^B \psi$, with $v$.
  2. $\langle D \rangle \psi$ appears in $\mathcal{C}_e$, but not in $\mathcal{C}_b$ and $\nu_e$. In this case, we associate the successor $n'$ of $n$, which includes $\langle D \rangle^E \psi$, with $v$.
  3. $\langle D \rangle \psi$ appears in $\mathcal{C}_b$ and $\mathcal{C}_e$, but not in $\nu_e$. In this case, we associate the successor $n'$ of $n$, which includes $\langle D \rangle^{BE} \psi$, with $v$.
  4. $\langle D \rangle \psi$ appears in $\mathcal{C}_b$, $\mathcal{C}_e$, and $\nu_e$. In this case, we associate the successor $n'$ of $n$, which includes $\langle D \rangle^M \psi$, with $v$.

- **The Persistent Beginning/Ending Rule is applicable.** We associate the unique successor $n'$ of $n$ with $v$.

- **The Reflexive Step Rule is applicable.** According to the expansion strategy, $n$ belongs to an irreflexive macronode and thus, by inductive hypothesis, $v$ is an irreflexive vertex. Let $\nu_b$ be a node in the beginning successor cluster of $v$, $\nu_e$ a node in the ending successor cluster of $v$, and $\nu_e$ the common reflexive successor of the two clusters. According to the expansion strategy, when such a rule turns out to be applicable, all $\langle D \rangle$ formulas have already been marked in accordance with $S$. Let $n = \{ \Phi, \langle D \rangle^B \Gamma, \langle D \rangle^M \Lambda, \langle D \rangle^E \Theta, \langle D \rangle^{BE} \Lambda, \langle D \rangle^M \Delta \}$, where $\Phi$ only contains atomic formulas. We have that $\langle D \rangle^B \Gamma, \langle D \rangle \Theta, \langle D \rangle^M \Lambda, \langle D \rangle^{BE} \Lambda, \langle D \rangle^M \Delta \subseteq \mathcal{L}(v_b)$, that $\langle D \rangle^M \Lambda, \langle D \rangle \Theta, \langle D \rangle^M \Delta \subseteq \mathcal{L}(v_e)$, and that $\langle D \rangle^E \Lambda, \langle D \rangle^{BE} \Lambda, \langle D \rangle^M \Delta \subseteq \mathcal{L}(v_e)$. We associate the first successor of $n$ with $\nu_b$, the second one with $\nu_e$, and the third one with $\nu_e$.

- **The Middle Step Rule is applicable.** According to the expansion strategy, $n$ belongs to a macronode whose root is the middle node generated by an application of the Reflexive Step Rule and thus, by inductive hypothesis, $n$ is associated with a middle reflexive vertex $\nu_e$. Since $S$ is fulfilling, for every formula $\langle D \rangle \psi \in n$ there exists a successor $\nu_b$ of $\nu_e$ such that $\psi \in \mathcal{L}(\nu_b)$ and for every $\langle D \rangle \theta \in n$, $\theta, \langle D \rangle \theta \in \mathcal{L}(\nu_b)$. For all $\langle D \rangle \psi \in n$, we associated the successor $n_0 \psi$ of $n$, with $\nu_b$.

- **The Build Beginning Cluster Rule is applicable.** Given the expansion strategy, by inductive hypothesis we have that $n$ is associated with a node $v$ that belongs to a beginning cluster $\mathcal{C}$. Let us consider the application of the rule to the formula $\langle D \rangle^B \psi$. Two cases may arise: either $S$ fulfills $(D) \psi$ outside $\mathcal{C}$ or not. In the former case, we associate the successor $n'$ of $n$, that contains $\langle D \rangle^{BNC} \psi$, with $v$; in the latter case, there exists a node $v' \in \mathcal{C}$ such that $\psi \in \mathcal{L}(v')$ and we associate the successor $n'$ of $n$, that contains both $\psi$ and $\langle D \rangle^{BC} \psi$, with $v'$.

- **The Build Ending Cluster Rule is applicable.** This case is analogous to the previous one and thus omitted.
The Exit Beginning Cluster Rule is applicable. Given the expansion strategy, by inductive hypothesis we have that \( n \) is associated with a node \( v \) that belongs to a beginning cluster \( \mathcal{C} \). Let \( v' \) be the unique irreflexive successor of \( \mathcal{C} \). We have that, for every formula \( (D)^{\text{BNC}} \psi \in n \), \( \psi \in \mathcal{L}(v') \) or \( (D)\psi \in \mathcal{L}(v') \). The labeling of the unique successor node \( n' \) of \( n \) is thus consistent with \( v' \) and we can associate \( n' \) with \( v' \).

The Exit Ending Cluster Rule is applicable. This case is analogous to the previous one and thus omitted.

At the end of the above construction, we have obtained (a portion of) a tableau for \( \varphi \). Since all its nodes are open, we can conclude that there exists an open tableau for \( \varphi \).

5.4 Implementation

In this section we briefly describe an implementation of the proposed tableau methods in Lotrec, a generic theorem prover for modal and description logics [23, 28]. We start with a short account of the main features of Lotrec. Then, we will point out the distinctive features of the proposed implementations of the tableau methods for \( D_\text{E} \) and \( D_\text{C} \), with a special attention to that for \( D_\text{C} \). Finally, we will illustrate the behavior of the system on a concrete example.

5.4.1 A short overview of Lotrec

Lotrec is a generic prover that can be used for most modal logics studied in the literature. It can be used to prove validity and satisfiability of formulas. Whenever a formula is satisfiable, it returns a model for it; whenever a formula is not valid, it returns a counter-model for it. In Lotrec, a tableau is a special kind of labeled graph that is built, and possibly revised, according to a set of user-specified rules. Every node of the graph is labeled with a set of formulae and can be enriched by auxiliary markings, if needed.

A Lotrec program is divided into three parts. The first part defines the syntax of the considered logic. The second part defines a set of expansion rules to apply to the nodes of the graph during the computation. Rules have the following form:

```
rule ruleName
  if condition_1
    : 
    if condition_n
    do action_1
      : 
      do action_m
end
```

The conditional part of a rule, which consists of all lines starting with `if`, defines the conditions that must be satisfied in order to apply the rule to a certain node of the graph. When a rule is applied to a node, the current labeled graph is modified in accordance with the action part of the rule, which consists of the lines starting with `do`. As an example, consider the following two rules that encode the Exit Beginning/Ending Cluster Rule of the tableau for \( D_\text{C} \):

```
rule clusterStep
  if isMarked node_0 CLUSTER
    if hasElement node_0 ((D) (variable A))
```

```
if isMarkedExpression node₀ ((D) (variable A)) DELAY
  do createOneSuccessor node₁ (R)
  do add node₁ (variable A) V ((D) (variable A))
end

rule clusterDelay
  if isMarked node₀ CLUSTER
  if isLinked node₀ node₁ (R)
  if hasElement node₀ ((D) (variable A))
  if isMarkedExpression node₀ ((D) (variable A)) DELAY
  do add node₁ (variable A) V ((D) (variable A))
end

The first rule deals with the case when the current node has no successors and it creates a unique irreflexive successor of a cluster (by the createOneSuccessor action). The rule is applicable only to nodes containing at least one ⟨D⟩-formula with a DELAY label, that is, a formula which is not satisfied in the current cluster, and it puts the formula (variable A) V (D)(variable A) in the labeling of the new node. The second rule completes the labeling of the new successor by adding the formula (variable A) V (D)(variable A) to it for every formula ⟨D⟩(variable A) with a DELAY label in the current node. In a similar way, all rules of our tableau methods can be encoded in Lotrec.

A Lotrec program ends with a third part devoted to the definition of the expansion strategy to be used to construct the tableau. As an example, the expansion strategy of the tableau for Dᵥ (see Section 5.2) can be easily encoded as follows:

strategy strictStrategy
  repeat
    firstRule
      stopRule
      notRule
      andRule
      orRule
      reflRule
      twoDensRule
      stepRule
      blockingRule
  end
end

In such a case, the computation proceeds by repeatedly applying the first expansion rule that is applicable, following the order given in the strategy. The stopRule simply checks for contradictions and stop with failure if a node contains both a formula \( \psi \) and its negation \( \neg \psi \). The notRule, andRule, orRule, reflRule, twoDensRule, and stepRule are Lotrec translations of the (NOT), (AND), (OR), (REFL), (2-DENS), and (STEP) Rules of the tableau method for Dᵥ. Finally, blockingRule implements the blocking conditions that avoid infinite expansions of the tableau.
5.4.2  Lotrec implementation of the tableau for $D_C$

The tableau methods for $D_C$ and $D_C$ can be implemented in Lotrec by appropriately encoding their expansion rules and their expansion strategies. In the case of $D_C$, we take advantage of Lotrec rewriting system to remove the Completion Rule.

The tableau method for $D_C$ presented in Section 5.3 applies expansion rules only to the leaves of the current tableau. Moreover, the labeling of a node is defined when the node is created and it is not changed by the application of successive expansion steps. Because of this, we need the Completion Rule to completely expand the temporal fragment of a node to make it possible to immediately detect possible contradictions.

The application of Lotrec rules is not confined to leaves, and it can modify nodes which are ancestors of the current one. This feature allows us to remove the Completion Rule from the Lotrec implementation of the tableau method for $D_C$ and to substitute it with the following rule that propagates back new $(D)$-formulas as soon as they appear in a node.  

\textbf{rule}  
\begin{verbatim}
backwardDiamond  
  if hasElement node0 ((D) (variable A))  
  if isLinked node1 node0 (R)  
  if hasNotElement node1 ((D) (variable A))  
  do add node1 ((D) (variable A))  
end

\end{verbatim}

This rule allows us to add a new $(D)$-formula to a node when it appears in some of its descendants, instead of guessing its belonging to the labeling of the node when the node is created, thus saving computation time in the average case. When this rule is applied, Lotrec behaves as follows. First, it adds $(D)$ (variable A) to the ancestor node of node0. If this causes a contradiction, the current branch of the tableau is declared closed. Otherwise, it proceeds with the expansion following the given expansion strategy.

5.4.3  An application example

In this section we describe the tableau generated by Lotrec for the following unsatisfiable formula:

$$\varphi = (D)p \land (D)q \land (D)^{-}(D)p \land (D)q \land (D)(D)r \rightarrow (D)^{-}(D)p \land (D)^{-}(D)q \land (D)(D)r$$

The formula $\varphi$ imposes that $p$ and $q$ must be satisfied on opposite sides of the given interval, that $r$ must be satisfied over some subinterval $[b_r, e_r]$ of the given one, and that every superinterval of $[b_r, e_r]$ is such that both $(D)^{-}p$ and $(D)^{-}q$ hold over it. The formula $\varphi$ can be easily shown to be unsatisfiable. Assume $p$ to be satisfied over some initial subinterval $[b_0, e_p]$, and take an initial subinterval $[b_0, e]$ that covers both $[b_0, e_p]$ and $[b_r, e_r]$. Since $[b_0, e_p] \subseteq [b_0, e]$, $[b_0, e]$ satisfies both $(D)p$ and $(D)r$, contradicting $\varphi$ requests. The case in which $p$ is satisfied over an ending interval leads to an analogous contradiction.

Such an example allows us to display the role of backward propagation of diamond formulas in order to ensure correctness. The tableau for $\varphi$ generated by Lotrec is depicted in Figure 5.7. Its construction starts with the root A and proceeds with the creation of its three reflexive successors B, C, and D. Let us assume that the partition rule classifies $(D)p$ as a beginning request, $(D)q$ as an ending request, and $(D)(D)r$ as a middle one. At this stage, the formula $(D)r$ does not appear explicitly in any node of the tableau. Hence, B (resp., C) can satisfy $p$ (resp., $q$) in its cluster without contradiction. The request $(D)r$ appears for the first time in node E when Lotrec tries to fulfill the $(D)(D)r$ request of node D.

At this point, Lotrec propagates $(D)r$ backward to D and then to A. When $(D)r$ is added to A, it must be classified either as a begin, an end, or a middle request. The only admissible
option is to classify it as a middle request (otherwise, we should put $\langle D \rangle r$ in $D$ and immediately close the tableau). Since $\langle D \rangle r$ is a middle request, it is propagated downward from $A$ to $B$ and $C$. The presence of both $\langle D \rangle r$ and $\langle D \rangle r \to \lnot (\langle D \rangle p \land \langle D \rangle q)$ in $B$ forces the addition of $\lnot p$ in $B$, thus causing a contradiction with $\langle D \rangle p$. An analogous contradiction occurs in $C$ between $\lnot q$ and $\langle D \rangle q$. In both cases, at least one successor of the root is declared closed and thus the entire tableau is closed. In Figure 5.7 formulas in bold are those added to a node as an effect of the backward propagation of $\langle D \rangle r$. 

---

Figure 5.7: An example of Lotrec tableau.
5. D-logic over dense linear orderings
6 Cone Logic

6.1 Introduction

Spatial reasoning has both a strong theoretical relevance and applications in many areas of computer science, including robotics, natural language processing, geographical information systems [1, 31, 57]. However, despite the widespread interest in the topic, few techniques have been developed to automatically (and efficiently) reason about spatial relations over infinite structures. As a matter of fact, spatial reasoning has been mainly investigated in quite restricted algebraic settings.

In this chapter, we introduce a novel spatial modal logic, called Cone Logic, which allows one to reason about cone-shaped directional relations between points in the rational plane. While the satisfiability problem for spatial modal logics with projection modalities turns out to be highly undecidable [42, 48], we prove that Cone Logic enjoys a decidable satisfiability problem (in fact, PSPACE-complete) by taking advantage of a suitable filtration technique. We also show that Cone Logic subsumes interesting interval temporal logics such as the temporal logic of subintervals/superintervals/begin/begun by/future/past, thus generalizing previous results in the literature [12] and basically disproving a conjecture by Lodaya [39].

6.2 Syntax and semantics of Cone Logic

In this section, we introduce syntax and semantics of Cone Logic. Let $\mathbb{P} = \mathbb{Q} \times \mathbb{Q}$ denote the rational plane and let $p = (x, y)$ be one of its points. We denote by $LL(p)$, $LR(p)$, $UL(p)$, and $UR(p)$ the open lower-left, lower-right, upper-left, and upper-right quadrants of $p$, respectively, which are defined as follows:

$$
LL(p) = \{(x', y') : x' < x, y' < y\} \quad LR(p) = \{(x', y') : x' > x, y' < y\}
$$

$$
UL(p) = \{(x', y') : x' < x, y' > y\} \quad UR(p) = \{(x', y') : x' > x, y' > y\}.
$$

Note that, up to a rotation of the axes, these open quadrants can be viewed as the Frank’s cone-shaped cardinal directions ‘North’, ‘West’, ‘East’, ‘South’ [29] (see Figure 6.1). Similarly, one can denote by $LL^+(p)$, $LR^+(p)$, $UL^+(p)$, and $UR^+(p)$ the semi-closed quadrants of $p$, which are defined in the natural way, e.g., $LL^+(p) = \{(x', y') : x' \leq x, y' \leq y\} \setminus \{p\}$. Moreover given $p = (x, y)$ we define the upper (resp. lower) direction as follows $UP(p) = \{(x, y') : y' > y\} \quad (resp. \quad LOW(p) = \{(x, y') : y' < y\}).$

Given a set $\text{Prop}$ of propositional variables, formulas of Cone Logic are built up from $\text{Prop}$ using the boolean connectives $\neg$, $\vee$ and ten modal operators $\Box$, $\Diamond$, $\Diamond^*$, $\Box^*$, $\Box^+$, $\Diamond^+$, $\Diamond^*$, and $\Box^*$. The size $|\phi|$ of a formula $\phi$ is given by the number of its subformulas (for instance, $\Box a \vee \neg \Diamond \neg b$ is a formula of size 7). Formulas of Cone Logic are evaluated over labeled regions of the rational plane. Precisely, let $\mathcal{P} = (\mathbb{P}, \sigma)$ be a labeled region, where $\mathbb{P} \subseteq \mathbb{P}$ is a non-empty subset of the rational plane and $\sigma : \mathbb{P} \rightarrow \mathcal{P}(\text{Prop})$ is a labeling function. We define the semantics of a formula with respect to a distinguished initial point $p \in \mathbb{P}$ as follows:

- $\mathcal{P}, p \models a$ iff $a \in \sigma(p)$,
- $\mathcal{P}, p \models \neg \phi$ iff $\mathcal{P}, p \not\models \phi$,
Cone Logic

UR(p) \quad UR(p)

North

West \quad P \quad East

South

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{cone_logic_diagram}
\caption{The four quadrants and the cone-shaped cardinal directions.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{labeled_rational_plane}
\caption{A labeled rational plane satisfying $\varphi_\prec$.}
\end{figure}

- $\mathcal{P}, p \models \varphi_1 \lor \varphi_2$ iff $\mathcal{P}, p \models \varphi_1$ or $\mathcal{P}, p \models \varphi_2$,
- $\mathcal{P}, p \models \Diamond \varphi$ (resp., $\mathcal{P}, p \models \Diamond^+ \varphi$) iff $P$ contains a point $q$ such that $q \in LL(p)$ (resp., $q \in LL^+(p)$) and $\mathcal{P}, q \models \varphi$ (and similarly for the other modal operators $\Box$, $\Diamond$, $\Box\Diamond$, $\Box^+$, $\Diamond^+$, $\Box^\ast$ and $\Diamond^\ast$).

We further use shorthands such as $\varphi_1 \land \varphi_2 = \neg(\neg \varphi_1 \lor \neg \varphi_2)$, $\Box \varphi = \neg \Diamond \neg \varphi$, $\varphi = \Diamond \Box \varphi$, $\varphi = \Diamond \Box \varphi = \Diamond \Box \varphi$, $\varphi = \Diamond \Box \varphi$, etc.

Cone Logic is well-suited for expressing spatial relationships between points, curves, and regions inside the rational plane. Below, we give an intuitive account of its expressiveness through a couple of examples. To begin with, we show how to define an $\alpha$-labeled open rectangular region, whose edges are aligned with the $x$- and $y$-axes, by means of a Cone Logic formula:

$$\varphi = \Diamond a \land \Diamond b \land \Diamond c \land \Diamond d \land \Diamond e \land (a \rightarrow \Diamond a \land \Diamond a \land \Diamond a \land \Diamond a) \land (\neg a \leftrightarrow b \lor c \lor d \lor e) \land (b \rightarrow \Diamond b) \land (c \rightarrow \Diamond c) \land (d \rightarrow \Diamond d) \land (e \rightarrow \Diamond e).$$

The second example uses two symmetric modal operators, namely, $\Diamond$ and $\Box$, to enforce non-trivial spatial relationships between labeled regions of the rational plane. Let $\text{Prop}$ be a signature containing $n+2$ propositional variables $a, b_1, \ldots, b_n, c$ and let $< be the partial order over $\text{Prop}$ such that $a < b_1 < c$, for every $1 \leq i \leq n$, and $b_i \nless b_j$, for every pair of distinct indices $1 \leq i, j \leq n$. We shortly write $p \leq q$ (resp., $p \gtrless q$) whenever $p = q$ or $p < q$ (resp., $p > q$). Consider now the (Hintikka-like) formula induced by the partial order $<:

$$\varphi_\prec = \bigvee_{p \in \text{Prop}} p \land \bigwedge_{p \neq q} \neg(p \land q) \land \bigwedge_{p \in \text{Prop}} \left(p \rightarrow \bigwedge_{q \leq p} \Diamond q \land \bigwedge_{q \geq p} \Box q \land \bigvee_{q \leq p} q \land \bigvee_{q \geq p} q\right).$$

The unique (up to homeomorphism) labeled rational plane that satisfies $\varphi_\prec$ is depicted in Figure 6.2. Notice that (i) for every propositional variable $b_i$, with $1 \leq i \leq n$, the $b_i$-labeled region is
an (infinite) union of disjoint open rectangles (in fact, the coordinates of their corners are given by pairs of irrational numbers) and (ii) the $b_i$-labeled open rectangles are arranged densely in the rational plane, that is, for all indices $1 \leq i, j, k \leq n$, with $i \neq j$, all $b_i$-labeled points $(x_1, y_1)$, and all $b_j$-labeled points $(x_2, y_2)$, with $x_1 < x_2$ and $y_1 > y_2$, there is a $b_k$-labeled point $(x, y)$ such that $x_1 < x < x_2$ and $y_1 > y > y_2$.

The *satisfiability problem* for Cone Logic consists of deciding whether a given formula $\varphi$ holds at some point of a labeled region of the rational plane. In particular, we are interested in satisfiability problems under interpretation over (open or closed) rectangular regions, namely, regions of the form $I \times J$, with $I$ and $J$ being two fixed (open or closed) intervals of the rational line$^1$. As a matter of fact, note that the whole rational plane $\mathbb{P}$ is homeomorphic to any open rectangular region of the form $I \times J$, with $I = (x_0, x_1)$ and $J = (y_0, y_1)$. Moreover, any formula $\varphi$ interpreted over an open rectangular region of the form $I \times J$, with $I = (x_0, x_1)$, can be rewritten into an equi-satisfiable formula $\varphi'$ interpreted over the semi-closed rectangular region $I' \times \mathbb{Q}$, where $I' = [x_0, x_1)$. Taking advantage of the reducibility of the satisfiability problem over open rectangular regions to that over semi-closed rectangular regions, we can restrict our attention to labeled regions of the form $\mathbb{P} = \{1 \times \mathbb{Q}, \sigma\}$, where $I$ is a closed (non-singleton) interval (hereafter, we call such structures *labeled stripes*).

The relationships between Cone Logic and spatial logics with projection modalities deserve a little discussion. Projection-based spatial logics (most notably, Compass Logic [60]) are two-dimensional modal logics whose accessibility relations allow one to move along one of the two coordinates while keeping the other coordinate constant. On the one hand, Cone Logic inherits from projection-based modal logics some of their desirable features. For instance, it allows one to write suitable formulas that constrain labels to occur along some distinguished axes, e.g., the formula $\Diamond a \land \Box \neg a \land \Box \neg a$ forces a to hold at the origin or at some point over the positive x-axis. On the other hand, unlike projection-based modal logics, only a *bounded* number of constraints ‘along the axes’ can be enforced by Cone Logic. We will see that such a limitation can be traded for a positive decidability result.

### 6.3 Basic machinery: types, dependencies and shadings

Let us fix a labeled region $\mathbb{P} = \{\mathbb{P}, \sigma\}$ and a formula $\varphi$ of Cone Logic. In the sequel, we compare points in $\mathbb{P}$ with respect to the set of subformulas of $\varphi$ they satisfy. To do that, we introduce the key notions of $\varphi$-atom, $\varphi$-type, $\varphi$-cluster, and $\varphi$-shading.

First of all, we denote by $\mathcal{E}(\varphi)$ the set of all subformulas of $\varphi$ and of their negations (we identify $\neg \neg \alpha$ with $\alpha$, $\neg (\Diamond \alpha)$ with $\Box \neg \alpha$, etc.). We denote with $\mathcal{F}(\varphi)$ the set of all subformulas of $\alpha$ that appear in the scope of some temporal operators $\mathcal{F}(\varphi) = \{\alpha \mid \exists \sigma \in \{\Diamond, \Box, \Diamond, \Box, \Diamond, \Box, \Diamond, \Box, \Diamond, \Box\} \land \forall \alpha \in \mathcal{E}(\varphi)\}$.

Over $\mathcal{E}(\varphi)$ we build $\mathcal{E}^+(\varphi)$ as $\mathcal{E}^+(\varphi) = \mathcal{E}(\varphi) \cup \{\neg \alpha \mid \forall \sigma \in \{\Diamond, \Box, \Diamond, \Box, \Diamond, \Box, \Diamond, \Box\} \land \forall \alpha \in \mathcal{F}(\varphi)\}$.

A $\varphi$-*atom* is any non-empty set $A \subseteq \mathcal{E}^+(\varphi)$ such that:

- for every formula $\alpha \in \mathcal{E}^+(\varphi)$, $\alpha \in A$ iff $\neg \alpha \notin A$;
- for every formula $\gamma = \alpha \lor \beta \in \mathcal{E}^+(\varphi)$, $\gamma \in A$ iff $\alpha \in A$ or $\beta \in A$;
- for every formula $\psi \in \mathcal{E}^+(\varphi)$ and every $\sigma \in \{\Diamond, \Box, \Diamond, \Box\}$ s.t. $\{\sigma \psi, \sigma^+ \psi\} \subseteq \mathcal{E}^+(\varphi)$ if $\sigma \psi \in A$ then $\sigma^+ \psi \in A$;
- for every formula $\psi \in \mathcal{E}^+(\varphi)$ s.t. $\{\Diamond^+ \psi, \Box \psi\} \subseteq \mathcal{E}^+(\varphi)$ (resp. $\{\Box^+ \psi, \Diamond \psi\} \subseteq \mathcal{E}^+(\varphi)$) if $\Diamond \psi \in A$ then $\Box^+ \psi \in A$ (resp. $\Diamond^+ \psi \in A$);
- for every formula $\psi \in \mathcal{E}^+(\varphi)$ s.t. $\{\Diamond^+ \psi, \Box \psi\} \subseteq \mathcal{E}^+(\varphi)$ (resp. $\{\Box^+ \psi, \Diamond \psi\} \subseteq \mathcal{E}^+(\varphi)$) if $\Box \psi \in A$ then $\Diamond^+ \psi \in A$ (resp. $\Box^+ \psi \in A$);

$^1$Hereafter, square brackets are used to denote closed intervals, e.g., $[0,1]$, while round brackets are used to denote open intervals, e.g., $(0,1)$. Semi-open intervals are represented by mixing the two notations, e.g., $[0,1)$. 


Intuitively, a $\varphi$-atom is a maximal locally consistent set of subformulas of $\varphi$ (first two conditions). Moreover it ensures that diamond requests in open quadrants are repeated in the closed one (third condition). Finally if a request appear on the upper (resp. lower) vertical axis it is forced to appear as a request in the two upper (resp. lower) diamonds over closed quadrants (fourth and fifth conditions). The cardinality of $\mathcal{EL}^+(\varphi)$ is linear in $|\varphi|$, while the number of $\varphi$-atoms is (at most) exponential in $|\varphi|$. We then associate with each point $p \in \mathcal{P}$ the set of all formulas $\alpha \in \mathcal{EL}^+(\varphi)$ such that $\mathcal{P}, p \models \alpha$. Such a set is called $\varphi$-type of $p$ and it is denoted by $Type_\varphi(p)$. We have that every $\varphi$-type is a $\varphi$-atom, but not vice versa.

Given an atom $A$, we denote by $\text{Req}_{\mathcal{D}}(A)$ (resp., $\text{Req}_{\mathcal{LR}}(A)$, $\text{Req}_{\mathcal{UL}}(A)$, $\text{Req}_{\mathcal{UR}}(A)$, $\text{Req}_{\mathcal{LR}^+}(A)$, $\text{Req}_{\mathcal{UL}^+}(A)$, $\text{Req}_{\mathcal{UR}^+}(A)$, $\text{Req}_{\mathcal{LOW}}(A)$, $\text{Req}_{\mathcal{UP}}(A)$) the set of all formulas $\alpha \in \mathcal{EL}^+(\varphi)$ such that $\boxdot \alpha \in A$ (resp., $\boxdot \alpha \in A$, $\boxdot \alpha \in A$, $\boxdot \alpha \in A$, $\boxdot ^+ \alpha \in A$, $\boxdot ^+ \alpha \in A$, $\boxdot ^+ \alpha \in A$, $\boxdot \alpha \in A$); similarly, we denote by $\text{Obs}(A)$ the set $A \cap \mathcal{F}(\varphi)$. We call formulas belonging to one of the first (resp., last) ten sets requests (resp., observables). Taking advantage of these sets, for each direction $D \in \{LL, LR, UL, UR, LL^+, LR^+, UL^+, UR^+, LOW, UP\}$, we define two transitive relations $\xrightarrow{D}$ and $\xrightarrow{D}^{-}$ between atoms as follows:

$$A \xrightarrow{D} A' \iff \begin{cases} \text{Req}_D(A) \supseteq \text{Req}_D(A') \cup \text{Obs}(A') \\ \text{Req}_D(A) \supseteq \text{Req}_D(A) \cup \text{Obs}(A) \end{cases}$$

where $D$ denotes the direction opposite to $D$ (e.g., $LL \supseteq UR$). Note that $A \xrightarrow{D} A'$ (resp., $A \xrightarrow{D}^{-} A'$) if $A' \xrightarrow{D}^{-} A$ (resp., $A' \xrightarrow{D} A$). Moreover, the relations $\xrightarrow{D}$ and $\xrightarrow{D}^{-}$ satisfy the view-to-type dependency property, namely, for every pair of points $p, q \in \mathcal{P}$ and every direction $D \in \{LL, LR, UL, UR, LL^+, LR^+, UL^+, UR^+, LOW, UP\}$,

$$q \in D(p) \quad \text{implies} \quad Type_\varphi(p) \xrightarrow{D} Type_\varphi(q)$$

$$D(q) \subseteq D(p) \quad \text{implies} \quad Type_\varphi(p) \xrightarrow{D}^{-} Type_\varphi(q).$$

Below, we introduce analogous notions for regions. First, we define a $\varphi$-cluster as any non-empty set $C$ of atoms. Then, for a cluster $C$ and a direction $D \in \{LL, LR, UL, UR, LL^+, LR^+, UL^+, UR^+, LOW, UP\}$, we denote by $\text{Req}_D(C)$ and $\text{Obs}(C)$, respectively, the set $\bigcup_{A \in C} \text{Req}_D(A)$ and the set $\bigcup_{A \in C} \text{Obs}(A)$. Moreover, given a pair of clusters $C$, $C'$, we write $C \xrightarrow{D} C'$ (resp., $C \xrightarrow{D}^{-} C'$) whenever $A \xrightarrow{D} A'$ (resp., $A \xrightarrow{D}^{-} A'$) holds for all pairs of atoms $A \in C$ and $A' \in C'$. Finally, we associate with each non-empty region $P$ of $\mathcal{P}$ its $\varphi$-shading, which is defined as the set $Type_\varphi(P) = \{Type_\varphi(p) : p \in P\}$ of the $\varphi$-types of all points of $P$.

Note that, for every labeled region $\mathcal{P} = (P, \varphi)$, the formula $\varphi$ holds at some point $p$ of $\mathcal{P}$ iff the shading $Type_\varphi(P)$ contains an atom $A$ such that $\varphi \in A$. Hereafter, we shall omit the argument $\varphi$, thus calling a $\varphi$-atom (resp., a $\varphi$-type, a $\varphi$-cluster, etc.) simply an atom (resp., a type, a cluster, etc.).

### 6.4 From the rational plane to the infinite binary tree

In this section, we aim at establishing a tree (pseudo-)model property for satisfiable formulas of Cone Logic. This is done by introducing a suitable notion of decomposition of a labeled region (more precisely, of a labeled stripe) and by iteratively applying it in order to obtain an infinite decomposition tree structure that correctly represents the original model.

#### 6.4.1 Shading sequences and stripe expressions

To start with, we consider the shadings of the vertical straight lines inside a labeled rational region. A shading sequence is a non-empty finite sequence $S$ of atoms and clusters such that, for every
1 ≤ i ≤ |S|, if S(i) is an atom, then 1 < i < |S| and both S(i−1) and S(i+1) are clusters. Shading sequences allow one to represent the types that appear along some vertical straight lines of a labeled rational plane. As an example, Figure 6.3(a) depicts a labeled vertical line with an associated shading sequence S = C₁ A₂ C₃ C₄.

To represent the shadings of the two vertical borders of a labeled stripe, we introduce the notion of stripe expression, which is a pair E = (L, R) of (left and right) shading sequences having equal length ([L] = |R|) and such that, for every 1 ≤ i ≤ |E| (= |L| = |R|, L(i) is an atom (resp., a cluster) iff R(i) is an atom (resp., a cluster). We call any pair of the form (L(i), R(i)), with 1 ≤ i ≤ |E|, a matched pair. As an example, Figure 6.3(b) depicts the left border and the right border of a labeled stripe, together with the associated stripe expression E = (L, R), where L = C₁ A₂ C₃ and R = C₁ A₂ C₃ A₂. We say that an atom A is featured by the left (resp., right) sequence of a stripe expression E = (L, R) if there is an index 1 ≤ i ≤ |E| such that A = L(i) (resp., A = R(i)) or A ∈ L(i) (resp., A ∈ R(i)), depending on whether L(i) (resp., R(i)) is an atom or a cluster. By a slight abuse of notation, we denote by \( \bigcup_{1 \leq i \leq |E|} L(i) \) (resp., \( \bigcup_{1 \leq i \leq |E|} R(i) \)) the set of all atoms featured by the left (resp., right) sequence of the stripe expression E = (L, R).

For every labeled stripe \( \mathcal{P} \), there is a stripe expression E whose left (resp., right) sequence features all and only the types of the points along the left (resp., right) border of \( \mathcal{P} \). However, for a given stripe expression E, there might exist no labeled stripe \( \mathcal{P} \) such that the shading of the left (resp., right) border of \( \mathcal{P} \) coincides with the set of atoms featured by the left (resp., right) shading sequences of E. In the following, we show how to get rid of such a problem. As a first step, we enforce suitable consistency conditions on any stripe expression E = (L, R):

(C1) for every 1 ≤ i < j ≤ |E|, \( L(i) \rightarrow L(j) \) and \( R(i) \rightarrow R(j) \) hold for both \( D = UL \) and \( D = UR \), \( L(i) \rightarrow L(j) \) and \( R(i) \rightarrow R(j) \) hold for every \( D \in \{UL^+, UR^+, UP\} \);

(C2) for every 1 ≤ i ≤ |E|, if L(i) and R(i) are clusters, then \( L(i) \rightarrow L(i) \) and \( R(i) \rightarrow R(i) \) hold for both \( D = UL \) and \( D = UR \), \( L(i) \rightarrow L(i) \) and \( R(i) \rightarrow R(i) \) hold for every \( D \in \{UL^+, UR^+, UP\} \);

(C3) for every 1 ≤ i ≤ |E|, if \( \alpha \in Req_{UL}(L(i)) \) (resp. \( \alpha \in Req_{UP}(R(i)) \)) then there exists \( j > i \) for which \( \alpha \inObs(L[j]) \) (resp. \( \alpha \inObs(R[j]) \)) or \( L(i) \) (resp. \( R(j) \)) is a cluster and \( \alpha \inObs(L[i]) \) (resp. \( \alpha \inObs(R[i]) \));

(C4) for every 1 ≤ i ≤ |E|, if \( \alpha \in Req_{LOW}(L(i)) \) (resp. \( \alpha \in Req_{LOW}(R(i)) \)) then there exists \( j < i \) for which \( \alpha \inObs(L[j]) \) (resp. \( \alpha \inObs(R[j]) \)) or \( L(i) \) (resp. \( R(j) \)) is a cluster and \( \alpha \inObs(L[i]) \) (resp. \( \alpha \inObs(R[i]) \));
(C5) for every \( 1 \leq i \leq |E| \), \( L(i) \to D \Rightarrow R(i) \) (and hence \( R(i) \to D \Rightarrow L(i) \)) holds for both \( D = LR \) and \( D = UR \), and \( L(i) \to R \Rightarrow R(i) \) (and hence \( R(i) \to R \Rightarrow L(i) \)) holds for both \( D = LR^+ \) and \( D = UR^+ \);

(C6) for every \( 1 \leq i \leq |E| \), if \( L(i) \) and \( R(i) \) are atoms (resp., clusters), then \( L(i) \to R \Rightarrow L(i) \) and \( L(i) \to R \Rightarrow R(i) \) (and hence \( R(i) \to R \Rightarrow L(i) \) and \( R(i) \to R \Rightarrow R(i) \)) hold for all \( 1 \leq j < i \) (resp., \( 1 \leq j \leq i \)) and all \( i < k \leq |L| \) (resp., \( i \leq k \leq |L| \)).

We compare stripe expressions with respect to their generality by introducing a suitable partial order \( \leq \). Given two stripe expressions \( E = (L, R) \) and \( E' = (L', R') \), we write \( E \leq E' \) if \( |E| = |E'| \) and, for every index \( 1 \leq i \leq |E| \), we have either \( L(i) = L'(i) \) and \( R(i) = R'(i) \), or \( L(i) \subseteq L'(i) \) and \( R(i) \subseteq R'(i) \), depending on whether \( L(i) \), \( R(i) \), \( L'(i) \), \( R'(i) \) are atoms or clusters. Unless otherwise stated, we tacitly assume that a stripe expression is maximal with respect to the above-defined partial order \( \leq \). Note that a cluster appearing in a (maximal) stripe expression may contain an exponential number of distinct atoms; however, thanks to consistency conditions, the set of all its atoms can be characterized in terms of the sets of its requests and observables, namely, for every (maximal) stripe expression \( E = (L, R) \), every index \( 1 \leq i \leq |E| \), and every atom \( A \), we have that \( A \) belongs to the cluster \( C = L(i) \) (resp., \( C = R(i) \)) if and only if \( \text{REQ}_D(A) = \text{REQ}_D(C) \) and \( \text{OBS}(A) \subseteq \text{OBS}(C) \) hold for all directions \( D \in \{ LL, LR, UL, ULR, UL, LR^+, UR^+, UR^+, LIP, LOW \} \). This allows us to succinctly represent the two clusters of a matched pair of a (maximal) stripe expression by the sets of their requests and observables, whose size is linear in \(|\varphi|\). In addition, we can assume every (maximal) stripe expression \( E = (L, R) \) to contain pairwise distinct matched pairs \((L(i), R(i))\). From the above, it follows that the length \(|E|\) of any (maximal) stripe expression \( E = (L, R) \) is at most \( 10 \cdot |\varphi| \). At worst, for every pair of distinct indices \( 1 \leq i < j \leq |E| \), if \( L(i), R(i), L(j), \) and \( R(j) \) are clusters, then, for both \( D = UR \) and \( D = UL \), we have \( \text{REQ}_D(L(j)) \subseteq \text{REQ}_D(L(i)) \), \( \text{REQ}_D(R(j)) \subseteq \text{REQ}_D(R(i)) \), and either \( \text{REQ}_D(L(j)) \subseteq \text{REQ}_D(L(i)) \) or \( \text{REQ}_D(R(j)) \subseteq \text{REQ}_D(R(i)) \), and in both shading sequences there exist an atom between any pair of consecutive clusters. Hence, every (maximal) stripe expression can be represented using a polynomial space with respect to \(|\varphi|\).

### 6.4.2 Recursive decompositions of stripes

Roughly speaking, conditions C1–C6 above provide us with a guarantee that the natural spatial interpretation of a stripe expression \( E \) is consistent with view-to-type dependencies. To enforce the fulfillment of the existential requests of the atoms featured by the two shading sequences of \( E \), we further need to introduce a suitable notion of decomposition. We start by dividing a given labeled stripe into a pair of (thinner) adjacent labeled sub-stripes and then we recursively apply the decomposition to every emerging sub-stripe. This yields an infinite tree-shaped decomposition of the initial structure, where each vertex of the tree represents a labeled (sub-)stripe (and, thus, it is associated with a stripe expression) and each edge represents a containment relationship between two labeled (sub-)stripes.

To start with, we introduce a suitable equivalence relation between shading sequences. Two shading sequences \( S \) and \( S' \) are said to be equivalent if

i) every cluster \( S(i) \) of \( S \) (resp., \( S'(i') \) of \( S' \)) is also a cluster of \( S' \) (resp., \( S \));

ii) every atom \( S(i) \) of \( S \) (resp., \( S'(i') \) of \( S' \)) either is an atom of \( S' \) (resp., \( S \)) or it belongs to the two adjacent clusters \( S(i-1) = S(i+1) \) in \( S \) (resp., \( S'(i'-1) = S'(i'+1) \) in \( S' \)).

As an example, the shading sequences \( S = C_1 \ A_1 \ C_1 \ C_2 \) and \( S' = C_1 \ C_1 \ C_2 \) are equivalent, provided that \( A_1 \in C_1 \), while the shading sequences \( S = C_1 \ A_1 \ C_2 \ C_2 \) and \( S' = C_1 \ C_2 \ C_2 \) are not equivalent (unless \( A_1 \in C_1 \) and \( C_1 = C_2 \)).

Decompositions of stripe expressions are defined as follows. Let \( E = (L, R) \) be a stripe expression. A decomposition of \( E \) is any pair of stripe expressions \( (E_1, E_2) \), with \( E_1 = (L_1, R_1) \) and \( E_2 = (L_2, R_2) \), such that the following matching conditions hold:

(M1) \( L_1 \) and \( L \) are equivalent;

(M2) \( R_2 \) and \( R \) are equivalent;

(M3) \( R_1 \) and \( L_2 \) are equivalent.
We say that a matched pair \((L(i), R(i))\) of the stripe expression \(E\) corresponds to a matched pair \((L_1(i_1), R_1(i_1))\) (resp., \((L_2(i_2), R_2(i_2))\)) of the stripe expression \(E_1\) (resp., \(E_2\)) under the decomposition \((E_1, E_2)\) of \(E\) if there exists an index \(1 \leq i_2 \leq |E_2|\) (resp., \(1 \leq i_1 \leq |E_1|\)) such that (i) \(L(i) \equiv L_1(i_1)\), (ii) \(R(i) \equiv R_1(i_1)\), and (iii) \(R_1(i_1) \equiv L_2(i_2)\), where \(\equiv\) denotes either the equality relation \(=\), the membership relation \(\in\), or the containment relation \(\supseteq\) depending on the form of the left and right arguments (namely, whether they are atoms or clusters). As an example, Figure 6.3(c) depicts a decomposition of the stripe expression \(E = (L, R)\), where \(L = C_1 A_2 C_3 A_3 C_3\) and \(R = C_1 A_1 C_3 A_2 C_3\). Note that, under such a decomposition, the matched pair \((C_3, C_1)\) of \(E\) corresponds to the matched pairs \((C_3, C_1), (A_3, A_2'), (C_3, C_3)\) of \(E_1\) and to the matched pairs \((C_1, C_1), (A_2', A_1), (C_3, C_1)\) of \(E_2\). By iteratively applying decompositions, starting from a given stripe expression, one obtains an infinite tree-shaped structure, called decomposition tree.

**Definition 6.4.1.** A decomposition tree is an infinite complete binary labeled tree \(T = (V, E, \downarrow_1, \downarrow_2)\), where
- \(V\) is the set of tree vertices;
- \(\downarrow_1\) and \(\downarrow_2\) are the two successor relations;
- \(E\) is a labeling function associating a stripe expression \(E(v)\) with each \(v \in V\) such that the pair \((E(\downarrow_1(v)), E(\downarrow_2(v)))\) is a decomposition of \(E(v)\).

Note that, for every pair of vertices \(v\) and \(v'\) at the same level of a decomposition tree \(T = (V, E, \downarrow_1, \downarrow_2)\), if \(v'\) is right-adjacent to \(v\) (even without being its sibling) and \(E(v) = (L_v, R_v)\) and \(E(v') = (L_{v'}, R_{v'})\) are the associated stripe expressions, then the sequence \(R_v\) turns out to be equivalent to the sequence \(L_{v'}\).

Let \(T = (V, E, \downarrow_1, \downarrow_2)\) be a decomposition tree. We impose suitable conditions on \(T\) which guarantee that every existential request of every atom featured by a stripe expression \(E(v)\) is eventually fulfilled by an observable of an atom featured by a (possibly different) stripe expression \(E(v')\). Given a stripe expression \(E(v) = (L, R)\), let us denote by \(E(v)[L]\) (resp., \(E(v)[R]\)) its left shading sequence \(L\) (resp., right shading sequence \(R\)). In the following, we consider a generic vertex \(v\) of \(T\) and we look at the right-oriented (i.e., LR- and UL-oriented) requests of the atoms featured by \(E(v)[L]\); symmetrically, we look at the left-oriented (i.e., LL- and UL-oriented) requests of the atoms featured by \(E(v)[R]\).

Let us consider the UL-requests of a left shading sequence \(E(v)[L]\). Given a vertex \(v\) of \(T\), an index \(1 \leq i \leq |E(v)|\), and a subformula \(\alpha \in \text{Req}_{UL}(E(v)[L](i))\) (resp. \(\alpha \in \text{Req}_{UL^+}(E(v)[L](i))\)), we say that the LR-request (resp. UL-request) \(\alpha\) is postponed at position \(i\) of vertex \(v\), if we have \(\alpha \in \text{Req}_{UL}(E(v)[R](i))\);

(f1) postponed at position \(i\) of vertex \(v\), if we have \(\alpha \in \text{Obs}(E(v)[R](i))\) for some index \(i < j \leq |E(v)|\) if \(E(v)[L](i)\) is an atom and for some \(i \leq j < |E(v)|\) if \(E(v)[L](i)\) is a cluster (resp. if we have \(\alpha \in \text{Obs}(E(v)[R](i))\) for some \(i \leq j < |E(v)|\) if \(E(v)[L](i)\) );

(f2) fulfilled at position \(i\) of vertex \(v\), if we have \(\alpha \in \text{Obs}(E(v)[R](i))\) for some index \(i < j \leq |E(v)|\) if \(E(v)[L](i)\) is an atom and for some \(i \leq j < |E(v)|\) if \(E(v)[L](i)\) is a cluster (resp. \(\alpha \in \text{Obs}(E(v)[R](i))\) for some \(i \leq j < |E(v)|\) if \(E(v)[L](i)\) );

(f3) partially fulfilled at position \(i\) of vertex \(v\), if there is an index \(1 \leq i_1 \leq |E(v)|\) such that (i) the LR-request \(\alpha\) is fulfilled at position \(i_1\) of vertex \(\downarrow_1(v)\) and (ii) the matched pair \((E(v)[L](i_1), E(v)[R](i_1))\) of \(E(v)\) corresponds to the matched pair \((E(\downarrow_1(v))[L](i_1), E(\downarrow_1(v))[R](i_1))\) of the decomposition \((E(\downarrow_1(v)), E(\downarrow_2(v)))\) of \(E(v)\).

Similar definitions can be given for the LR-/LR+-requests of a left shading sequence \(E(v)[L]\) and for the UL-/UL+-/-LL-/LL+-requests of a right shading sequence \(E(v)[R]\).

We say that a decomposition tree \(T\) is globally fulfilled if, for every vertex \(v\), every index \(1 \leq i \leq |E(v)|\), and every direction \(D \in \{LR, UL, LR^+, UL^+\}\) (resp., \(D \in \{UL, LL, UL^+, LL^+\}\)), the following conditions hold:

(G1) if \(v\) is the root, then \(\text{Req}_D(E(v)[R](i)) = \emptyset\) (resp., \(\text{Req}_D(E(v)[L](i)) = \emptyset\));

(G2) for every subformula \(\alpha \in \text{Req}_D(E(v)[L](i))\) (resp., \(\text{Req}_D(E(v)[R](i))\)) and every infinite path \(\pi\) that starts at \(v\), there is a vertex \(v'\) in \(\pi\) (possibly \(v' = v\)) such that either \(\alpha \notin \text{Req}_D(E(v'[i])[L](i'))\) (resp., \(\alpha \notin \text{Req}_D(E(v'[i])[R](i'))\)) for all positions \(i'\) of vertex \(v'\) or \(\alpha\) is postponed (F1), fulfilled (F2), or partially fulfilled (F3) at some position \(i'\) of vertex \(v'\).
We are now ready to establish a tree (pseudo-)model property for satisfiable formulas of Cone Logic. The next theorem states that (i) given a globally fulfilled decomposition tree $T$, there is a labeled stripe $P = (1 \times Q, \sigma)$ whose shading coincides with the set of all atoms that are featured by the expressions of $T$ (soundness) and (ii) given a labeled stripe $P = (1 \times Q, \sigma)$, there is a globally fulfilled decomposition tree $T$ whose expressions feature (at least) the types of all points of $P$ (completeness).

**Theorem 6.4.2.** Soundness. For every globally fulfilled decomposition tree $T = (V, E, 1, 2)$, there is a labeled stripe $P = (1 \times Q, \sigma)$ such that

$$\forall i \in V \cup \text{E(i)} \cup \text{E(v)(i)}$$

Completeness. Conversely, for every labeled stripe $P = (1 \times Q, \sigma)$, there is a globally fulfilled decomposition tree $T = (V, E, 1, 2)$ such that

$$\forall i \in V \cup \text{E(i)} \cup \text{E(v)(i)}$$

Proof. We first prove the second claim (completeness) and then the first one (soundness).

**Completeness.** Let $P = (1 \times Q, \sigma)$ be a labeled stripe. Since all dense denumerable linear orders with minimum and maximum elements are pairwise isomorphic, we can assume that I coincides with the subset $\{ \frac{1}{2^n} : i, n \in \mathbb{N}, 0 \leq i \leq 2^n \}$ of the rational numbers. Let $T = (V, 1, 2)$ be the infinite complete binary (unlabeled) tree. We associate with each vertex $v$ of $T$, two values $x_{v, L}$ and $x_{v, R}$ belonging to I as follows. If $v$ is the root of $T$, then we define $x_{v, L} = \min(I) = 0$ and $x_{v, R} = \max(I) = 1$. If $v$ is a vertex of $T^-$ and $v_1 = \bar{1} | v$ and $v_2 = \bar{2} | v$ are its successors, then, assuming that both values $x_{v, L}$ and $x_{v, R}$ are defined, we let $x_{v_1, L} = x_{v, L}$, $x_{v_2, R} = x_{v, R}$, and $x_{v_1, R} = x_{v_2, L} = \frac{x_{v, L} + x_{v, R}}{2}$. Note that the set $X = \{ x_{v, L} : v \in V \} \cup \{ x_{v, R} : v \in V \}$ of the above-defined values actually coincides with I.

We show now how to associate a stripe expression $E(v)$ with each vertex $v$ of $T^-$ in such a way that the resulting decomposition tree is globally fulfilled and the statement of the theorem holds (completeness). Let us consider a vertex $v$ of $T^-$ together with the associated vertical lines $P_{v, L} = (x_{v, L}) \times Q$ and $P_{v, R} = (x_{v, R}) \times Q$. We define an equivalence relation $\sim$ over the set of pairs of points $(p, q)$ and $(p', q')$, with $p = (x_{v, L}, y)$, $q = (x_{v, R}, y)$, $p' = (x_{v, L}, y')$, and $q' = (x_{v, R}, y')$, such that $(p, q) \sim (p', q')$ if $\exists \mathcal{D} \subseteq \mathcal{D}(p) = \mathcal{D}(q)$, $\equiv \mathcal{D}(q') = \mathcal{D}(p')$, and $\equiv \mathcal{D}(q') = \equiv \mathcal{D}(p')$. We then consider the coarsest partition of the region $P_{v, L} \cup P_{v, R}$ into subregions $P_{v, L, \bar{i}}$ with $i$ ranging over a finite set $\{1, \ldots, k\}$ of indices, that respects the equivalence $\sim$ and that satisfies the following properties:

- for every index $1 \leq i \leq k$, $P_{v, L, \bar{i}} \subseteq P_{v, L}$ and $P_{v, i, R} \subseteq P_{v, R}$;
- for every index $1 \leq i \leq k$, either both $P_{v, L, \bar{i}}$ and $P_{v, i, R}$ are singletons or they are open vertical segments;
- for every pair of indices $1 \leq i < j \leq k$ and every pair of points $p = (x_{v, L}, y) \in P_{v, i, L}$ and $p' = (x_{v, L}, y') \in P_{v, i, L}$ (resp., $q = (x_{v, R}, y) \in P_{v, i, R}$ and $q' = (x_{v, R}, y') \in P_{v, i, R}$), $y < y'$ holds.

Next, we define a (non-maximal) stripe expression $E_v = (L_v, R_v)$, with $|E_v| = k$, by specifying, for every index $1 \leq i \leq k$, the two components $L_v(i)$ and $R_v(i)$ of its $i$-th matched pair. If both regions $P_{v, L, \bar{i}}$ and $P_{v, i, R}$ are singletons of the form $\{p_{v, i, L}\}$ and $\{p_{v, i, R}\}$, respectively, then we let $L_v(i)$ be the atom $\equiv \mathcal{D}(p_{v, i, L})$ and $R_v(i)$ be the atom $\equiv \mathcal{D}(p_{v, i, R})$. Otherwise, if both regions $P_{v, i, L}$ and $P_{v, i, R}$ are open vertical segments, then we let $L_v(i)$ be the cluster $\equiv \mathcal{D}(P_{v, i, L})$ and $R_v(i)$ be the cluster $\equiv \mathcal{D}(P_{v, i, R})$. Finally, we let $E(v)$ be the maximal stripe expression that generalizes $E_v$, that is, $E(v) = \max(E : E \geq E_v)$, and we define the decomposition tree $T = (V, E, 1, 2)$. By construction, we have that the stripe expressions of $T$ feature at least the types of all points of $P$. Moreover, the stripe expressions of $T$ satisfy Conditions M1-M3. It thus remains to prove that
the decomposition tree $\mathcal{T}$ is consistent and globally fulfilled. Let us consider a generic vertex $v$ of $\mathcal{T}$. First of all, we show that the stripe expression $E(v)$ is consistent, namely, it satisfies Conditions C1-C6. As an example, if $i$ is the position of an atom $E(v)[L][i]$ in the left sequence of $E(v)$, then, by construction, the region $P_{v,i,L}$ must be a singleton of the form $\{p_{v,i,L}\}$ and for every index $1 \leq j < i$, there must exist a point $q \in P_{v,i,L}$ such that $q \in LR(p)$, whence, by view-to-type dependency, we obtain $E(v)[L][i] \leq_{\alpha} E(v)[R][i]$. This proves part of Condition C6. Similar arguments can be used to prove the rest of Condition C6 and the other consistency conditions. We conclude the proof by showing that the decomposition tree $\mathcal{T}$ is globally fulfilled. By construction, the root $v_0$ of $\mathcal{T}$ satisfies $\mathcal{R}eq_{\mathcal{T}}(E(v)[R][i]) = 0$ (resp., $\mathcal{R}eq_{\mathcal{T}}(E(v)[L][i]) = 0$) for all indices $1 \leq i \leq |E(v)|$ and all directions $D \in \{LR, UR, LR^+, UR^+\}$ (resp., $D \in \{LL, UL, LL^+, UL^+\}$). As for the fulfillment properties F1-F3 satisfied by a generic vertex $v$ of $\mathcal{T}$, we consider the set of propositions $\mathcal{R}eq_{\mathcal{T}}(E(v)[L][i])$, with $1 \leq i \leq |E(v)|$ (similar arguments can be used for the other cases). By construction, there exists a point $p = (x_{v,R}, y)$ in the region $P_{v,i,L}$ whose type features the subformula $\text{Prop}$, that is, $\alpha \in \mathcal{R}eq_{\mathcal{T}}(\text{Prop}(p))$. By definition of type, there exists a point $q \in \mathcal{UR}(p)$ such that $q \in \mathcal{Obs}(\text{Prop}(q))$. Now, by construction, $q$ must be a point of the form $(x_{v',R}, y')$, with $v'$ being a vertex of $\mathcal{T}$, $x_{v',R} > x_{v,L}$, and $y' > y$. We distinguish between the following three cases:

1. $x_{v,R} < x_{v',R}$;
2. $x_{v,R} = x_{v',R}$;
3. $x_{v,R} > x_{v',R}$.

In the first case ($x_{v,R} < x_{v',R}$), we denote by $r$ the point $(x_{v,R}, y)$ (recall that $p = (x_{v,L}, y)$) and we observe that $r \in P_{v,R}$ and $q \in \mathcal{UR}(r)$. Thus, by view-to-type dependency, $\alpha$ belongs to the set of requests $\mathcal{R}eq_{\mathcal{T}}(\text{Prop}(r))$ and hence, by construction, to the set $\mathcal{R}eq_{\mathcal{T}}(E(v)[L][i])$ as well. This basically implies that the UR-request $\alpha$ is postponed at position $i$ of vertex $v$. In the second case ($x_{v,R} = x_{v',R}$), we immediately obtain that the UR-request $\alpha$ is fulfilled at position $i$ of vertex $v$. Finally, we consider the third case ($x_{v,R} > x_{v',R}$). By construction, we know that $v'$ is a proper descendant of $v$. Now, we assume, by way of contradiction, that there is an infinite path $\pi$ that starts at $v$ and such that, for every vertex $v''$ along $\pi$, the UR-request $\alpha$ is neither postponed nor fulfilled, nor partially fulfilled. We would thus have $\alpha \in \mathcal{R}eq_{\mathcal{T}}(E(v''[L][i]))$, $\alpha \notin \mathcal{R}eq_{\mathcal{T}}(E(v''[R][i]))$, and $\alpha \notin \mathcal{Obs}(E(v''[R][i]))$ for all vertices $v''$ along $\pi$ and all indices $1 \leq i \leq |E(v'')|$. In its turn, this implies the existence of an infinite $2$-directed chain of stripes $I_{v''} \times Q$, with $v''$ ranging over the set of all vertices of $\pi$, each one containing the point $q$. However, we know, by construction, that a boarder of one of these stripes $I_{v''} \times Q$ intersects the point $q$, which is against the hypothesis about the infinite path $\pi$. This basically proves that the decomposition tree $\mathcal{T}$ is globally fulfilled.

**Soundness.** Let $\mathcal{T} = (V, E, \downarrow_1, \downarrow_2)$ be a globally fulfilled decomposition tree. We associate with each vertex $v$ of $\mathcal{T}$ a stripe $I_v \times Q$, where $I_v = [x_{v,L}, x_{v,R}]$ is a closed interval and $x_{v,L}$ and $x_{v,R}$ are defined in the following way. If $v$ is the root of $\mathcal{T}$, then we let $x_{v,L} = 0$ and $x_{v,R} = 1$. If $v$ is a vertex of $\mathcal{T}$ and $v_1 = \downarrow_1(v)$ and $v_2 = \downarrow_2(v)$ are its successors, then, assuming that both values $x_{v,L}$ and $x_{v,R}$ are defined, we let $x_{v,L} = x_{v_1,L}$, $x_{v_2,R} = x_{v,R}$, and $x_{v,R} = x_{v,L} = \frac{x_{v_1,L} + x_{v_2,L}}{2}$. We denote by $X$ the set of all the above-defined values, namely, $X = \{x_{v,L} : v \in V\} \cup \{x_{v,R} : v \in V\} = \{\frac{i}{2^n} : i, n \in \mathbb{N}, 0 \leq i \leq 2^n\}$. The set $X$ is properly included in the closed interval $[0, 1]$. However, since all dense denumerable linear orders with minimum and maximum elements are pairwise isomorphic, the linear order of $X$ is isomorphic to that of $[0, 1]$. It easily follows that for every value $x \in [0, 1]$ (resp., $x \in (0, 1)$), there is a vertex $v \in V$ such that $x_{v,L}$ (resp., $x_{v,R}$) corresponds to $x$ under some fixed isomorphism between $X$ and $[0, 1]$. Thus, by a slight abuse of terminology, we call the structure $X \times Q$ a stripe.

We show now how to associate a valuation $\sigma(p)$ up to $\text{Prop}$ with every point $p \in X \times Q$ in such a way that the resulting labeled stripe $P = (X \times Q, \sigma)$ satisfies the statement of the theorem (soundness). As a first step, we associate with each vertex $v$ of $\mathcal{T}$ and each position $1 \leq i \leq |E(v)|$ either a singleton or an open interval $I_{v,i}$ in such a way that the following properties are satisfied:

- if $E(v)[L][i]$ and $E(v)[R][i]$ are atoms (resp., clusters), then $I_{v,i}$ is a singleton (resp., an open interval);
for every $1 \leq i < j \leq |E(v)|$, every $y \in J_{v,i}$, and every $y' \in J_{v,j}$, $y < y'$, and thus $J_{v,1} < J_{v,2} < \ldots < J_{v,|E(v)|}$;

$\bigcup_{1 \leq i \leq |E(v)|} J_{v,i} = \mathbb{Q}$.

Given a vertex $v$ of $\mathcal{T}$ and a position $1 \leq i \leq |E(v)|$, we denote by $P_{v,i,L}$ (resp., $P_{v,i,R}$) the region $\{x_{v,L}\} \times J_{v,i}$ (resp., $\{x_{v,R}\} \times J_{v,i}$), which is either a singleton or an open vertical segment of the rational plane. By exploiting the density of the rational numbers and the matching conditions by Definition 6.4.1 (see Conditions M1-M3), we can assume, without loss of generality, that for every pair of overlapping regions $P_{v,i,d}$ and $P_{v',i',d'}$, with $v, v' \in \mathcal{V}$, $1 \leq i \leq |E(v)|$, $1 \leq i' \leq |E(v')|$, and $d, d' \in [L, R]$, one of the following conditions holds:

1. both $J_{v,i}$ and $J_{v',i'}$ are singletons and the corresponding atoms $E(v)[d](i)$ and $E(v')[d'](i')$ coincide;
2. both $J_{v,i}$ and $J_{v',i'}$ are open intervals and the corresponding clusters $E(v)[d](i)$ and $E(v')[d'](i')$ coincide;
3. $J_{v,i}$ is a singleton, $J_{v',i'}$ is an open interval, and the atom $E(v)[d](i)$ belongs to the cluster $E(v')[d'](i')$;
4. $J_{v',i'}$ is a singleton, $J_{v,i}$ is an open interval, and the atom $E(v')[d'](i')$ belongs to the cluster $E(v)[d](i)$.

We now associate a suitable labeling function $\sigma_{v,i,d}$ with each region $P_{v,i,d}$. Such a labeling function is defined by a case analysis as follows. If $P_{v,i,d}$ consists of a single point $p = \{x_{v,d}, y_{v,i}\}$, then $E(v)[d](i)$ is an atom $A$ and $\sigma_{v,i,d}(p)$ is defined as the set $\text{Var}(A)$ of all positive literals, that is, propositional variables, in $A$. If $P_{v,i,d}$ is an open vertical segment of the form $\{x_{v,d}\} \times J_{v,i}$, $E(v)[d](i)$ is a cluster $C$ and $\sigma_{v,i,d}$ is defined as the shuffle\footnote{The shuffle of a set $Y$ over a dense (denumerable) linearly ordered set $X$ is the (unique up to isomorphism) function $\sigma : X \to Y$ that satisfies (i) $\sigma(X) \subseteq Y$ and (ii) for every $x < x' \in X$ and every $y \in Y$, $\sigma(x^x) = y$ for some $x < x' < x'$. By a slight abuse of terminology, we use shuffles of sets of valuations over open vertical segments, as the latter ones can be viewed as dense (denumerable) linearly ordered sets.} of the set $\text{Var}(C) = \{\text{Var}(A) : A \in C\}$ over the open vertical segment $P_{v,i,d}$. In virtue of basic properties of shuffles, we can assume that for every pair of labeling functions $\sigma_{v,i,d}$ and $\sigma_{v',i',d'}$ and every point $p$ belonging to the intersection $P_{v,i,d} \cap P_{v',i',d'}$ of the two domains, $\sigma_{v,i,d}(p) = \sigma_{v',i',d'}(p)$ holds.

The above arguments allow us to define a labeled stripe $\mathcal{P} = (X \times \mathbb{Q}, \sigma)$, where $\sigma$ is the union of the labeling functions $\sigma_{v,i,d}$, for all vertices $v$ of $\mathcal{T}$, all positions $1 \leq i \leq |E(v)|$, and all directions $d \in [L, R]$. Finally, by exploiting an argument similar to that we used in the completeness proof, one can show that the shading of $\mathcal{P}$ coincides with the set of all atoms that are featured by the stripe expressions of the decomposition tree $\mathcal{T}$.

\[ \square \]

### 6.5 Reducing Cone Logic to a proper fragment of CTL

In this section we briefly describe a decision procedure that solves the satisfiability problem for Cone Logic taking advantage of the tree (pseudo-)model property stated in Section 6.4. According to such a property, the problem of establishing whether or not a Cone Logic formula $\varphi$ is satisfiable can be reduced to the problem of checking the existence of a globally fulfilled decomposition tree $\mathcal{T}$ that features a ($\varphi$-)atom $A$ such that $\varphi \in A$. The effectiveness of such an approach stems from the fact that the properties that characterize a globally fulfilled decomposition tree can be expressed in a proper fragment of CTL. The satisfiability problem for Cone Logic can thus be decided in (at most) exponential time [44]. Given the state of the art of the decision procedures for CTL, deciding the satisfiability problem for Cone Logic turns out to be quite efficient from a practical point of view. In the following, we show that the satisfiability problem for Cone Logic is actually in PSPACE. In the next section, we will prove that the PSPACE complexity bound is strict, namely, that the satisfiability problem for Cone Logic is PSPACE-hard.
6.5. Reducing Cone Logic to a proper fragment of CTL

Theorem 6.5.1. The satisfiability problem for Cone Logic, interpreted over the rotational plane, is in PSPACE.

sketch. We first show how to reduce the satisfiability problem for a Cone Logic formula \( \phi \) to the satisfiability problem for a suitable CTL formula \( \overline{\phi} \), which is a conjunction of CTL formulas of the forms \( \lambda, \text{AG} \lambda, \text{EF} \lambda, \text{AG EX} \lambda, \text{AG} \delta, \) and \( \text{AG} (\lambda \rightarrow \text{AF} \delta) \), where \( \lambda \) is a propositional formula and \( \delta \) is a CTL formula that uses the modal operator \( \text{AX} \) in a positive way only (and it has no occurrences of other modal operators). Let us call these formulas basic CTL formulas.

To start with, we introduce three distinguished propositional variables, say, 0, 1, and 2, to encode the two successor relations \( \downarrow_1 \) and \( \downarrow_2 \) of a decomposition tree \( T \) in a labeled tree structure \( \Sigma \). For each vertex \( v \) of \( T \), we associate either 0, 1, or 2 with \( v \) depending on whether \( v \) is the root, \( v = \downarrow_1 (u) \), or \( v = \downarrow_2 (u) \) for some parent vertex \( u \). Such a labeling can be enforced by means of a suitable conjunction of basic CTL formulas over the signature \( \{0, 1, 2\} \) (see below). Next, the stripe expressions associated with \( T \) vertices can be encoded as follows. Since the number of shading sequences can be exponential in \( |\phi| \), we need to encode one by one the elements that belong to each atom featured by each shading sequence. To this end, we introduce a new set of basic CTL formulas over the signature \( \{0, 1, 2\} \) (see below). Theorem 6.5.1. The satisfiability problem for Cone Logic, interpreted over the rotational plane, is in PSPACE.

Theorem 6.5.1. The satisfiability problem for Cone Logic, interpreted over the rotational plane, is in PSPACE.

Theorem 6.5.1. The satisfiability problem for Cone Logic, interpreted over the rotational plane, is in PSPACE.

Theorem 6.5.1. The satisfiability problem for Cone Logic, interpreted over the rotational plane, is in PSPACE.
contains only positive occurrences of the modal operators AG, AF, and AX, we can turn \( \varphi_{\text{path}} \) into an equivalent LTL formula \( \varphi_{\text{path}}^{\LTL} \) by replacing all occurrences of AG, AF, and AX by G, F, and X, respectively. Formally, we have that for any labeled tree structure \( T \), \( \varphi \) holds at the root of \( T \) if and only if (i) \( \varphi_{\text{tree}} \) and \( \varphi_{\text{init}} \) hold at the root of \( T \) and (ii) \( \varphi_{\text{path}}^{\LTL} \) holds along all infinite paths of \( T \). By taking advantage of the structure of \( \varphi_{\text{path}}^{\LTL} \), (no G operator is nested into an F operator), it is possible to show that there exists a deterministic Büchi automaton \( A_{\text{path}} \), which can be computed in polynomial time\(^3\), that recognizes the \( \omega \)-language of all linear models of \( \varphi_{\text{path}}^{\LTL} \). Given the automaton \( A_{\text{path}} \) over the input alphabet \( \{0, 1, 2\} \times \mathcal{P}(\Sigma) \), we build a non-deterministic Büchi automaton \( A_{\text{path}}^3 \) that recognizes the projection language \( \pi_{0,1,2}(\mathcal{L}^{\omega}(A_{\text{path}})) \). We have that \( A_{\text{path}}^3 \) recognizes the \( \omega \)-language \( \{0\} \cdot (1, 2)^\omega \) if and only if there exists a labeled tree structure \( T \) that satisfies both \( \varphi_{\text{tree}} \) and \( \varphi_{\text{path}}^{\LTL} \). Since the inclusion problem for regular \( \omega \)-languages is in PSPACE \(^{53}\), this gives a procedure that decides, in polynomial space, whether both formulas \( \varphi_{\text{tree}} \) and \( \varphi_{\text{path}} \) hold at the root of some labeled tree structure \( T \). Verifying whether \( \varphi_{\text{init}} \) holds at the root of \( T \) as well amounts to solve a reachability problem over a slightly modified version of the non-deterministic Büchi automaton \( A_{\text{path}}^3 \).

6.6 Cone Logic and interval temporal logics

In this section, we prove that Cone Logic subsumes an interesting interval temporal logic, called \( \mathcal{B}\mathcal{D}\mathcal{D}\mathcal{L}\mathcal{L} \)-logic, which comprises six modal operators \( (\langle D \rangle, \langle \neg D \rangle, \langle B \rangle, \langle \neg B \rangle, \langle L \rangle, \langle \neg L \rangle) \). Intuitively, these operators quantify over sub-intervals, super-intervals, begin intervals, begun-by intervals, future intervals and past intervals. From now on, we assume that the underlying temporal domain is (homeomorphic to) the linear ordering \( (\mathbb{Q}, <) \) of the rational numbers and that intervals are non-singleton closed convex subsets of such an ordering, namely, sets of the form \( [x, y) = \{z \in \mathbb{Q} : x \leq z < y \} \), where \( x, y \in \mathbb{Q} \) and \( x < y \). We shortly denote by \( \mathbb{I} \) the set of all intervals over \( (\mathbb{Q}, <) \).

We recall the semantics of this 6 interval operators \( (\langle D \rangle, \langle \neg D \rangle, \langle B \rangle, \langle \neg B \rangle, \langle L \rangle, \langle \neg L \rangle) \). Let \( I = [x, y) \) and \( I' = [x', y') \) be two intervals. If \( x < x' < y' < y \), then we say that \( I' \) is a (strict) sub-interval of \( I \), or, conversely, that \( I \) is a (strict) super-interval of \( I' \). Similarly, if \( x' = x \) and \( y' < y \), we say that \( I' \) begins \( I \), conversely, \( I \) is begun by \( I' \). Finally we say that \( I' \) is in the future of \( I \) if \( x' > y \), or, conversely, \( I \) is in the past of \( I' \).

As for the semantics of \( \mathcal{B}\mathcal{D}\mathcal{D}\mathcal{L}\mathcal{L} \)-logic, let \( \mathcal{P} = (\mathbb{I}, \sigma) \) be an interval structure, where \( \sigma \) is a valuation function that maps intervals in \( \mathbb{I} \) to sets of propositional variables. The formulas of \( \mathcal{B}\mathcal{D}\mathcal{D}\mathcal{L}\mathcal{L} \)-logic are built up from propositional variables using the boolean connectives and the modal operators \( (\langle D \rangle, \langle \neg D \rangle, \langle B \rangle, \langle \neg B \rangle, \langle L \rangle, \langle \neg L \rangle) \), with the obvious semantics (for instance, \( \mathcal{P}, I \models (D)\varphi \) holds iff there is a sub-interval \( I' \) of \( I \) such that \( \mathcal{P}, I' \models \varphi \)).

In [39] Lodaya conjectured the undecidability of the satisfiability problem for the fragment of \( \mathcal{B}\mathcal{D}\mathcal{D}\mathcal{L}\mathcal{L} \)-logic that features the two modal operators \( (\langle D \rangle, \langle \neg D \rangle) \) only when interpreted over various classes of linear orderings. Here, we partially disprove such a conjecture by showing that formulas of \( \mathcal{B}\mathcal{D}\mathcal{D}\mathcal{L}\mathcal{L} \)-logic, interpreted over the rational line, can actually be translated into equi-satisfiable formulas of Cone Logic. Such a translation exploits the fact that there exists a natural bijection between intervals \( I = [x, y) \) in \( \mathbb{I} \) and points \( p = (x, y) \), with \( x < y \), of the rational plane (hereafter, we call these points interval points). Moreover, the region of all and only the interval points of the rational plane can somehow be described by a suitable formula of Cone Logic. More precisely, let \( T, \bot, \pi \) be three fresh propositional variables and let \( \varphi_\pi \) be the following formula of Cone Logic:

\[^3\text{First, we turn each conjunct of } \varphi_{\text{path}}^{\LTL} \text{ into an equivalent deterministic Büchi automaton and then we compute the product automaton for the whole formula } \varphi_{\text{path}}^{\LTL} \text{. The resulting automaton has size polynomial in } \varphi_{\text{path}}^{\LTL} \text{, provided that the transition labels are symbolically represented by means of suitable propositional formulas over the signature } \{0, 1, 2\} \cup \Sigma.\]
Consider now a labeled rational plane \( P = (\mathbb{P}, \sigma) \) that satisfies \( \Psi_{\pi} \). We can partition \( P \) in three regions, namely, (i) the region \( \top_P \) of all \( \top \)-labeled points, (ii) the region \( \bot_P \) of all \( \bot \)-labeled points, and (iii) the region \( \pi_P \) of all \( \pi \)-labeled points (see Figure 6.4). The region \( \pi_P \) has the form of a \("thin"\) oriented trajectory inside the rational plane such that, for every pair of points \( p, q \in \pi_P \), there exists another point \( r \in \pi_P \) such that either \( r \in \text{UR}(p) \) and \( q \in \text{UR}(r) \), or \( r \in \text{LL}(p) \) and \( q \in \text{LL}(r) \). Even though we cannot claim that \( \pi_P \) coincides with the diagonal \( \{(x, x) : x \in \mathbb{Q}\} \) and \( \top_P \) coincides with the set of all interval-points of the rational plane, we can prove the following proposition.

**Proposition 6.6.1.** For every formula \( q \) of Cone Logic and every labeled rational plane \( P = (\mathbb{P}, \sigma) \) that satisfies \( q_P = q \wedge \Psi_{\pi} \), there is a labeled rational plane \( P' = (\mathbb{P}, \sigma') \) that satisfies \( q \) such that (i) the \( \pi \)-labeled region \( \pi_P \) coincides with the diagonal \( \{(x, x) : x \in \mathbb{Q}\} \) and (ii) the \( \top \)-labeled region \( \top_P \) coincides with the set of all interval-points of the rational plane.

**Proof.** Let \( P = (\mathbb{P}, \sigma) \) be a model of the formula \( q_P = q \wedge \Psi_{\pi} \). Without loss of generality, we can assume that for every \( x \in \mathbb{Q} \), there is a \( \pi \)-labeled point of the form \( p = (x, y) \) with \( y \in \mathbb{Q} \) (this follows easily from Theorem 6.4.2). We can thus view the region \( \pi_P \) as the graph of a strictly increasing function \( f_{\pi} : \mathbb{Q} \rightarrow \mathbb{Q} \) such that, for every point \( p = (x, y) \) \( \pi \in \sigma(p) \) if and only if \( f_{\pi}(x) = y \). Thus, we can denote by \( f_{\pi}^{-1} \) the inverse of \( f_{\pi} \), which is a strictly increasing function as well, and we can introduce the (monotone) transformation \( t \) that maps any point \( p = (x, y) \) to the point \( t(p) = (x, f_{\pi}^{-1}(y)) \). We then exploit such a transformation to define a new labeling function \( \sigma' \) as follows: for every point \( p \), we let \( \sigma'(p) = \sigma(t(p)) \). By definition of \( t \), the resulting structure \( P' = (\mathbb{P}, \sigma') \) is homeomorphic to \( P \) and hence it also satisfies the formula \( q_P \). Moreover, on construction, the region \( \pi_P \) coincides with the diagonal \( \{(x, x) : x \in \mathbb{Q}\} \) and, similarly, the region \( \top_P \) coincides with the set of all interval-points of the rational plane. \( \square \)

Proposition 6.6.1 yields a straightforward translation of any given formula \( q \) of \( \mathcal{BBBDLLLL} \)-logic into an equi-satisfiable formula \( q' \) of Cone Logic, which is obtained by first replacing \( q \) every occurrence of the subformula \( \langle D \rangle \alpha \) (resp. \( \langle D \rangle \alpha \), \( \langle B \rangle \alpha \), \( \langle L \rangle \alpha \) and \( \langle \bot \rangle \alpha \)) with the formula \( \bigodot(T \wedge \alpha) \) (resp. \( \bigodot(T \wedge \alpha) \), \( \bigodot(T \wedge \alpha) \), \( \bigodot(T \rightarrow \bigodot(T \wedge \alpha)) \), \( \bigodot^+(T \rightarrow \bigodot(T \wedge \alpha)) \)) and then adding the conjunct \( \Psi_{\pi} \) to the resulting formula. Since the translation of the operators \( \langle D \rangle /\langle \bot \rangle \) and \( \langle B \rangle /\langle \bot \rangle \) is intuitive we enclose the encoding of \( \langle \bot \rangle \) operator in the cone-logic. As we can see in Figure 6.5 if a point \( p = (x, y) \) satisfies \( P, p \models \models \bigodot^+(T \rightarrow \bigodot(T \wedge \alpha)) \) and \( p \models T \) we have that all the points \( p' = (x', y') \) with \( p' \neq p \), \( x \leq x' \), \( y' \leq y \), \( p \models T \) satisfies \( P, p' \models \bigodot(T \wedge \alpha) \). In particular the point \( p'' = (y, y) \) satisfies \( P, p'' \models \bigodot^+(T \wedge \psi) \) and we have that there exists a point \( p''(x'', y'') \) such that \( y < x'' \leq y'' \), \( p''(x'', y'') \models T \) and \( P, p'' \models \psi \) and by translation \( p'' \) is in the future of \( p \). On the other hand if \( P, p \not\models \bigodot^+(T \wedge \bigodot(L \rightarrow \neg \psi)) \) we have that there exists a point \( p' = (x', y') \) with \( x \leq x' \leq y' \leq y \) (red triangle in Figure 6.6) and \( p \not\models p' \) for which \( P, p' \models T \wedge \bigodot(L \rightarrow \neg \psi) \) and since \( P, p' \models T \) we have \( P, p' \models \bigodot(L \rightarrow \neg \psi) \) then we have that either \( p' = (y, y) \) or \( (y, y) \in \text{UR}(p') \). The first case immediately impose that
for every point \( p'' = (x'', y'') \) with \( x'' \leq y'' \) and \( x'' > y \) satisfies \( \forall \, p'' \models \neg \psi \). The second case imposes that \( \forall \, (y, y) \models (\top \lor \neg \psi) \) and we can immediately reduce to ourselves the first case.

Taking advantage of such a translation and of the decision procedure described in Section 6.5, we immediately obtain that the satisfiability problem for \( B\bar{B}D\bar{D}L \)-logic is in PSPACE. As a matter of fact, this subsumes previous results from [12]. Moreover, from [56] we know that the satisfiability problem for \( D \)-logic, that is, the interval temporal logic that features the subinterval operator \( \langle D \rangle \) only, and hence that for \( B\bar{B}D\bar{D}L \)-logic, is PSPACE-hard. Summing up, we have the following corollary.

**Corollary 6.6.2.** The satisfiability problem for Cone Logic, interpreted over the rational plane, and that for \( B\bar{B}D\bar{D}L \)-logic, interpreted over the rational line, are PSPACE-complete.

### 6.7 Conclusions

We would like to conclude by mentioning some natural generalizations of our work. First, we may consider various possible extensions of Cone Logic. For instance, we may think of evaluating formulas of (extended) Cone Logic over multi-dimensional spaces (in general, \( 2^n \) distinct cone-shaped directions exist in a space with \( n \) dimensions) and/or to partition the two-dimensional space
into more than four cone-shaped cardinal directions (the same for higher-dimensional spaces). In all such cases, we believe it possible to generalize the achieved results in a rather natural way, preserving the tree pseudo-model property of the logic and, possibly, the PSPACE-completeness of its satisfiability problem. Further generalizations envisage the use of region-based spatial logics. As an example, the correspondence between intervals over the rational line and points over the rational plane can be lifted to higher-dimensional objects, proving, for instance, that a suitable spatial logic based on rectangular regions, that is, 2-dimensional intervals, is subsumed by a 4-dimensional point-based modal logic very similar to Cone Logic. This establishes an interesting bridge between Cone Logic and modal logics of topological relations. Finally, it is worth studying the satisfiability problem for Cone Logic, and, similarly, for BDDLL-logic, interpreted over different (classes of) structures, e.g., the infinite discrete grid or the Euclidean plane. Even though we expect the satisfiability problem to remain decidable, radically different approaches might be necessary to cope with spaces having discrete or Euclidean topologies.
In this chapter, we focus our attention on the product logic $\text{ABBB}$, obtained from the join of $\text{BB}$ and $A$ (the case of $\text{ABB}$ is fully symmetric), interpreted over the linear order $\mathbb{N}$ of the natural numbers (or a finite prefix of it). The decidability of $\text{BB}$ can be proved by translating it into the point-based propositional temporal logic of linear time LTL with temporal modalities $F$ (sometime in the future) and $P$ (sometime in the past), which has the finite (pseudo-)model property and is decidable, e.g., [30]. In general, such a reduction to point-based temporal logics does not work: formulas of interval temporal logics are evaluated over pairs of points and translate into binary relations. For instance, this is the case with $A$. Unlike the case of $\text{BB}$, when dealing with $A$ one cannot abstract away from the left endpoint of intervals, as contradictory formulas may hold over intervals with the same right endpoint and a different left endpoint. The decidability of $A\overline{A}$, and thus that of its fragment $A$, over various classes of linear orderings has been proved by Bresolin et al. by reducing its satisfiability problem to that of the two-variable fragment of first-order logic over the same classes of structures [13], whose decidability has been proved by Otto in [52]. Optimal tableau methods for $A$ with respect to various classes of interval structures can be found in [18, 21]. A decidable metric extension of $A$ over the natural numbers has been proposed in [22].

$\text{ABBB}$ retains the simplicity of its constituents $\text{BB}$ and $A$, but it improves a lot on their expressive power (as we shall show, such an increase in expressiveness is achieved at the cost of an increase in complexity). First, it allows one to express assertions that may be true at certain intervals, but at no subinterval of them, such as the conditions of accomplishment. Moreover, it makes it possible to easily encode the until operator of point-based temporal logic (this is possible neither with $\text{BB}$ nor with $A$). Finally, meaningful metric constraints about the length of intervals can be expressed in $\text{ABBB}$, that is, one can constrain an interval to be at least (resp., at most, exactly) $k$ points long. We prove the decidability of $\text{ABBB}$ interpreted over $\mathbb{N}$ by providing a small model theorem based on an original contraction method. To prove it, we take advantage of a natural (equivalent) interpretation of $\text{ABBB}$ formulas over grid-like structures based on a bijection between the set of intervals over $\mathbb{N}$ and (a suitable subset of) the set of points of the $\mathbb{N} \times \mathbb{N}$ grid. In addition, we prove that the satisfiability problem for $\text{ABBB}$ is EXPSPACE-complete (that for $A$ is NEXPTIME-complete). In the proof of hardness, we use a reduction from the exponential-corridor tiling problem.

The chapter is organized as follows. In Section 7.1 we introduce $\text{ABBB}$. In Section 7.2, we prove the decidability of its satisfiability problem. We first describe the application of the contraction method to finite models and then we generalize it to infinite ones. In Section 7.3 we deal with computational complexity issues.

### 7.1 The interval temporal logic $\text{ABBB}$

In this section, we briefly introduce syntax and semantics of the logic $\text{ABBB}$, which features three modal operators $A$, $B$, and $B$ corresponding to the three Allen’s relations $A$ (“meets”), $B$ (“begins”), and $B$ (“begun by”), respectively. We show that $\text{ABBB}$ is expressive enough to capture
the notion of accomplishment, to define the standard until operator of point-based temporal logics, and to encode metric conditions. Then, we introduce the basic notions of atom, type, and dependency. We conclude the section by providing an alternative interpretation of AB\overline{B} over labeled grid-like structures.

### 7.1.1 Syntax and semantics

Given a set \( \mathcal{P} \) of propositional variables, formulas of AB\overline{B} are built up from \( \mathcal{P} \) using the boolean connectives \( \neg \) and \( \lor \) and the unary modal operators \( \langle A \rangle \), \( \langle B \rangle \), \( \langle \overline{B} \rangle \). As usual, we shall take advantage of shorthands like \( \varphi_1 \land \varphi_2 = \neg \neg \varphi_1 \lor \neg \neg \varphi_2 \), \( [A] \varphi = \neg \langle A \rangle \neg \varphi \), \( [B] \varphi = \neg \langle B \rangle \neg \varphi \), \( T = p \lor \neg p \), and \( \bot = p \land \neg p \), with \( p \in \mathcal{P} \). Hereafter, we denote by \( |\varphi| \) the size of \( \varphi \).

We interpret formulas of AB\overline{B} in interval temporal structures over natural numbers endowed with the relations “meets”, “begins”, and “begun by”. Precisely, we identify any given ordinal \( \alpha \leq \omega \) with the prefix of length \( \alpha \) of the linear order of the natural numbers and we accordingly define \( \mathbb{N}^\alpha \) as the set of all non-singleton closed intervals \([x,y]\), with \( x, y \in \mathbb{N} \) and \( x < y \). For any pair of intervals \([x,y],[x',y']\) \( \in \mathbb{N}^\alpha \), the Allen’s relations “meets” A, “begins” B, and “begun by” \( \overline{B} \) are defined as follows (note that \( \overline{B} \) is the inverse relation of B):

- **“meets” relation**: \([x,y] \ A [x',y']\) iff \( y = x'\);
- **“begins” relation**: \([x,y] \ B [x',y']\) iff \( x = x'\) and \( y < y'\);
- **“begun by” relation**: \([x,y] \ \overline{B} [x',y']\) iff \( x = x'\) and \( y < y'\).

Given an interval structure \( S = (\mathbb{N}, A, B, \overline{B}, \sigma) \), where \( \sigma : \mathbb{N}^\alpha \to \mathcal{P}(\mathcal{P}) \) is a labeling function that maps intervals in \( \mathbb{N}^\alpha \) to sets of propositional variables, and an initial interval I, we define the semantics of an AB\overline{B} formula as follows:

- \( S, I \models \alpha \iff \alpha \in \sigma(I) \), for any \( \alpha \in \mathcal{P} \);
- \( S, I \models \neg \varphi \iff S, I \not\models \varphi \);
- \( S, I \models \varphi_1 \lor \varphi_2 \iff S, I \models \varphi_1 \) or \( S, I \models \varphi_2 \);
- for every relation \( R \in \{A, B, \overline{B}\} \), \( S, I \models (R) \varphi \iff \) there is an interval \( J \in \mathbb{N}^\alpha \) such that \( I R J \) and \( S, J \models \varphi \).

Given an interval structure \( S \) and a formula \( \varphi \), we say that \( S \) satisfies \( \varphi \) if there is an interval \( I \) in \( S \) such that \( S, I \models \varphi \). We say that \( \varphi \) is satisfiable if there exists an interval structure that satisfies it. We define the satisfiability problem for AB\overline{B} as the problem of establishing whether a given AB\overline{B}-formula \( \varphi \) is satisfiable.

We conclude the section with some examples that account for AB\overline{B} expressive power. The first one shows how to encode in AB\overline{B} conditions of accomplishment (think of formula \( \varphi \) as the assertion: “Mr. Jones flew from Venice to Nancy”): \( \langle A \rangle (\varphi \land [B](\neg \varphi \land [A] \neg \varphi) \land [B] \neg \varphi) \). Formulas of point-based temporal logics of the form \( \psi \varphi \), using the standard until operator, can be encoded in AB\overline{B} (where atomic intervals are two-point intervals) as follows: \( \langle A \rangle ([B] \varphi) \lor \langle A \rangle ([A]([B] \neg \varphi \land [A] \neg \varphi) \lor [B]([A]([B] \neg \varphi \land [A] \neg \varphi)))) \). Finally, metric conditions like: “\( \varphi \) holds over a right neighbor interval of length greater than \( k \) (resp., less than \( k \), equal to \( k \))” can be captured by the following AB\overline{B} formula: \( \langle A \rangle (\varphi \land [B]^{k+\top}) \) (resp., \( \langle A \rangle (\varphi \land [B]^{k-1 \top}) \) \( \langle A \rangle (\varphi \land [B]^{k-1 \bot} \land (B)^{k-1 \bot}) \).\(^1\)

### 7.1.2 Atoms, types, and dependencies

Let \( S = (\mathbb{N}, A, B, \overline{B}, \sigma) \) be an interval structure and \( \varphi \) be a formula of AB\overline{B}. In the sequel, we shall compare intervals in \( S \) with respect to the set of subformulas of \( \varphi \) they satisfy. To do that, we introduce the key notions of \( \varphi \)-atom, \( \varphi \)-type, \( \varphi \)-cluster, and \( \varphi \)-shading.

First of all, we define the closure \( \mathcal{C}(\varphi) \) of \( \varphi \) as the set of all subformulas of \( \varphi \) and of their negations (we identify \( \neg \varphi \) with \( \neg \varphi \), \( \neg (A) \varphi \) with \( [A] \neg \varphi \), etc.). For technical reasons, we also

\(^1\)It is not difficult to show that AB\overline{B} subsumes the metric extension of A given in [22]. A simple game-theoretic argument shows that the former is in fact strictly more expressive than the latter.
introduce the extended closure $\mathcal{C}l^+(\phi)$, which is defined as the set of all formulas in $\mathcal{C}l(\phi)$ plus all formulas of the forms $\langle R \rangle \alpha$ and $\neg \langle R \rangle \alpha$, with $R \in \{A, B, \bar{B}\}$ and $\alpha \in \mathcal{C}l(\phi)$.

A $\varphi$-atom is any non-empty set $F \subseteq \mathcal{C}l^+(\phi)$ such that (i) for every $\alpha \in \mathcal{C}l^+(\phi)$, we have $\alpha \in F$ if and only if $\alpha \notin F$ and (ii) for every $\gamma = \alpha \lor \beta \in \mathcal{C}l^+(\phi)$, we have $\gamma \in F$ if and only if $\alpha, \beta \in F$ (intuitively, a $\varphi$-atom is a maximal locally consistent set of formulas chosen from $\mathcal{C}l^+(\phi)$). Note that the cardinalities of both sets $\mathcal{C}l(\phi)$ and $\mathcal{C}l^+(\phi)$ are linear in the number $|\psi|$ of subformulas of $\varphi$, while the number of $\varphi$-atoms is at most exponential in $|\psi|$ (precisely, we have $|\mathcal{C}l(\phi)| = 2|\psi|$, $|\mathcal{C}l^+(\phi)| = 14|\psi|$, and there are at most $2^{|\psi|}$ distinct atoms).

We also associate with each interval $I \in \mathcal{S}$ the set of all formulas $\alpha \in \mathcal{C}l^+(\phi)$ such that $\mathcal{S}, I \models \alpha$. Such a set is called $\varphi$-type $I$ and it is denoted by $\text{Type}_\mathcal{S}(I)$. We have that every $\varphi$-type is a $\varphi$-atom, but not vice versa. Hereafter, we shall omit the argument $\varphi$, thus calling a $\varphi$-atom (resp., a $\varphi$-type) simply an atom (resp., a type).

Given an atom $F$, we denote by $\text{Obs}(F)$ the set of all observables of $F$, namely, the formulas $\alpha \in \mathcal{C}l(\phi)$ such that $\alpha \in F$. Similarly, given an atom $F$ and a relation $R \in \{A, B, \bar{B}\}$, we denote by $\text{Req}_R(F)$ the set of all $R$-requests of $F$, namely, the formulas $\alpha \in \mathcal{C}l(\phi)$ such that $\langle R \rangle \alpha \in F$. Taking advantage of the above sets, we can define the following two relations between atoms $F$ and $G$:

$$
F \mathcal{A} \rightarrow G \quad \text{iff} \quad \text{Req}_A(F) = \text{Obs}(G) \cup \text{Req}_B(G) \cup \text{Req}_\bar{B}(G);
$$

$$
F \mathcal{B} \rightarrow G \quad \text{iff} \quad \begin{cases}
\text{Obs}(F) \cup \text{Req}_B(F) \subseteq \text{Req}_B(G) \\
\text{Obs}(F) \cup \text{Req}_\bar{B}(F) \subseteq \text{Obs}(G) \cup \text{Req}_B(G) \\
\text{Obs}(G) \cup \text{Req}_B(G) \cup \text{Req}_\bar{B}(G).
\end{cases}
$$

Note that the relation $\mathcal{B} \rightarrow$ is transitive, while $\mathcal{A} \rightarrow$ is not. Moreover, both $\mathcal{A} \rightarrow$ and $\mathcal{B} \rightarrow$ satisfy a view-to-type dependency, namely, for every pair of intervals $I, J$ in $\mathcal{S}$, we have that

$$
\begin{align*}
I A J & \implies \text{Type}_\mathcal{S}(I) \mathcal{A} \rightarrow \text{Type}_\mathcal{S}(J) \\
I B J & \implies \text{Type}_\mathcal{S}(I) \mathcal{B} \rightarrow \text{Type}_\mathcal{S}(J).
\end{align*}
$$

Relations $\mathcal{A} \rightarrow$ and $\mathcal{B} \rightarrow$ will come into play in the definition of consistency conditions (see Definition 7.1.1).

### 7.1.3 Compass structures

The logic $\mathbb{ABB}$ can be equivalently interpreted over grid-like structures (the so-called compass structures [61]) by exploiting the existence of a natural bijection between the intervals $I = \{x, y\}$ and the points $p = (x, y)$ of an $N \times N$ grid such that $x < y$. As an example, Figure 7.1 depicts four intervals $I_0, \ldots, I_3$ such that $I_0 = A, I_1 = B, I_2 = B, I_3 = C$, together with the corresponding points $p_0, \ldots, p_3$ of a discrete grid (note that the three Allen’s relations $A, B, \bar{B}$ between intervals are mapped to corresponding spatial relations between points; for the sake of readability, we name the latter ones as the former ones).

**Definition 7.1.1.** Given an $\mathbb{ABB}$ formula $\varphi$, a (consistent and fulfilling) compass ($\varphi$-)structure of length $N \leq \omega$ is a pair $\mathcal{S} = (\mathbb{P}_N, \mathcal{L})$, where $\mathbb{P}_N$ is the set of points $p = (x, y)$, with $0 \leq x < y < N$, and $\mathcal{L}$ is function that maps any point $p \in \mathbb{P}_N$ to a ($\varphi$-)atom $\mathcal{L}(p)$ in such a way that

- for every pair of points $p, q \in \mathbb{P}_N$ and every relation $R \in \{A, B\}$, if $p \sim R \sim q$ holds, then $\mathcal{L}(p) \mathcal{A} \rightarrow \mathcal{L}(q)$ follows (consistency);

- for every point $p \in \mathbb{P}_N$, every relation $R \in \{A, B, \bar{B}\}$, and every formula $\alpha \in \text{Req}_R(\mathcal{L}(p))$, there is a point $q \in \mathbb{P}_N$ such that $p \sim R \sim q$ and $\alpha \in \text{Obs}(\mathcal{L}(q))$ (fulfilment).

We say that a compass ($\varphi$-)structure $\mathcal{S} = (\mathbb{P}_N, \mathcal{L})$ features a formula $\alpha$ if there is a point $p \in \mathbb{P}_N$ such that $\alpha \in \mathcal{L}(p)$. The following proposition implies that the satisfiability problem for $\mathbb{ABB}$ is reducible to the problem of deciding, for any given formula $\varphi$, whether there exists a $\varphi$-compass structure that features $\varphi$. 

Proposition 7.1.2. An $\text{ABB}$-formula $\varphi$ is satisfied by some interval structure if and only if it is featured by some $(\varphi,\cdot)$-compass structure.

7.2 Deciding the satisfiability problem for $\text{ABB}$

In this section, we prove that the satisfiability problem for $\text{ABB}$ is decidable by providing a “small-model theorem” for the satisfiable formulas of the logic. For the sake of simplicity, we first show that the satisfiability problem for $\text{ABB}$ interpreted over finite interval structures is decidable and then we generalize such a result to all (finite or infinite) interval structures.

As a preliminary step, we introduce the key notion of shading. Let $\mathcal{G} = (\mathbb{P}_N, \mathcal{L})$ be a compass structure of length $N \leq \omega$ and let $0 \leq y < N$. The shading of the row $y$ of $\mathcal{G}$ is the set $\text{Shading}_y(\mathcal{G}) = \{ \mathcal{L}(x,y) : 0 \leq x < y \}$, namely, the set of the atoms of all points in $\mathbb{P}_N$ whose vertical coordinate has value $y$ (basically, we interpret different atoms as different colors). Clearly, for every pair of atoms $F$ and $F'$ in $\text{Shading}_y(\mathcal{G})$, we have $\text{Req}_A(F) = \text{Req}_A(F')$.

7.2.1 A small-model theorem for finite structures

Let $\varphi$ be an $\text{ABB}$ formula. Let us assume that $\varphi$ is featured by a finite compass structure $\mathcal{G} = (\mathbb{P}_N, \mathcal{L})$, with $N < \omega$. In fact, without loss of generality, we can assume that $\varphi$ belongs to the atom associated with a point $p = (0, y)$ of $\mathcal{G}$, with $0 < y < N$. We prove that we can restrict our attention to compass structures $\mathcal{G} = (\mathbb{P}_N, \mathcal{L})$, where $N$ is bounded by a double exponential in $|\varphi|$. We start with the following lemma that proves a simple, but crucial, property of the relations $\triangleleft$ and $\triangleright$.

Lemma 7.2.1. If $F \triangleleft H$ and $G \triangleright H$ hold for some atoms $F, G, H$, then $F \triangleright G$ holds as well.

Proof. Suppose that $F \triangleleft H$ and $G \triangleright H$ hold for some atoms $F, G, H$. By applying the definitions of the relations $\triangleleft$ and $\triangleright$, we immediately obtain:

\[
\text{Req}_A(F) = \mathcal{O}bs(H) \cup \text{Req}_B(H) \cup \text{Req}_B(H) \quad \text{(since } F \triangleleft H) \\
\text{Req}_A(G) = \mathcal{O}bs(G) \cup \text{Req}_B(G) \cup \text{Req}_B(G) \quad \text{(since } G \triangleright H).
\]

This shows that $F \triangleright G$.

The next lemma shows that, under suitable conditions, a given compass structure $\mathcal{G}$ may be reduced in length, preserving the existence of atoms featuring $\varphi$. 

Figure 7.1: Correspondence between intervals and points of a discrete grid.
Lemma 7.2.2. Let $\mathcal{G}$ be a compass structure featuring $\varphi$. If there exist two rows $0 < y_0 < y_1 < N$ in $\mathcal{G}$ such that $Shading_\mathcal{G}(y_0) \subseteq Shading_\mathcal{G}(y_1)$, then there exists a compass structure $\mathcal{G}'$ of length $N' < N$ that features $\varphi$.

Proof. Suppose that $0 < y_0 < y_1 < N$ are two rows of $\mathcal{G}$ such that $Shading_\mathcal{G}(y_0) \subseteq Shading_\mathcal{G}(y_1)$. Then, there is a function $f : [0, \ldots, y_0 - 1] \to [0, \ldots, y_1 - 1]$ such that, for every $0 \leq x < y_0$, $L(x, y_0) = L(f(x), y_1)$. Let $k = y_1 - y_0$, $N' = N - k$ ($< N$), and $\mathcal{P}_{N'}$, be the portion of the grid that consists of all points $p = (x, y)$, with $0 \leq x < y < N'$. We extend $f$ to a function that maps points in $\mathcal{P}_{N'}$ to points in $\mathcal{P}_N$ as follows:

- if $p = (x, y)$, with $0 \leq x < y < y_0$, then we simply let $f(p) = p$;
- if $p = (x, y)$, with $0 \leq x < y_0 \leq y$, then we let $f(p) = (f(x), y + k)$;
- if $p = (x, y)$, with $y_0 \leq x < y$, then we let $f(p) = (x + k, y + k)$.

We denote by $\mathcal{L}'$ the labeling of $\mathcal{P}_{N'}$ such that, for every point $p \in \mathcal{P}_{N'}$, $\mathcal{L}'(p) = L(f(p))$ and we denote by $\mathcal{G}'$ the resulting structure $(\mathcal{P}_{N'}, \mathcal{L}')$ (see Figure 7.2). We have to prove that $\mathcal{G}'$ is a consistent and fulfilling compass structure that features $\varphi$ (see Definition 7.1.1). First, we show that $\mathcal{G}'$ satisfies the consistency conditions for the relations $B$ and $A$; then we show that $\mathcal{G}'$ satisfies the fulfillment conditions for the $B_\pm$, $B_-$, and $A$-requests; finally, we show that $\mathcal{G}'$ features $\varphi$.

Consistency with relation $B$. Consider two points $p = (x, y)$ and $p' = (x', y')$ in $\mathcal{G}'$ such that $p \mathbin{B} p'$, i.e., $0 \leq x = x' < y' < y < N'$. We prove that $\mathcal{L}'(p) \not\mathbin{\Rightarrow} \mathcal{L}'(p')$ by distinguishing among the following three cases (note that exactly one of such cases holds):

1. $y < y_0$ and $y' < y_0$.
2. $y \geq y_0$ and $y' \geq y_0$.
3. $y \geq y_0$ and $y' < y_0$.

If $y < y_0$ and $y' < y_0$, then, by construction, we have $f(p) = p$ and $f(p') = p'$. Since $\mathcal{G}$ is a (consistent) compass structure, we immediately obtain $\mathcal{L}'(p) = \mathcal{L}(p) \not\mathbin{\Rightarrow} \mathcal{L}(p') = \mathcal{L}'(p')$.

If $y \geq y_0$ and $y' \geq y_0$, then, by construction, we have either $f(p) = (f(x), y + k)$ or $f(p) = (x + k, y + k)$, depending on whether $x < y_0$ or $x \geq y_0$. Similarly, we have either $f(p') = (f(x'), y' + k)$ or $f(p') = (x' + k, y' + k) = (x + k, y' + k)$. This implies $f(p) \mathbin{B} f(p')$ and thus, since $\mathcal{G}$ is a (consistent) compass structure, we have $\mathcal{L}'(p) = \mathcal{L}(f(p)) \not\mathbin{\Rightarrow} \mathcal{L}(f(p')) = \mathcal{L}'(p')$. 

Figure 7.2: Contraction $\mathcal{G}'$ of a compass structure $\mathcal{G}$.
If \( y \geq y_0 \) and \( y' < y_0 \), then, since \( x < y' < y_0 \), we have by construction \( f(p) = (f(x), y + k) \) and \( f(p') = p' \). Moreover, if we consider the point \( p'' = (x, y_0) \) in \( \mathcal{G}' \), we easily see that (i) \( f(p'') = (f(x), y_1) \), (ii) \( f(p) B f(p'') \) (whence \( L(f(p)) \triangleright L(f(p'')) \)), (iii) \( L(f(p'')) = L(p') \), and (iv) \( p'' B p' \) (whence \( L(p'') \triangleright L(p') \)). It thus follows that \( L'(p) = L(f(p)) \triangleright L(f(p'')) = L(p') \). Finally, by exploiting the transitivity of the relation \( \triangleright \), we obtain \( L'(p) \triangleright L'(p') \).

**Consistency with relation A.** Consider two points \( p = (x, y) \) and \( p' = (x', y') \) such that \( p A p' \), i.e., \( 0 \leq x < y = x' < y' < N' \). We define \( p'' = (y, y + 1) \) in such a way that \( p A p'' \) and \( p' B p'' \) and we distinguish between the following two cases:

1. \( y \geq y_0 \).
2. \( y < y_0 \).

If \( y \geq y_0 \), then, by construction, we have \( f(p) A f(p'') \). Since \( \mathcal{G} \) is a (consistent) compass structure, it follows that \( L'(p) = L(f(p)) \rightarrow L(f(p'')) = L(p'') \).

If \( y < y_0 \), then, by construction, we have \( L(p'') = L(f(p'')) \). Again, since \( \mathcal{G} \) is a (consistent) compass structure, it follows that \( L'(p) = L(f(p)) \rightarrow L(p'') = L(f(p'')) = L(p'') \).

In both cases we have \( L'(p) \rightarrow L(p'') \). Now, we recall that \( p' B p'' \) and that, by previous arguments, \( \mathcal{G}' \) is consistent with the relation B. We thus have \( L'(p') \rightarrow L(p'') \). Finally, by applying Lemma 7.2.1, we obtain \( L'(p) \rightarrow L'(p') \).

**Fulfillment of B-requests.** Consider a point \( p = (x, y) \) in \( \mathcal{G}' \) and some B-request \( \alpha \in \mathcal{R}_{B}(L'(p)) \) associated with it. Since, by construction, \( \alpha \in \mathcal{R}_{B}(L(f(p))) \) and \( \mathcal{G} \) is a (fulfilling) compass structure, we know that \( \mathcal{G} \) contains a point \( q' = (x', y') \) such that \( f(p) B q' \) and \( \alpha \in \mathcal{O}b(B(q')) \). We prove that \( \mathcal{G}' \) contains a point \( p' \) such that \( p B p' \) and \( \alpha \in \mathcal{O}b(B(L'(p'))) \) by distinguishing among the following three cases (note that exactly one of such cases holds):

1. \( y < y_0 \).
2. \( y' \geq y_1 \).
3. \( y \geq y_0 \) and \( y' < y_1 \).

If \( y < y_0 \), then, by construction, we have \( p = f(p) \) and \( q' = f(q') \). Therefore, we simply define \( p' = q' \) in such a way that \( p' B q = p' \) and \( \alpha \in \mathcal{O}b(B(L'(p'))) \). If \( y' \geq y_1 \), then, by construction, we have either \( f(p) = (f(x), y + k) \) or \( f(p) = (x + k, y + k) \), depending on whether \( x < y_0 \) or \( x \geq y_0 \). We define \( p' = (x, y' - k) \) in such a way that \( p B p' \).

Moreover, we observe that either \( f(p') = (f(x), y') \) or \( f(p') = (x + k, y') \), depending on whether \( x < y_0 \) or \( x \geq y_0 \), and in both cases \( f(p') \rightarrow q' \) follows. This shows that \( \alpha \in \mathcal{O}b(B(L'(p'))) \) \( = \mathcal{O}b(B(L'(q'))) \).

If \( y \geq y_0 \) and \( y' < y_1 \), then we define \( p = (x, y_0) \) and \( q = (x', y_1) \) and we observe that \( f(p) B q \), \( q B q' \), and \( f(p) = q' \). From \( f(p) B q \) and \( q B q' \), it follows that \( \alpha \in \mathcal{R}_{B}(L(q)) \) and hence \( \alpha \in \mathcal{R}_{B}(L(p)) \). Since \( \mathcal{G} \) is a (fulfilling) compass structure, we know that there is a point \( p' \) such that \( p B p' \) and \( \alpha \in \mathcal{O}b(B(L'(p'))) \). Moreover, since \( p B p' \), we have \( f(p') = p' \), from which we obtain \( p' B p' \) and \( \alpha \in \mathcal{O}b(B(L'(p'))) \).

**Fulfillment of B-requests.** The proof that \( \mathcal{G}' \) fulfills all B-requests of its atoms is symmetric with respect to the previous one.

**Fulfillment of A-requests.** Consider a point \( p = (x, y) \) in \( \mathcal{G}' \) and some A-request \( \alpha \in \mathcal{R}_{A}(L'(p)) \) associated with \( p \) in \( \mathcal{G}' \). Since, by previous assumptions, \( \mathcal{G}' \) fulfills all B-requests of its atoms, it is sufficient to prove that either \( \alpha \in \mathcal{O}b(B(L'(p'))) \) or \( \alpha \in \mathcal{R}_{B}(L'(p')) \), where \( p' = (y, y + 1) \). This can be easily proved by distinguishing among the three cases \( y < y_0 - 1 \), \( y = y_0 - 1 \), and \( y \geq y_0 \).

**Featured formulas.** Recall that, by previous assumptions, \( \mathcal{G} \) contains a point \( p = (0, y) \), with \( 0 < y < N \), such that \( \varphi \in L(p) \). If \( y \leq y_0 \), then, by construction, we have \( \varphi \in L'(p) \) \( = L(f(p)) \). Otherwise, if \( y > y_0 \), we define \( q = (0, y_0) \) and we observe that \( q B p \). Since \( \mathcal{G} \) is a (consistent) compass structure and \( \mathcal{B}\varphi \in \mathcal{C}l^+ \), we have that \( \varphi \in \mathcal{R}_{B}(L(q)) \).

Moreover, by construction, we have \( L'(q) = L(f(q)) \) and hence \( \varphi \in \mathcal{R}_{B}(L'(q)) \). Finally, since
§’ is a (fulfilling) compass structure, we know that there is a point p’ in §’ such that f(q) £ p’ and ϕ ∈ Obs(L’(p’)).

On the grounds of the above result, we can provide a suitable upper bound for the length of a minimal finite interval structure that satisfies ϕ, if there exists any. This yields a straightforward, but inefficient, 2EXPSPACE algorithm that decides whether a given ABP-formula ϕ is satisfiable over finite interval structures.

**Theorem 7.2.3.** An ABP-formula ϕ is satisfied by some finite interval structure if and only if it is satisfied by some compass structure of length $N \leq 2^{2^{|ϕ|}}$ (i.e., double exponential in |ϕ|).

**Proof.** One direction is trivial. We prove the other one (“only if” part). Suppose that ϕ is satisfied by a finite interval structure §. By Proposition 7.1.2, there is a compass structure § that features ϕ and has finite length $N < ω$. Without loss of generality, we can assume that N is minimal among all finite compass structures that feature ϕ. We recall from Section 7.1.2 that § contains at most $2^{2^{|ϕ|}}$ distinct atoms. This implies that there exist at most $2^{2^{|ϕ|}}$ different shadings of the form Shading$_{G, \tilde{ϕ}}(y)$, with $0 \leq y < N$. Finally, by applying Lemma 7.2.2, we obtain $N \leq 2^{2^{|ϕ|}}$ (otherwise, there would exist two rows $0 < y_0 < y_1 < N$ such that Shading$_{G, \tilde{ϕ}}(y_0) = Shading_{G, \tilde{ϕ}}(y_1)$, which is against the hypothesis of minimality of N).

7.2.2 A small-model theorem for infinite structures

In general, compass structures that feature ϕ may be infinite. Here, we prove that, without loss of generality, we can restrict our attention to sufficiently “regular” infinite compass structures, which can be represented in double exponential space with respect to |ϕ|. To do that, we introduce the notion of periodic compass structure.

**Definition 7.2.4.** An infinite compass structure § = (Pω, L) is periodic, with threshold $\tilde{y}_0$, period $\tilde{y}$, and binding $\tilde{g} : \{0, ..., \tilde{y}_0 + \tilde{y} - 1\} \rightarrow \{0, ..., \tilde{y}_0 - 1\}$, if the following conditions are satisfied:

- for every $\tilde{y}_0 + \tilde{y} \leq x < y$, we have $L(x, y) = L(x - \tilde{y}, y - \tilde{y})$,
- for every $0 \leq x < \tilde{y}_0 + \tilde{y} \leq y$, we have $L(x, y) = L(\tilde{g}(x), y - \tilde{y})$.

Figure 7.3 gives an example of a periodic compass structure (the arrows represent some relationships between points induced by the binding function $\tilde{g}$). Note that any periodic compass structure § = (Pω, L) can be finitely represented by specifying (i) its threshold $\tilde{y}_0$, (ii) its period $\tilde{y}$, (iii) its binding $\tilde{g}$, and (iv) the labeling L restricted to the portion $P_{\tilde{y}_0 + \tilde{y} - 1}$ of the domain.

The following theorem leads immediately to a 2EXPSPACE algorithm that decides whether a given ABP-formula ϕ is satisfiable over infinite interval structures.

**Theorem 7.2.5.** An ABP-formula ϕ is satisfied by an infinite interval structure if and only if it is satisfied by a periodic compass structure with threshold $\tilde{y}_0 < 2^{2^{|ϕ|}}$ and period $\tilde{y} < 2^{|ϕ|} \cdot 2^{2^{|ϕ|}} \cdot 2^{2^{|ϕ|}}$.

**Proof.** One direction is trivial. We prove the other one (“only if” part). Suppose that ϕ is satisfied by an infinite interval structure §. By Proposition 7.1.2, there is an infinite compass structure § that features ϕ. Below, we show how to turn § into a periodic compass structure §’ that still features ϕ and whose threshold and period satisfy the bounds given by the theorem.

**Threshold $\tilde{y}_0$.** Since § is infinite, we know that there exist infinitely many rows $y_0, y_1, y_2, ...$ such that Shading$_{G, \tilde{ϕ}}(y_0) = Shading_{G, \tilde{ϕ}}(y_1)$ for every pair of indices i, j ∈ N. We define $\tilde{y}_0$ as the least of all such rows. By simple counting arguments, we have that $\tilde{y}_0 < 2^{2^{|ϕ|}}$.

**Period $\tilde{y}$.** Since § is a (fulfilling) compass structure, there is a function f that maps any point $p = (x, y_0)$, any relation $R \in \{A, \tilde{B}\}$, and any request $\alpha \in \text{Req}_R(L(p))$ to a point $p’ = f(p, R, \alpha)$ such that $p \text{ R p’}$ and $\alpha \in \text{Obs}_L(L’(p’))$. Let $f$ be one such function. We denote by $\text{Img}(f)$ the image set of f, namely, the set of all points of the form $p’ = f(p, R, \alpha)$, with $p = (x, y_0)$, $R \in \{A, \tilde{B}\}$, and $\alpha \in \text{Req}_R(L(p))$. Moreover, we denote by $\text{Im}_{\tilde{y}}(f)$ the projection of $\text{Img}(f)$ on the
y-component. Intuitively, $3mg_y(f)$ is a minimal set of rows that fulfill all A-requests and all B-requests of atoms along the row $\bar{y}_0$ in $\mathcal{G}$ (see, for instance, Figure 7.4). Clearly, $\min(3mg_y(f)) > \bar{y}_0$ and $3mg_y(f)$ contains at most $2|\varphi| \cdot \bar{y}_0$ (possibly non-contiguous) rows (namely, at most one row for each choice of $0 \leq x < \bar{y}_0$, $R \in \{A, B\}$, and $\alpha \in Req_R(\mathcal{L}(x, \bar{y}_0))$). We call gap of $3mg_y(f)$ any set $Y = \{y, y+1, ..., y'\}$ of contiguous rows of $\mathcal{G}$ such that $\bar{y}_0 < y < y' < \max(3mg_y(f))$ and $3mg_y(f) \cap Y = \emptyset$. From previous results (in particular, from the proofs of Lemma 7.2.2 and Theorem 7.2.3), we can assume, without loss of generality, that every gap $Y$ of $3mg_y(f)$ has size at most $2^{2^{2^{|\varphi|}}} - 1$ (otherwise, we can find two rows $y'_0$ and $y'_1$ in $Y$ that satisfy the hypothesis of Lemma 7.2.2 and hence we can “remove” the rows from $y'_0$ to $y'_1$ from $\mathcal{G}$, without affecting consistency and fulfillment). This shows that $\max(3mg_y(f)) \leq \bar{y}_0 + 2|\varphi| \cdot 2^{2^{2^{|\varphi|}}}$. We then define $\tilde{y}$ as the least value such that $\bar{y}_0 \tilde{y} > \max(3mg_y(f))$ and $Shading_2(\bar{y}_0) = Shading_2(\bar{y}_0 + \tilde{y})$. Again, by exploiting simple counting arguments, one can prove that $\tilde{y} < \max(3mg_y(f)) - \bar{y}_0 + 2^{2^{2^{|\varphi|}}} \leq 2|\varphi| \cdot 2^{2^{2^{|\varphi|}}} + 2^{2^{2^{|\varphi|}}} \leq 2|\varphi| \cdot (\bar{y}_0 + 1) \cdot 2^{2^{2^{|\varphi|}}} \leq 2|\varphi| \cdot 2^{2^{2^{|\varphi|}}}.$

**Binding $\tilde{g}$**. Since $Shading_2(\bar{y}_0) = Shading_2(\bar{y}_0 + \tilde{y})$, we know that there is a (surjective) function $g$ that maps any value $x \in \{0, ..., \bar{y}_0 + \tilde{y} - 1\}$ to a value $g(x) \in \{0, ..., \bar{y}_0 - 1\}$ in such a way that $\mathcal{L}(x, \bar{y}_0 + \tilde{y}) = \mathcal{L}(g(x), \bar{y}_0)$. We choose one such function as $\tilde{g}$.

**Periodic compass structure $\mathcal{G}'$**. According to Definition 7.2.4, the threshold $\bar{y}_0$, the period $\bar{y}$, the binding $\tilde{g}$, and the labeling $\mathcal{L}$ of $\mathcal{G}$ restricted to the finite domain $\mathbb{P}_{\bar{y}_0 + \bar{y} - 1}$ uniquely determine a periodic structure $\mathcal{G}' = (\mathbb{P}_\omega, \mathcal{L}')$. It thus remains to show that $\mathcal{G}'$ is a (consistent and fulfilling) compass structure that features $\varphi$. The proof that the labeling $\mathcal{L}'$ is consistent with the relations $A$, $B$, and $\bar{B}$ is straightforward, given the above construction. As for the fulfillment of the various requests, one can prove, by induction on $n$, that, for every $n \in \mathbb{N}$, every point $p = (x, y)$ with $y = \bar{y}_0 + ny$, every relation $R \in \{A, \bar{B}\}$ (resp., $R = B$), and every $R$-request $\alpha \in Req_R(\mathcal{L}'(p))$, there is a point $p' = (x', y')$ such that $y' \leq \bar{y}_0 + (n + 1)\bar{y}$ (resp., $y' < \bar{y}_0 + n\bar{y}$), $pRp'$, and $\alpha \in Obs(\mathcal{L}'(p'))$. This suffices to claim that $\mathcal{G}'$ is a consistent and fulfilling compass structure. Consider the case of relation $\bar{B}$ (the case of relation $B$ is fully symmetric and the case of relation
Figure 7.4: A set $\mathfrak{M}_{\tilde{y}}(f)$ of rows that fulfill all requests at row $\tilde{y}_0$.

A can be easily reduced to that of $\mathcal{B}$). By contradiction, let us suppose that there is a point $p = (x, y)$, with $\tilde{y}_0 + n\tilde{y} < y < \tilde{y}_0 + (n+1)\tilde{y}$, such that $\alpha \in \mathcal{R}_{\mathcal{B}}(L(p))$ and $\alpha \notin \mathcal{B} \{L(p')\}$ for all points $p'$ such that $p \notin p'$. Since $\mathcal{G}'$ is consistent, we have $\alpha \in \mathcal{R}_{\mathcal{B}}(L(q))$, where $q = (x, \tilde{y}_0 + (n+1)\tilde{y})$ (note that $p \notin q$ holds) and thus, by construction, there is a point $q' = (x, y')$, with $\tilde{y}_0 + (n+1)\tilde{y} < y' \leq \tilde{y}_0 + (n+2)\tilde{y}$, such that $\alpha \in \mathcal{B} \{L(q')\}$ (a contradiction). Finally, one can show that $\mathcal{G}'$ features the formula $\varphi$ by exploiting the same argument that was given in the proof of Lemma 7.2.2.

\section{7.3 Tight complexity bounds to the satisfiability problem for $\mathcal{ABB}$}

In this section, we show that the satisfiability problem for $\mathcal{ABB}$ interpreted over (either finite or infinite) interval temporal structures is EXPSPACE-complete.

The EXPSPACE-hardness of the satisfiability problem for $\mathcal{ABB}$ follows from a reduction from the exponential-corridor tiling problem, which is known to be EXPSPACE-complete [59]. Formally, an instance of the exponential-corridor tiling problem is a tuple $T = \langle T, t_\bot, t_\top, H, V, n \rangle$ consisting of a finite set $T$ of tiles, a bottom tile $t_\bot \in T$, a top tile $t_\top \in T$, two binary relations $H, V$ over $T$ (specifying the horizontal and vertical constraints), and a positive natural number $n$ (represented in unary notation). The problem consists in deciding whether there exists a tiling $f : \mathbb{N} \times \mathbb{N} \rightarrow T$ of the infinite discrete corridor of height $2^n$, that associates the tile $t_\bot$ (resp., $t_\top$) with the bottom (resp., top) row of the corridor and that respects the horizontal and vertical constraints $H$ and $V$, namely,

i) for every $x \in \mathbb{N}$, we have $f(x, 0) = t_\bot$,

ii) for every $x \in \mathbb{N}$, we have $f(x, 2^n - 1) = t_\top$,

iii) for every $x \in \mathbb{N}$ and every $0 \leq y < 2^n$, we have $f(x, y) H f(x + 1, y)$,

iv) for every $x \in \mathbb{N}$ and every $0 \leq y < 2^n - 1$, we have $f(x, y) V f(x, y + 1)$.

The proof of the following lemma, which reduces the exponential-corridor tiling problem to the satisfiability problem for $\mathcal{ABB}$. Intuitively, such a reduction exploits (i) the correspondence between the points $p = (x, y)$ inside the infinite corridor $\mathbb{N} \times \{0, \ldots, 2^n - 1\}$ and the intervals of the form $I_p = [y + 2^n \times x, y + 2^n \times x + 1]$, (ii) [T] propositional variables which represent the tiling function $f$, (iii) $n$ additional propositional variables which represent (the binary expansion of) the $y$-coordinate of each row of the corridor, and (iv) the modal operators $\langle A \rangle$ and $\langle B \rangle$ by means of which one can enforce the local constrains over the tiling function $f$ (as a matter of fact, this shows that the satisfiability problem for the $\mathcal{AB}$ fragment is already hard for EXPSPACE).
Lemma 7.3.1. There is a polynomial-time reduction from the exponential-corridor tiling problem to the satisfiability problem for \( \mathcal{A}\mathcal{B}\).

Proof. Consider a generic instance \( \mathcal{I} = (T, t_\perp, t_\top, H, V, n) \) of the exponential-corridor tiling problem, where \( T = \{t_1, \ldots, t_k\} \). We guarantee the existence of a tiling function \( f : \mathbb{N} \times \{0, \ldots, 2^n - 1\} \rightarrow T \) that satisfies the instance \( \mathcal{I} \) through the existence of a labeled (infinite) interval structure \( S = (I_o, A, B, \sigma) \) that satisfies a suitable \( \mathcal{A}\mathcal{B} \) formula with size polynomial in \( \mathcal{I} \). We use \( k \) propositional variables \( t_1, \ldots, t_k \) to represent the tiles from \( T \), \( n \) propositional variables \( y_0, \ldots, y_{n-1} \) to represent the binary expansion of the \( y \)-coordinate of a row, and one propositional variable \( c \) to identify those intervals in \( I_o \) that correspond to points of the infinite corridor of height \( 2^n \). The correspondence between the points \( p = (x, y) \), with \( x \in \mathbb{N} \) and \( 0 \leq y < 2^n \), of the infinite corridor and the intervals \( I_p \in I_o \) is obtained by letting \( I_p = [y + 2^n x, y + 2^n x + 1] \) (Figure 7.5 can be used as a reference example through the rest of the proof). According to such an encoding, the labeling function \( \sigma \) is related to the tiling function \( f \) as follows:

\[
\text{for every point } p = [x, y] \in \mathbb{N} \times \{0, \ldots, 2^n - 1\} \text{ and every index } 1 \leq i \leq k, \text{ if } f(p) = t_i, \text{ then } \sigma(I_p) = \{c, t_i, y_{j_1}, \ldots, y_{j_h}\}, \text{ where } \{j_1, \ldots, j_h\} \subseteq \{0, \ldots, n-1\} \text{ and } y = \sum_{i \in \{j_1, \ldots, j_h\}} 2^i.
\]

For the sake of brevity, we introduce a universal modal operator \([U]\), which is defined as follows:

\[
[U] \alpha = \alpha \land [A] \alpha \land [A][A] \alpha.
\]

We now show how to express the existence of a tiling function \( f \) that satisfies \( \mathcal{I} \). First of all, we associate the propositional variable \( c \) with all and only the intervals of the form \( I_p = [y + 2^n x, y + 2^n x + 1] \), with \( x \in \mathbb{N} \) and \( 0 \leq y < 2^n \) (atomic intervals), as follows:

\[
\varphi_c = [U][c \leftrightarrow [B] \perp].
\]
The tiling function $f : N \times \{0, ..., 2^n - 1\} \rightarrow \mathbb{T}$ is represented by associating with each $c$-labeled interval $I_p = [y + 2^n x, y + 2^n x + 1]$ a unique propositional variable $\phi(p)$ in $T$ as follows:

$$\varphi_f = [\mathbb{U}] \left( (c \rightarrow \bigvee_{1 \leq i \leq k} t_i) \right) \land [\mathbb{U}] \left( (c \rightarrow \bigwedge_{1 \leq i < j \leq k} -(t_i \land t_j)) \right).$$

Next, we associate with each (possibly non-minimal) interval of the form $I = [y + 2^n x, y + 2^n x + 1]$ a subset of the propositional variables $y_0, ..., y_{n-1}$ that encodes the binary expansion of $y$. Such a labeling can be enforced by the formula:

$$\varphi_y = \left( \bigwedge_{0 \leq i < n} -y_i \right) \land [\mathbb{U}] \left( \bigwedge_{0 \leq i < n} (y_i \leftrightarrow [B]y_i) \land (-y_i \leftrightarrow [B]-y_i) \right) \land [\mathbb{U}] \left( c \rightarrow \varphi_{\text{inc}}^{i} \right),$$

where the formula $\varphi_{\text{inc}}^{i}$ is defined (by induction on $i \in \{n, ..., 0\}$) as follows:

$$\varphi_{\text{inc}}^{i} = \begin{cases} \top & \text{if } i = n, \\ \left( y_i \land (A)(c \land -y_i) \land \varphi_{\text{eq}}^{i+1} \right) \lor \left( -y_i \land (A)(c \land y_i) \land \varphi_{\text{eq}}^{i+1} \right) & \text{if } i < n, \end{cases}$$

The formula $\varphi_{\text{eq}}^{i}$ involves the formula $\varphi_{\text{eq}}^{i}$, which is defined (by induction on $i \in \{n, ..., 0\}$) as follows:

$$\varphi_{\text{eq}}^{i} = \begin{cases} \top & \text{if } i = n, \\ \left( (y_i \land (A)(c \land y_i)) \lor (y_i \land (A)(c \land -y_i)) \right) \land \varphi_{\text{eq}}^{i+1} & \text{if } i < n. \end{cases}$$

It remains to express the constraints on the tiling function $f$. This can be done by using the following formulas (for the sake of simplicity, we assume, without loss of generality, that $(t_{\top}, t_{\bot}) \in V$):

$$\varphi_{\bot} = [\mathbb{U}] \left( c \land \bigwedge_{0 \leq i < n} -y_i \rightarrow t_{\bot} \right)$$

$$\varphi_{\top} = [\mathbb{U}] \left( c \land \bigwedge_{0 \leq i < n} y_i \rightarrow t_{\top} \right)$$

$$\varphi_H = [\mathbb{U}] \bigwedge_{1 \leq i \leq k} \left( \varphi_{\text{corr}} \land (B)(c \land t_i) \right) \rightarrow [V_{(i,t_i)} \in H \langle A \rangle (c \land t_i)]$$

$$\varphi_V = [\mathbb{U}] \bigwedge_{1 \leq i \leq k} \left( (c \land t_i) \rightarrow [V_{(i,t_i)} \in V \langle A \rangle (c \land t_i)] \right),$$

where $\varphi_{\text{corr}} = \varphi_{\text{eq}}^{0} \land [B]^{-\varphi_{\text{eq}}^{0}}$ (intuitively, the formula $\varphi_{\text{corr}}$ holds over all and only the intervals of the form $I = [y + 2^n x, y + 2^n (x + 1)]$, in such a way that, if $J$ and $K$ are the shortest intervals such that $I \cap J$ and $I \cap K$, then $J$ corresponds to the point $p = (x, y)$ and $K$ corresponds to the point $q = (x + 1, y)$).

Summing up, we have that the formula $\varphi = \varphi_c \land \varphi_f \land \varphi_y \land \varphi_{\bot} \land \varphi_{\top} \land \varphi_H \land \varphi_V$, which has polynomial size in $|T|$ and uses only the modal operators $\langle A \rangle$ and $\langle B \rangle$, is satisfiable if and only if $T$ is a positive instance of the exponential-corridor tiling problem.

As for the EXPSPACE-completeness, we claim that the existence of a compass structure $\mathcal{S}$ that features a given formula $\varphi$ can be decided by verifying suitable local (and stronger) consistency conditions over all pairs of contiguous rows. In fact, in order to check that these local conditions hold between two contiguous rows $y$ and $y + 1$, it is sufficient to store into memory a bounded amount of information, namely, (i) a counter $y$ that ranges over $\{1, ..., 2^{7w} + |\varphi| \cdot 2^{7w}\}$, (ii) the two guessed shadings $S$ and $S'$ associated with the rows $y$ and $y + 1$, and (iii) a function $g : S \rightarrow S'$
that captures the horizontal alignment relation between points with an associated atom from \( S \) and points with an associated atom from \( S' \). This shows that the satisfiability problem for \( \text{AB}\bar{\text{B}} \) can be decided in exponential space, as claimed by Lemma 7.3.2 below.

In order to prove this lemma, we preliminarily need to introduce two variants of the dependency relations \( \sim \) and \( \rightarrow \), which are more restrictive than the previous ones and which are evaluated (locally) over pairs of atoms that lie along two contiguous rows. Precisely, we define the following relations between atoms \( F \) and \( G \):

\[
\begin{align*}
F & \sim G \quad \text{iff} \quad \begin{cases} 
\mathsf{Req}_A(F) = \mathcal{O}bs(G) \cup \mathsf{Req}_B(G) \\
\mathsf{Req}_B(G) = \emptyset
\end{cases} \\
F & \equiv G \quad \text{iff} \quad \begin{cases} 
\mathsf{Req}_A(F) = \mathcal{O}bs(G) \cup \mathsf{Req}_B(G) \\
\mathsf{Req}_B(G) = \emptyset \cup \mathsf{Req}_F(G)
\end{cases}
\end{align*}
\]

Note that \( F \sim G \) (resp., \( F \equiv G \)) implies \( F \rightarrow G \) (resp., \( F \equiv G \)), but the converse implications are not true in general. Moreover, it is easy to see that any consistent and fulfilling finite compass structure \( \mathcal{G} = (\mathbb{N}, \mathcal{L}) \), with \( N \in \mathbb{N} \), satisfies the following properties, and, conversely, any finite structure \( \mathcal{G} = (\mathbb{N}, \mathcal{L}) \), with \( N \in \mathbb{N} \), that satisfies the following properties is a consistent and fulfilling compass structure:

i) for every pair of points \( p = (x, y) \) and \( q = (y, y + 1) \) in \( \mathcal{G} \), we have \( \mathcal{L}(p), \rightarrow \mathcal{L}(q) \),

ii) for every pair of points \( p = (x, y) \) and \( q = (x, y + 1) \) in \( \mathcal{G} \), we have \( \mathcal{L}(q), \rightarrow \mathcal{L}(p) \),

iii) for the lower-left point \( p = (0, 1) \) in \( \mathcal{G} \), we have \( \mathsf{Req}_B(\mathcal{L}(p)) = \emptyset \),

iv) for every upper point \( p = (x, N) \) in \( \mathcal{G} \), we have \( \mathsf{Req}_B(\mathcal{L}(p)) = \emptyset \) and \( \mathsf{Req}_A(\mathcal{L}(p)) = \emptyset \).

**Lemma 7.3.2.** There is an EXPSPACE non-deterministic procedure that decides whether a given formula of \( \text{AB}\bar{\text{B}} \) is satisfiable or not.

**Proof.** We first consider the (easier) case of satisfiability with interpretation over finite interval structures; then, we shall deal with the more general case of satisfiability with interpretation over infinite interval structures.

**Finite Case.** In Figure 7.6, we describe an EXPSPACE non-deterministic procedure that decides whether a given \( \text{AB}\bar{\text{B}} \) formula is satisfiable over finite labeled interval structures. Below, we prove that such a procedure is sound and complete.

(Soundness) As for the soundness, we consider a successful computation of the procedure and we show that there is a finite compass structure \( \mathcal{G} = (\mathbb{N}, \mathcal{L}) \) of guesses \( \phi \), where \( N \in \mathbb{N} \) is exactly the value that was guessed at the beginning of the computation. We build such a structure \( \mathcal{G} \) inductively on the value of the variable \( y \in \{1, ..., N\} \) as follows.

- If \( y = 1 \), then we let \( \mathcal{G}_1 = (\mathbb{I}_1, \mathcal{L}_1) \), where \( \mathcal{L}_1 \) maps the unique point of \( \mathbb{I}_1 \) to the atom \( F \) that was guessed at the beginning of the computation. Note that \( \mathcal{G}_1 \) satisfies the consistency condition of Definition 7.1.1, but it may not satisfy the fulfillment condition for the relations \( A \) and \( B \).

- If \( y > 1 \), then assuming that \( \mathcal{G}_{y-1} = (\mathbb{I}_{y-1}, \mathcal{L}_{y-1}) \) is the consistent (possibly non-fulfilling) compass structure obtained during the \( y - 1 \)-th iteration, we define \( \mathcal{G}_y = (\mathbb{I}_y, \mathcal{L}_y) \), where:
  i) \( \mathcal{L}_y(p) = \mathcal{L}_{y-1}(p) \) for every point \( p = (x', y') \) that belongs to \( \mathbb{I}_{y-1} \), namely, such that \( 0 \leq x' < y' < y \);
  ii) \( \mathcal{L}_y(p) = f(\mathcal{L}_{y-1}(q)) \) for every pair of points of the form \( p = (x, y) \) and \( q = (x, y - 1) \), with \( 0 \leq x < y - 1 \), where \( f \) is the function guessed during the \( y \)-th iteration;
  iii) \( \mathcal{L}_y(p) = \mathcal{G}_\bar{p} \), where \( \mathcal{G}_\bar{p} = (y - 1, y) \) and \( \mathcal{G} \) is the atom guessed during the \( y \)-th iteration.

We then define \( \mathcal{G} \) to be the structure \( \mathcal{G}_N \). Now, knowing that every call to the function **Check-Consistency** was successful, we can conclude that the structure \( \mathcal{G} \) satisfies the following two properties:
7.3. Tight complexity bounds to the satisfiability problem for \( \text{ABF} \) 

let \( \varphi \) be an input formula

\[
\text{procedure CheckConsistency}(S, f, \vec{G})
\]

for each \( \varphi \)-atom \( F \in S \)
\[
\begin{cases}
\text{do} & \text{if } F, \vec{G} \not\rightarrow \neg G \\
& \text{or } f(F), \vec{G} \not\rightarrow F \\
& \text{then return false}
\end{cases}
\]
return true

\[
\text{procedure CheckFulfillment}(S)
\]

for each \( \varphi \)-atom \( F \in S \)
\[
\begin{cases}
\text{do} & \text{if } \mathcal{R}eq_A(F) \neq \emptyset \text{ or } \mathcal{R}eq_B(F) \neq \emptyset \\
& \text{then return false}
\end{cases}
\]
return true

main
\[
\begin{align*}
N & \leftarrow \text{any value in } \{1, \ldots, 2^{27\nu}\} \\
F & \leftarrow \text{any } \varphi \text{-atom such that } \mathcal{R}eq_B(F) = \emptyset \text{ and } \varphi \in \mathcal{O}bs(F) \cup \mathcal{R}eq_B(F) \\
S & \leftarrow \{F\}
\end{align*}
\]
for \( y \leftarrow 1 \) to \( N \)
\[
\begin{cases}
\{f \leftarrow \text{any mapping from } S \text{ to the set of all } \varphi \text{-atoms}\} \\
\vec{G} & \leftarrow \text{any } \varphi \text{-atom} \\
\text{do} & \text{if not CheckConsistency}(S, g, \vec{G}) \\
& \text{then return false}
\end{cases}
\]
\[
S \leftarrow \{f(F) : F \in S\} \cup \{\vec{G}\}
\]
return CheckFulfillment(S)

Figure 7.6: Algorithm for the satisfiability problem over finite structures.

i) for every pair of points \( p = (x, y) \) and \( q = (y, y + 1) \) in \( \mathcal{S} \), we have \( \mathcal{L}(p), \rightarrow \mathcal{L}(q) \),

ii) for every pair of points \( p = (x, y) \) and \( q = (x, y + 1) \) in \( \mathcal{S} \), we have \( \mathcal{L}(q), \rightarrow \mathcal{L}(p) \).

Moreover, since the first guessed atom \( F \) was such that \( \mathcal{R}eq_B(F) = \emptyset \) and since the call to the function CheckFulfillment at the end of the computation was successful, we know that \( \mathcal{S} \) satisfies also the following two properties:

iii) for the lower-left point \( p = (0, 1) \), we have \( \mathcal{R}eq_B(\mathcal{L}(p)) = \emptyset \),

iv) for every upper point \( p = (x, N) \), we have \( \mathcal{R}eq_B(\mathcal{L}(p)) = \emptyset \) and \( \mathcal{R}eq_A(\mathcal{L}(p)) = \emptyset \).

By previous arguments, this shows that \( \mathcal{S} \) is a consistent and fulfilling compass structure. Finally, since the first guessed atom \( F \) was such that \( \varphi \in \mathcal{O}bs(F) \cup \mathcal{R}eq_B(F) \), we have that \( \mathcal{S} \) features the input formula \( \varphi \). Proposition 7.1.2 finally implies that there is a labeled finite interval structure that satisfies \( \varphi \).

(COMPLETENESS) As for completeness, we consider a finite labeled interval structure \( \mathcal{S} = (I_N, A, B, \vec{S}, \sigma) \) that satisfies \( \varphi \). By Theorem 7.2.3, we know that there is a (consistent and fulfilling) compass structure \( \mathcal{S} = (I_N, \mathcal{L}) \) of length \( N \leq 2^{27\nu} \) that features \( \varphi \). We exploit such a structure \( \mathcal{S} \) to show that there is a successful computation of the algorithm of Figure 7.6. To do that, it is sufficient to describe, at each step of the computation where the value of a variable needs to be guessed, which is the right choice for that value. Clearly, at the beginning of the computation, the variable \( N \) will take as value exactly the length of the compass structure \( \mathcal{S} \). Similarly, the initial value for the variable \( F \) is chosen to be the atom \( \{\mathcal{L}(p)\} \) associated with the lower-left point \( p = (0, 1) \). Then, at each iteration of the main loop, we choose the values for \( f \) and for \( \vec{S} \) as follows. We assume that, at the \( y \)-th iteration, \( S \) is exactly the shading associated with the row \( y \) in \( \mathcal{S} \) (it can be easily proved that this is an invariant of the computation) and, for every atom \( F \) in \( S \), we denote by \( p_F = (x_F, y) \)
a generic point along the row $y$ such that $L(p_F) = F$ (such a point exists by assumption). We then choose $f$ to be the function that maps every atom $F \in S$ to the atom $f(F) = L(x_F, y + 1)$. It is routine to prove that the computation that results from the above-defined sequence of guesses is successful.

**Infinite case.** Figure 7.7 reports an EXPSPACE non-deterministic procedure that decides whether a given AB5 formula is satisfiable over infinite labeled interval structures.

(Soundness) In order to prove that the described procedure is sound, we consider a successful computation of the procedure and we show that there is an infinite periodic compass structure $\mathcal{J} = (\mathcal{P}_\omega, \mathcal{L})$ that features $\varphi$. The threshold $y_0$ and the period $y$ of $\mathcal{J}$ are defined to be the values of the corresponding variables that were guessed at the beginning of the computation. As for the binding function $g$, we choose any arbitrary mapping $g$ from $S$ to $\mathcal{J}$ such that $g \circ f$ is the identity on $S$, where $S$, $\mathcal{S}$, and $f$ are the values of the corresponding variables at the end of the computation. It now remains to describe the labeling of the finite portion $\mathcal{P}_{y_n + y - 1}$ of $\mathcal{J}$ (note that this labeling uniquely determines the infinite periodic compass structure $\mathcal{J}$). This can be done by following the same construction given in the finite case. Similarly, the fact that $\mathcal{J}$ satisfies the consistency conditions of Definition 7.1.1 can be proved by exploiting arguments analogous to the finite case.

The proof that $\mathcal{J}$ satisfies also the fulfillment condition requires more details. In particular, one can prove, again by exploiting induction on $y$, that for every row $y$, with $y_0 < y < y_0 + y$, every point $p = (x, y_0)$, every relation $R \in \{A, B\}$, and every $R$-request $\alpha \in Req_R(\mathcal{L}(p))$, if $\mathcal{L}(p) = F (\in \mathcal{S})$ and fulfilled[$F, R, \alpha$] is true during the $y$-th iteration of the main loop, then there exists a point $q = (x', y)$ such that $p R q$ and $\alpha \in Obs(\mathcal{L}(q))$. Thus, at the end of the computation, since all entries of the variable fulfilled are set to true, we know that all $A$-requests and all $B$-requests of atoms associated with row $y_0$ are fulfilled below row $y_0 + y$. This shows that $\mathcal{J}$ is a consistent and fulfilling compass structure. As before, one can conclude that $\mathcal{J}$ features the input formula $\varphi$ and hence there exists an infinite labeled interval structure that satisfies $\varphi$.

(Completeness) As for completeness, we consider an infinite labeled interval structure $S = (I_\omega, A, B, \mathcal{S}, \mathcal{O})$ that satisfies $\varphi$. By Theorem 7.2.5, we know that there is a periodic (consistent and fulfilling) compass structure $\mathcal{J} = (I_\omega, \mathcal{L})$, with threshold $y_0 < 2^{2^{\omega}}$, period $y < 2|\varphi| \cdot 2^{2^{\omega}}$, and binding $g : [0, ..., y_0 + y - 1] \rightarrow [0, ..., y_0 - 1]$. We exploit such a periodic structure $\mathcal{J}$ to show that there is a successful computation of the algorithm of Figure 7.7. In particular, at each step of the computation where the value of a variable needs to be guessed, we describe which is the right choice for that value. Clearly, at the beginning of the computation, the variables $y_0$ and $y$ will take as values exactly the threshold and the period of the compass structure $\mathcal{J}$. Similarly, the initial value for the variable $F$ is chosen to be the atom $\{L(p)\}$ associated with the lower-left point $p = (0, 1)$. Then, at each iteration of one of the two main loops, we choose the values for $f$ and for $\mathcal{G}$ as follows. We assume that, at each iteration of one of the two loops, $S$ is the shading associated with the row $y$ in $\mathcal{J}$, where $y$ is the value of the corresponding variable (it can be easily proved that this is an invariant of the computation) and, for every atom $F \in S$, we denote by $p_F = (x_F, y)$ a generic point along the row $y$ such that $L(p_F) = F$ (such a point exists by assumption). We then choose $f$ to be the function that maps every atom $F \in S$ to the atom $f(F) = L(x_F, y + 1)$. It is routine to prove that the computation that results from the above-defined sequence of guesses is successful.

Summing up, we obtain the following tight complexity result.

**Theorem 7.3.3.** The satisfiability problem for AB5 interpreted over (prefixes of) natural numbers is EXPSPACE-complete.
let \( \varphi \) be an input formula

procedure CheckConsistency\((S, f, G)\)

as before

procedure UpdateFulfillment\((fulfilled, \tilde{S}, f, S, G)\)

for each \( \varphi \)-atom \( F \in S \) and A-request \( \alpha \in \text{Req}_A(F) \)

do if \( \alpha \in \text{Obs}(G) \)

doi\( \text{fulfilled}[F, A, \alpha] \) ← true

for each \( \varphi \)-atom \( F \in \tilde{S} \) and B-request \( \alpha \in \text{Req}_B(F) \)

do if \( \alpha \in \text{Obs}(f(F)) \)

doi\( \text{fulfilled}[F, B, \alpha] \) ← true

procedure CheckFulfillment\((fulfilled, \tilde{S}, \tilde{f}, S)\)

if \( S \neq \tilde{S} \)
then return false

for each \( \varphi \)-atom \( \tilde{F} \in \tilde{S} \), relation \( R \in \{A, B\} \), and R-request \( \alpha \in \text{Req}_R(F) \)

do if not \( \text{fulfilled}[F, R, \alpha] \)
then return false

return true

main

\( \tilde{y}_0 \) ← any value in \( \{1, \ldots, 2^{2^{|\varphi|}} - 1\} \)

\( \tilde{y} \) ← any value in \( \{1, \ldots, 2^{2^{|\varphi|}} \cdot 2^{2^{|\varphi|}} - 1\} \)

\( \tilde{F} \) ← any \( \varphi \)-atom such that \( \text{Req}_B(F) = \emptyset \) and \( \varphi \in \text{Obs}(F) \cup \text{Req}_B(F) \)

\( S \) ← \( \{F\} \)

for \( y \) ← \( \tilde{y}_0 \) to \( \tilde{y}_0 \)

do if not \( \text{CheckConsistency}(S, g, G) \)
then return false

\( \tilde{S} \) ← \( S \)

\( \tilde{f} \) ← the identity function on \( \tilde{S} \)

for each \( \varphi \)-atom \( \tilde{F} \in \tilde{S} \), relation \( R \in \{A, B\} \), and R-request \( \alpha \in \text{Req}_R(F) \)

do \( \text{fulfilled}[F, R, \alpha] \) ← false

for \( y \) ← \( \tilde{y}_0 + 1 \) to \( \tilde{y}_0 + \tilde{y} \)

do if not \( \text{CheckConsistency}(S, g, G) \)
then return false

\( \tilde{f} \) ← \( f \circ \tilde{f} \)

\( S \) ← \( \{f(F) : F \in S\} \cup \{G\} \)

UpdateFulfillment\((fulfilled, \tilde{S}, \tilde{f}, S, G)\)

return CheckFulfillment\((fulfilled, \tilde{S}, \tilde{f}, S)\)

Figure 7.7: Algorithm for the satisfiability problem over infinite structures.
In this chapter, we show that the addition of the modality (\( \langle \overline{A} \rangle \)) to \( \overline{A} \overline{B} \overline{B} \) drastically changes the characteristics of the logic. First, decidability is preserved (only) if \( \overline{A} \overline{B} \overline{B} \) is interpreted over finite linear orders, but there is a non-elementary blow up in complexity: the satisfiability problem is not primitive recursive anymore ([46]). Moreover, we show that the addition of any modality in the set \( \{ \langle D \rangle, \langle \overline{D} \rangle, \langle E \rangle, \langle \overline{E} \rangle, \langle O \rangle, \langle \overline{O} \rangle \} \) (modalities \( \langle O \rangle, \langle \overline{O} \rangle \) correspond to Allen’s “overlaps/overlapped by” relations) to \( \overline{A} \overline{B} \overline{B} \) leads to undecidability. This allows us to conclude that \( \overline{A} \overline{B} \overline{B} \), interpreted over finite linear orders, is maximal with respect to decidability. Next, we prove that the satisfiability problem for \( \overline{A} \overline{B} \overline{B} \) becomes undecidable when it is interpreted over any class of linear orders that contains at least one linear order with an infinitely ascending sequence, thus including the natural time flows \( \mathbb{N}, \mathbb{Z}, \mathbb{Q}, \) and \( \mathbb{R} \). As matter of fact, we prove that the addition of \( B \) to \( \overline{A} \overline{B} \) suffices to yield undecidability (the proof can be easily adapted to the case of \( \overline{B} \)). Paired with undecidability results in [10, 14], this shows the maximality of \( \overline{A} \overline{B} \) with respect to decidability when interpreted over these classes of linear orders.

### 8.1 The interval temporal logic \( \overline{A} \overline{B} \overline{B} \)

In this section, we extend the definitions given in Section 7.1 in order to deal with the \( \overline{A} \) operator. The syntax and the semantics of the operators \( \overline{A}, \overline{B} \) and \( \overline{B} \) are the same of the ones provide in Subsection 7.1.1. We interpret formulas of \( \overline{A} \overline{B} \overline{B} \) in interval temporal structures over finite linear orders with the relations “meets”, “met by”, “begins”, and “begun by”. Precisely, given \( N \in \mathbb{N} \), and \( I_N \) as the set of all (non-singleton) closed intervals \([x, y]\), with \( 0 \leq x < y \leq N \) for any pair of intervals \([x, y], [x', y'] \in I_N \), we introduce the new Allen’s relation “met by” \( \overline{A} \) defined as follows:

- “\( \text{met by} \)”:
  \[ [x, y] \overline{A} [x', y'] \text{ iff } x = y'. \]

Given an interval structure \( S = (I_N, A, B, \overline{B}, \sigma) \), where \( \sigma : I_N \rightarrow \mathcal{P}(\text{Prop}) \) is a labeling function that maps intervals in \( I_N \) to sets of propositional variables, and an initial interval \( I \), we define the semantics of \( \langle \overline{A} \rangle \) as follows:

- \( S, I \vDash \langle \overline{A} \rangle \phi \) iff there is an interval \( J \in I_N \) such that \( I \overline{A} J \) and \( S, J \vDash \phi \).

The satisfiability problem for \( \overline{A} \overline{B} \overline{B} \) is analogous of the one introduced in Subsection 7.1.1. Given an \( \overline{A} \overline{B} \overline{B} \) formula \( \phi \) the notions of \( \phi \)-atom and \( \phi \)-type are the same of the ones proposed in 7.1.2 as well as the definitions of \( \mathcal{L}(\phi) \) and \( \mathcal{L}^+(\phi) \).

Note that the cardinalities of both sets \( \mathcal{L}(\phi) \) and \( \mathcal{L}^+(\phi) \) still remain linear in the number \( |\phi| \) of subformulas of \( \phi \). We have \(|\mathcal{L}(\phi)| = 2|\phi|, |\mathcal{L}^+(\phi)| = 18|\phi|\), and there are at most \( 2^{|\phi|} \) distinct atoms.
We redefine the relations between atoms , , and the , in order to deal with the ( ) operator:

\[
\begin{align*}
F, \Delta \rightarrow G & \quad \text{iff} \quad \begin{cases}
\mathcal{R}_{\mathcal{A}}(F) = \mathcal{O}_{bs}(G) \cup \mathcal{R}_{\mathcal{A}}(G) \\
\mathcal{O}_{bs}(F) \subseteq \mathcal{R}_{\mathcal{A}}(G)
\end{cases} \\
F, \Leftrightarrow \rightarrow G & \quad \text{iff} \quad \begin{cases}
\mathcal{R}_{\mathcal{B}}(F) = \mathcal{O}_{bs}(G) \cup \mathcal{R}_{\mathcal{B}}(G) \\
\mathcal{R}_{\mathcal{B}}(G) = \mathcal{O}_{bs}(F) \cup \mathcal{R}_{\mathcal{B}}(F).
\end{cases}
\end{align*}
\]

Note that the above relations preserve a view-to-type dependency, namely, for every pair of intervals \( I = [x, y] \) and \( I' = [x', y'] \), we have

\[
x' = y \quad \land \quad y' = y + 1 \quad \text{implies} \quad \text{Type}_{\mathcal{A}}(I), \Delta \rightarrow \text{Type}_{\mathcal{A}}(I')
\]

\[
x' = x \quad \land \quad y' = y - 1 \quad \text{implies} \quad \text{Type}_{\mathcal{B}}(I), \Leftrightarrow \rightarrow \text{Type}_{\mathcal{B}}(I').
\]

The logic \( \mathbb{A}\mathbb{B}\mathbb{B} \) can be equivalently interpreted over the so-called compass structures [61] introduced in Subsection 7.1.3. We recall that such an alternative interpretation exploits the existence of a natural bijection between the intervals introduced in subsection – redefining the relations between atoms.

Definition 8.1.1. Given an \( \mathbb{A}\mathbb{B}\mathbb{B} \) formula \( \varphi \), a (finite, consistent, and fulfilling) compass (\( \varphi \)-)structure of length \( N \in \mathbb{N} \) is a pair \( \mathcal{G} = (\mathcal{P}_N, \mathcal{L}) \), where \( \mathcal{P}_N \) is the set of points \( p = (x, y) \), with \( 0 \leq x < y \leq N \), and \( \mathcal{L} \) is function that maps any point \( p \in \mathcal{P}_N \) to a (\( \varphi \)-)atom \( \mathcal{L}(p) \) in such a way that

- for every relation \( R \in \{A, \overline{A}, B, \overline{B}\} \) and every pair of points \( p, q \in \mathcal{P}_N \) such that \( p R q \), we have \( \mathcal{O}_{bs}(\mathcal{L}(q)) \subseteq \mathcal{R}_R(\mathcal{L}(p)) \) (consistency);
- for every relation \( R \in \{A, \overline{A}, B, \overline{B}\} \), every point \( p \in \mathcal{P}_N \), and every formula \( \alpha \in \mathcal{R}_R(\mathcal{L}(p)) \), there is a point \( q \in \mathcal{P}_N \) such that \( p R q \) and \( \alpha \in \mathcal{O}_{bs}(\mathcal{L}(q)) \) (fulfillment).

It is easy to see that the (finite, consistent, and fulfilling) compass structures are exactly those structures \( \mathcal{G} = (\mathcal{P}_N, \mathcal{L}) \), with \( N \in \mathbb{N} \), that satisfy the following conditions for all pair of points \( p, q \) in \( \mathcal{G} \):

i) if \( p = (x, y) \) and \( q = (y, y + 1) \), then \( \mathcal{L}(p), \Delta \rightarrow \mathcal{L}(q) \);
ii) if \( p = (x, y) \) and \( q = (x, y + 1) \), then \( \mathcal{L}(q), \Leftrightarrow \rightarrow \mathcal{L}(p) \);
iii) if \( p = (y - 1, y) \), then \( \mathcal{R}_R(\mathcal{L}(p)) = \bigcup_{0 \leq x < y - 1} \mathcal{O}_{bs}(\mathcal{L}(x, y - 1)) \);
Decidability and complexity of the satisfiability problem for $\mathsf{A\bar{A}B\bar{B}}$ over finite linear orders

In this section, we prove that the satisfiability problem for $\mathsf{A\bar{A}B\bar{B}}$ interpreted over finite linear orders is decidable, but not primitive recursive. In order to do that, we use a technique similar to the one proposed in subsection 7.2.1, namely, we fix a formula $\varphi$ and a finite compass structure $\mathcal{G} = (\mathcal{P}, \mathcal{L})$ satisfying $\varphi$ and we show that, under suitable conditions, $\mathcal{G}$ can be reduced in length while preserving the existence of atoms featuring $\varphi$. For the sake of brevity, we call contraction the operation that reduces the length of a given compass structure $\mathcal{G}$ while preserving the existence of atoms featuring $\varphi$. Such an operation has been introduced in its simple variant in Lemma 7.2.2 and it precisely consists of removing the portion of the compass structure $\mathcal{G}$ included between two distinguished rows $y_0$ and $y_1$ and selecting a subset of atoms from the upper row $y_1$ that match with the atoms of the lower row $y_0$. Hereafter, we refer the reader to Figure 8.2 for an intuitive account of the contraction operation (the colored nodes represent the atoms associated with the points of $\mathcal{G}$). According to the definition of Lemma 7.2.2, the contraction operation is applicable whenever the set of atoms of the lower row $y_0$ is included in the set of atoms of the upper row $y_1$ (the arrows in Figure 8.2 represent a matching function $f$ between the atoms of the lower row $y_0$ and the atoms of the upper row $y_1$). Such a condition on the set of atoms associated with the rows $y_0$ and $y_1$ guarantees the correctness of the contraction operation with respect to the definition of consistent and fulfilling compass structure, provided that the use of the modal operator $\langle \mathcal{A} \rangle$ is avoided. However, in the presence of the modal operator $\langle \mathcal{A} \rangle$, things get more involved, since some points $p = (x, y_1)$ from the upper row $y_1$ (e.g., the one labeled by $F_4$ in Figure 8.2) might be necessary in order to fulfill the $\mathcal{A}$-requests enforced by other points $p' = (x', y')$, with $x' = y_1$ and $y' > y_1$. In the following, we describe a suitable variant of the contraction operation which is applicable to models of $\mathsf{A\bar{A}B\bar{B}}$ formulas.

Let us fix an $\mathsf{A\bar{A}B\bar{B}}$ formula $\varphi$ that is featured by a finite compass structure $\mathcal{G} = (\mathcal{P}, \mathcal{L})$. Without loss of generality, we can assume that $\varphi$ is of the form $(\varphi \land [B] \bot) \lor (\langle \mathcal{B} \rangle \varphi) \lor (\langle \mathcal{B} \rangle \langle \mathcal{A} \rangle \varphi)$ and, furthermore, it belongs to the atom associated with the point $p = (0, 1)$ at the bottom of the
structure $\mathcal{S}$. Before turning to our main result, we need to introduce some preliminary notation and terminology.

For every $1 \leq y \leq N$, we denote by $Row_2(y)$ the row $y$ of $\mathcal{S}$, namely, the set of all points $p = (x, y)$ of $\mathcal{S}$. We associate with each row $y$ the set $Shading_2(y) = \mathcal{L}(Row_2(y))$, which consists of the atoms associated with the points in $Row_2(y)$. Clearly, for every pair of atoms $F, G$ in $Shading_2(y)$, we have $Req_A(F) = Req_A(G)$. We also associate with the row $y$ the function $Count_2(y)$, which maps an atom $F$ to the number $Count_2(y)(F)$ of $F$-labeled points in $Row_2(y)$.

In order to deal with $\mathcal{A}$-requests, we need to introduce the notion of cover of a compass structure. Intuitively, this is a selection of points that fulfills all $\mathcal{A}$-requests coming from other points (hence the points in a cover should not disappear during the operation of contraction). Formally, a cover of a compass structure $\mathcal{S} = (\mathbb{P}_N, \mathcal{L})$ is a subset $C$ of $\mathbb{P}_N$ that satisfies the following two conditions:

- if $(x, y) \in C$ and $x < y - 1$, then $(x, y - 1) \in C$ as well;
- for every point $q = (y - 1, y) \in \mathbb{P}_N$, the set $\mathcal{R}(\mathcal{A}(q))$ coincides with the union of the sets $\mathcal{O}(\mathcal{L}(p))$ for all $p = (x, y - 1) \in C$.

Given a cover $C$ of $\mathcal{S}$, we extend the notations $Row_2(y)$, $Shading_2(y)$, and $Count_2(y)$ respectively to $Row_{2|C}(y)$, $Shading_{2|C}(y)$, and $Count_{2|C}(y)$, having the obvious meaning (e.g., $Row_{2|C}(y)$ is the set of all points of $\mathcal{S}$ along the row $y$ that also belong to $C$). Moreover, we say that a cover is minimal if it does not include properly any other cover. We can easily verify that every minimal cover $C$ of $\mathcal{S} = (\mathbb{P}_N, \mathcal{L})$ satisfies

$$\begin{align*}
Row_{2|C}(N) & = \emptyset \\
|Row_{2|C}(y)| - 1 & \leq |Row_{2|C}(y - 1)| \leq |Row_{2|C}(y)| + |\mathcal{A}|. 
\end{align*} \tag{8.1}$$

The following proposition shows that, under suitable conditions, a given compass structure $\mathcal{S}$ can be reduced in length while preserving the existence of atoms featuring $\mathcal{A}$. Note that such a result can be thought of as a strengthening of the original “contraction lemma” for structures over the signature $A, B, B$ (indeed, if the logic does not allow the modal operator $\mathcal{A}$, then the empty set is the unique minimal cover of any compass structure $\mathcal{S}$ and hence the proposition below becomes equivalent to Lemma 7.2.2). For the sake of brevity, hereafter we use $\subseteq$ to denote the componentwise partial order between functions that map atoms to natural numbers, i.e., $f \subseteq g$ iff $f(F) \leq g(F)$ holds for all atoms $F$.

**Proposition 8.2.1.** Let $\mathcal{S} = (\mathbb{P}_N, \mathcal{L})$ be a compass structure that features a formula $\mathcal{A}$ in its bottom row. If there exist a cover $C$ of $\mathcal{S}$ and two rows $y_0$ and $y_1$ in $\mathcal{S}$, with $1 < y_0 < y_1 \leq N$, such that

i) $Shading_2(y_0) \subseteq Shading_2(y_1)$,

ii) $Count_2(y_0) \geq Count_{2|C}(y_1)$,

then there exists a cover $\mathcal{S}'$ of length $N' < N$ that features $\mathcal{A}$.

**Proof.** Suppose that $C$ is a cover of $\mathcal{S}$ and that $1 < y_0 < y_1 \leq N$ are two rows satisfying the hypothesis of the proposition. Then we know that there is a function $f : [0, ..., y_0 - 1] \rightarrow [0, ..., y_1 - 1]$ such that

i) for every point $p = (x, y_0)$ along the row $y_0$, the corresponding point $q = (f(x), y_1)$ along the row $y_1$ satisfies $\mathcal{L}(q) = \mathcal{L}(p)$;

ii) for every point $q = (x', y_1)$ along the row $y_1$ that also belongs to the cover $C$, there is a point $p = (x, y_0)$ along the row $y_0$ such that $f(x) = x'$ (and hence, from the previous property, $\mathcal{L}(q) = \mathcal{L}(p)$).

Let $k = y_1 - y_0$, $N' = N - k$ ($< N$), and $\mathbb{P}_{N'}$ be the portion of the grid that consists of all points $p = (x, y)$, with $0 \leq x < y \leq N'$. We extend the above function $f$ to a function that maps points in $\mathbb{P}_{N'}$ to points in $\mathbb{P}_N$ as follows:

- if $p = (x, y)$, with $0 \leq x < y < y_0$, then we simply let $f(p) = p$;
- if $p = (x, y)$, with $0 \leq x < y_0 \leq y$, then we let $f(p) = (f(x), y + k)$;
if \( p = (x, y) \), with \( y_0 \leq x < y \), then we let \( f(p) = (x + k, y + k) \).

We denote by \( \mathcal{L}' \) the labeling of \( \mathbb{P}_N \) such that, for every point \( p \in \mathbb{P}_N \), \( \mathcal{L}'(p) = \mathcal{L}(f(p)) \) and we denote by \( \mathcal{G}' \) the resulting structure \( (\mathbb{P}_N, \mathcal{L}') \) (see Figure 8.2). We have to prove that \( \mathcal{G}' \) is a consistent and fulfilling compass structure that features \( \varphi \) (see Definition 8.1.1). As a preliminary remark, we recall that, by hypothesis, the bottom row of \( \mathcal{G} \), and hence the bottom row of \( \mathcal{G}' \), features the formula \( \varphi \). Moreover, since the above definition of matching function \( f \) is a specialization of the definition given in Lemma 7.2.2, the proof that \( \mathcal{G}' \) is consistent and fulfilling with respect to the relations \( A, B \), and \( \mathcal{B} \) is the exactly same as the proof of Lemma 3.2. In that proof, it is also implicitly shown that \( \mathcal{G}' \) is consistent with respect to the relation \( \mathcal{A} \).

In order to conclude the proof, it is sufficient to show that \( \mathcal{G}' \) is fulfilling with respect to the relation \( \mathcal{A} \).

**FULMEMENT of \( \tilde{A} \)-REQUESTS.** Let \( p = (x, y) \) be a point in \( \mathcal{G}' \) and let \( \alpha \) be a subformula in \( \mathcal{R}eq\mathcal{X}(\mathcal{L}'(p)) \). The following cases arise:

1. \( x < y_0 \) and \( y \leq y_0 \). In such a case, we have \( f(p) = p \) and, since \( \mathcal{G} \) is a (fulfilling) compass structure, there exists a point \( p' = (x', y') \) such that \( f(p') = p' \) and hence \( f(p) = p \). Moreover, since \( x' < x < y_0 \), we have \( f(p') = p' \) and hence \( f(p) = p \).

2. \( x < y_0 \) and \( y > y_0 \). In such a case, we define \( p' = (x', y') \), where \( x' = f(x) \) and \( y' = y + k \), in such a way that \( \mathcal{L}'(p) = \mathcal{L}(f(p')) = \mathcal{L}(p') \). By construction, we have \( \mathcal{L}(x', y_1) = \mathcal{L}(f(x, y_1)) = \mathcal{L}(x, y_0) \) and hence, from basic properties of types, \( \mathcal{R}eq\mathcal{X}(\mathcal{L}(p')) = \mathcal{R}eq\mathcal{X}(\mathcal{L}(x, y_0)) = \mathcal{R}eq\mathcal{X}(\mathcal{L}(x, y_0)) \). Now, since \( \alpha \in \mathcal{R}eq\mathcal{X}(\mathcal{L}(x, y_0)) \) and \( \mathcal{G} \) is a (fulfilled) compass structure, we know that there is a point \( p'' = (x'', y'') \) in \( \mathcal{G} \) such that \( (x, y_0) \subseteq \mathcal{A}p'' \) and \( \alpha \in \mathcal{O}bs(\mathcal{L}(p'')) \). Moreover, since \( y'' = x < y_0 \), we have \( f(p'') = p'' \), from which we obtain \( p'' \subseteq \tilde{A}p \) and \( \alpha \in \mathcal{O}bs(\mathcal{L}(p'')) \).

3. \( x \geq y_0 \) (and hence \( y > y_0 \)). In such a case, we have \( f(p) = (x + k, y + k) \) and, since \( \mathcal{G} \) is a (fulfilling) compass structure, there is a point \( p' = (x', y') \) such that \( f(p) \subseteq \tilde{A}p' \) and \( \alpha \in \mathcal{O}bs(\mathcal{L}(p')) \). Note that \( y' = x + k \geq y_1 \). We further distinguish between two subcases.

   - If \( x' \geq y_1 \), then we simply define \( p'' = (x' - k, y' - k) \) in such a way that \( p'' = f(p'') \) and hence \( p \subseteq \tilde{A}p'' \) and \( \alpha \in \mathcal{O}bs(\mathcal{L}(p'')) \). Otherwise, if \( x' < y_1 \) and \( (x', y_1) \in C \) (namely, \( (x', y_1) \) is a point inside the cover \( C \) of \( \mathcal{G} \)), then, from the properties satisfied by the function \( f \), we know that there is \( x'' < y_0 \) such that \( f(x'') = x' \). We thus define \( p'' = (x'', y' - k) \) in such a way that \( f(p'') = p' \), from which we obtain \( p \subseteq \tilde{A}p'' \) and \( \alpha \in \mathcal{O}bs(\mathcal{L}(p'')) \) and \( \mathcal{G} \subseteq \mathcal{G}' \).

Finally, if \( x' < y_1 \) but \( (x', y_1) \in \mathcal{C} \), then, by definition of cover, we know that there exists another point \( (x'', y_1) \) along the row such that \( \alpha \in \mathcal{O}bs(\mathcal{L}(x'', y_1)) \), and we can then use an argument similar to the previous case to devise the fulfillment of the \( \tilde{A} \)-request \( \alpha \) in \( \mathcal{G}' \).

On the grounds of Proposition 8.2.1, it makes sense to restrict ourselves to the **minimal** models of \( \varphi \) and, in particular, to those compass structures \( \mathcal{G} = (\mathbb{P}_N, \mathcal{L}) \) that feature \( \varphi = (\mathcal{G} \land (\mathbb{B}) \varphi) \lor ((\mathbb{B}) \varphi) \lor ((\mathbb{B}) \varphi) \) in the bottom row and that cannot be contracted. The above argument leads to a non-deterministic procedure that decides whether a given formula \( \varphi \) is satisfied by a (contraction-free) interval structure \( \mathcal{G} \). The pseudo-code of such an algorithm is given in Figure 8.3: the variable \( \Delta \) represents the value \( N - y + 1 \), where \( N \) is the length of the model \( \mathcal{G} \) to be guessed and \( y \) is the current row (note that we cannot use \( y \) in place of \( \Delta \) since there is no a priori bound on the length \( N \) of the model), the variable \( F_\Delta \) represents the atom associated with the rightmost point \( p = (y - 1, y) \) along the current row \( y \), the variable \( S_\Delta \) represents an over-approximation of the set \( Shadings_\Delta(y) \), and the variable \( C_\Delta \) represents the function \( Count_{\mathcal{G},\mathcal{C}}(y) \) for a suitable cover \( \mathcal{C} \) of \( \mathcal{G} \) (note that the content of such a variable can be guessed because the sum of its values is bounded in virtue of Equation 8.1).

The decidability of the satisfiability problem for \( \mathcal{A}\mathcal{B}\mathcal{B} \) interpreted over finite linear orders is thus reduced to a proof of termination, soundness, and completeness for the algorithm given in Figure 8.3 as formally stated by Theorem 8.2.3 (its proof is reported in the Appendix). As a matter of fact, termination relies on the following crucial lemma, which is often attributed to Dickson.
Lemma 8.2.2 (Dickson’s Lemma). Let \( (\mathbb{N}^k, \leq) \) be the \( k \)-dimensional vector space over \( \mathbb{N} \) equipped with the componentwise partial order \( \leq \). Then, \( (\mathbb{N}^k, \leq) \) admits no infinite anti-chains, namely, every subset of \( \mathbb{N}^k \) that consists of pairwise \( \leq \)-incomparable vectors must be finite.

Theorem 8.2.3. The satisfiability problem for \( \mathbb{AAB} \), interpreted over finite linear orders, is decidable.

Proof. We prove that the non-deterministic algorithm described in Figure 8.3 terminates on every input formula \( \phi \) and it returns \( \text{true} \) iff \( \phi \) is satisfied by some finite interval structure. It is convenient to divide the proof into three parts: first, we prove termination (i.e., every computation of the algorithm terminates), then soundness (i.e., if \( \phi \) is a computation of the algorithm that returns \( \text{true} \) on \( \phi \), then \( \phi \) is satisfiable), and finally completeness (i.e., if \( \phi \) is satisfiable, then there is a computation on \( \phi \) that returns \( \text{true} \)).

Termination. Let \( \phi \) be an input formula and suppose, by way of contradiction, that there is a non-terminating computation of the algorithm. In particular, this means that the function \( \text{CheckContraction} \) returns \( \text{false} \) on all sequences of arguments \( F_1, S_1, C_1, ..., F_\lambda, S_\lambda, C_\lambda \). Therefore, for all pair of positive natural numbers \( \lambda < \lambda' \), one of the following conditions must hold:

1. \( S_\lambda \cup \{ F_\lambda \} \not\subseteq S_{\lambda'} \cup \{ F_{\lambda'} \} \).
2. \( C_\lambda \not\subseteq C_{\lambda'} \).

We now recall that there only exist finitely many distinct \( \varphi \)-atoms and hence finitely many distinct sets \( S_\lambda \). This implies that there is an infinite sequence of indices \( \lambda_1 < \lambda_2 < ... \) such that, for all \( i > i' \), \( S_{\lambda_i} = S_{\lambda_i} \), and hence, by previous assumptions, \( C_{\lambda_i} \not\subseteq C_{\lambda_i} \). Similarly, since every function \( C_\lambda \) dominates (with respect to the componentwise partial order \( \leq \)) only finitely many functions \( C_\lambda \), with \( i' < i \), we can find an infinite subsequence \( i_1 < i_2 < ... \) of indices for which the functions \( C_{\lambda_{i_1}}, C_{\lambda_{i_2}}, ... \) (thought of as vectors in the \( k \)-dimensional space \( \mathbb{N}^k \)) turn out to be pairwise \( \leq \)-incomparable. This is in contradiction with Lemma 8.2.2 and therefore the algorithm must terminate.

Soundness. We consider a successful computation of the algorithm on a formula \( \phi \) and we show that there is a finite compass structure \( \mathcal{G} = (\mathbb{N}, L) \) that features \( \varphi \), where the length \( N \) coincides with the value of the variable \( \Delta \) at the end of the computation. For every \( 1 \leq \Delta \leq N \), we denote by
8.2. Decidability and complexity of the satisfiability problem for $\overline{\exists A \forall B \forall}$ over finite linear orders

$F_\Delta$, $S_\Delta$, and $C_\Delta$ the content of the omonymous variables which are guessed during the computation. Moreover, we use $y$ (resp., $y-1$) as a shorthand for the value $N-\Delta+1$ (resp., $N-(\Delta+1)-1$).

Below, we specify the atom $L(x,y)$ associated with each point $p=(x,y)$ of the compass structure $\mathbb{S} = (P_N,L)$ by exploiting an indiction on $y = N-\Delta+1$ (that is, starting from the lower rows and going upward). While doing this, we also build a cover $C$ of $\mathbb{S}$ in such a way that the two conditions $Shadings_\mathbb{S}(y) \subseteq S_\Delta \cup \{F_\Delta\}$ and $\text{Count}_{\mathbb{S} \cap C}(y)(F) = C_\Delta(F)$ are guaranteed for every row $y = N-\Delta+1$ and every atom $F$. Let us consider a point $p = (x,y)$, with $0 \leq x < y \leq N$ and $y = N-\Delta+1$:

- If $x = y-1$, then we let $L(p) = F_\Delta$. Moreover, we let $p$ belong to the set $C$ iff $C_\Delta(F_\Delta) = 1$ (we can shortly write $[C \cap \{p\}] = C_\Delta(F_\Delta)$). Note that, when $y = 1$, we have $Shadings_\mathbb{S}(y) = \{F_\Delta\} \subseteq S_\Delta \cup \{F_\Delta\}$, and $\text{Count}_{\mathbb{S} \cap C}(y)(L(p)) = C_\Delta(F_\Delta) \leq 1 = \text{Count}_{\mathbb{S} \cap C}(y)(L(p))$.

- If $x < y-1$, then, by exploiting the inductive hypothesis, we assume that both $L(q)$ and $C \cap \{q\}$ are specified for all points $q = (x',y-1)$ along the row $y-1$ and we accordingly define $L(p)$ and $C \cap \{p\}$, as follows. First, we denote by $f : S^+_{\Delta+1} \to S_\Delta$ and $g : M_\Delta \to M^+_{\Delta+1}$ the two functions that have been guessed during the execution of the procedure CheckRows on arguments $(F_\Delta,S_\Delta,C_\Delta,F_{\Delta+1},S_{\Delta+1},C_{\Delta+1})$ (the sets $S^+_{\Delta+1}$, $M_\Delta$, and $M^+_{\Delta+1}$ are defined as in the body of the procedure). Then, given an atom $F$, we shortly denote by $C^F_{y-1}$ the set of all F-labeled points that lie along the row $y-1$ and belong to the cover $C$. From the inductive hypothesis, we know that $\{C^F_{y-1} = \text{Count}_{\mathbb{S} \cap C}(y-1)(F) = C_{\Delta+1}(F)\}$ and hence, by construction, there is a bijection $h^F_{y-1}$ from the set $C^F_{y-1}$ to the set of all pairs $(F,i)$ in $M^+_{\Delta+1}$, with $1 \leq i \leq C_{\Delta+1}(F)$ (we fix a unique bijection $h^F_{y-1}$ for each row $y-1$ and for each atom $F$). We now let $q = (x,y-1)$ (i.e., the point just below $p$) and we distinguish between two cases, depending on whether $g^{-1}(h^F_{y-1}(q))$ is defined or not (recall that the inverse $g^{-1}$ of the injective function $g$ is a partial surjective function from $M^+_{\Delta+1}$ to $M_\Delta$). If $g^{-1}(h^F_{y-1}(q))$ is defined and equal to the pair $(F',t') \in M_\Delta$, we let $L(p) = F'$ and $p \in C$. Otherwise, if $g^{-1}(h^F_{y-1}(q))$ is not defined, then we let $L(p) = f(L(q))$ and $p \notin C$. Note that, if we apply the above definitions of $L(p)$ and $C \cap \{p\}$ for all points $p$ along the same row $y$, we then obtain $Shadings_\mathbb{S}(y) \subseteq S_\Delta \cup \{F_\Delta\}$ and $\text{Count}_{\mathbb{S} \cap C}(y)(F) = C_\Delta(F)$ for all atoms $F$. By exploiting the fact that every call to the procedure CheckConsistency is successful, we can easily verify that, for every pair of points $p, q \in \mathbb{S}$, the following conditions hold:

i) if $p = (x,y)$ and $q = (y,y+1)$, then $L(p) : \Delta \to L(q)$;

ii) if $p = (x,y)$ and $q = (x,y+1)$, then $L(q) \subseteq L(p)$;

iii) if $p = (y-1,y)$, then $\text{Req}(\mathbb{L}(p)) = \bigcup_{0 \leq x < y-1} \text{Obs}(L(x,y-1))$;

iv) if $p = (x,N)$, then $\text{Req}(\mathbb{L}(p)) = \emptyset$ and $\text{Req}(\mathbb{L}(p)) = \emptyset$.

This shows that $\mathbb{S}$ is a (consistent and fulfilling) compass structure that features $\varphi$ in its bottom row. Therefore, by Proposition 8.1.2, we can conclude that the input formula $\varphi$ is satisfied over a finite interval structure.

Completeness. As for completeness, we consider a finite labeled interval structure $\mathbb{S} = (I_N,A,\overline{A},B,\overline{B},\sigma)$ that satisfies $\varphi$. By Proposition 8.1.2, we know that there is a (consistent and fulfilling) compass structure $\mathbb{S} = (I_N,L)$ that features the formula $\varphi = (\varphi \land \overline{[B]}) \lor (\overline{([B]\varphi) \lor ([B]A)\varphi})$ in its bottom row. Let us also fix a minimal cover $C$ of $\mathbb{S}$. We can exploit the existence of $\mathbb{S}$ and $C$ to devise the existence of a successful computation of the algorithm. Precisely, we let the guessed contents for the variables $F_\Delta$, $S_\Delta$, and $C_\Delta$ be, respectively, the atom $L(p)$ associated with the rightmost point $p = (y-1,y)$ along the row $y = N-\Delta+1$, the set of atoms associated with the non-rightmost points $p = (x,y)$, with $x < y-1$, along the same row $y = N-\Delta+1$, and the function $\text{Count}_{\mathbb{S} \cap C}(y)$ that maps every atom $F$ to the number of $F$-labeled points along the row $y$ that also belong to the cover $C$. On the grounds of Equation 8.1, it is clear that the above defined values can be correctly guessed at each iteration of the main loop. Moreover, for each call to the procedure CheckRows with arguments $(F_\Delta,S_\Delta,C_\Delta,F_{\Delta+1},S_{\Delta+1},C_{\Delta+1})$, we assume that the variables $f$ and $g$ are guessed as follows:
\begin{itemize}
  \item $f$ is any function between atoms such that, for every $F \in S_{\Delta+1} \cup \{F_{\Delta+1}\}$, there exist two points $p = (x, y-1)$ and $q = (x, y)$, with $0 \leq x < y - 1$, satisfying $L(p) = F$ and $L(q) = f(F)$ (note that such a function $f$ exists since, by construction, $F \in S_{\Delta+1} \cup \{F_{\Delta-1}\} = \text{Shadings}_q(y)$ and $f(F) \in S_{\Delta} \cup \{F_{\Delta}\} = \text{Shadings}_q(y)$, where $y = N - \Delta + 1$);
  \item $g$ is any injective function from $M_\Delta = \{(F, i) : F \in S_\Delta, 1 \leq i \leq C_\Delta(F)\}$ to $M^+_{\Delta+1} = \{(F, i) : F \in S^+_{\Delta+1}, 1 \leq i \leq C_{\Delta+1}(F)\}$ such that, for every pair $(F, i) \in M_\Delta$, the cover $C$ contains two points $p = (x, y)$ and $q = (x, y-1)$ satisfying $L(p) = F$ and $L(q) = F'$, with $g(F, i) = (F', i')$ (note that such an injective function $g$ exists since, by construction, $C_\Delta(F) = \text{Count}_{\{i\}}(y)|F|$ and $C_{\Delta+1}(F') = \text{Count}_{\{i\}}(y-1)|F')$).
\end{itemize}

The above definitions guarantee that every call to the procedure $\text{CheckRows}$ terminates by returning $\text{true}$. As for the calls to the procedure $\text{CheckContraction}$, we can assume, without loss of generality, that $q$ has minimal length. In particular, by Proposition 8.2.1, this means that, for every pair of rows $y = N - \Delta + 1$ and $y' = N - \Delta' + 1$, with $1 \leq \Delta' < \Delta < N$ (hence $S_\Delta \neq \emptyset$), at least one of the following two conditions holds:

1. $S_\Delta \cup \{F_\Delta\} = \text{Shadings}_q(y) \not\subseteq \text{Shadings}_q(y') = S_\Delta \cup \{F_{\Delta'}\}$,
2. $C_\Delta = \text{Count}_{\{i\}}(y) \not\subseteq \text{Count}_{\{i\}}(y') = C_{\Delta'}$.

This immediately implies that every call to the procedure $\text{CheckContraction}$ terminates by returning $\text{true}$. Finally, since the algorithm terminates, the formula $\phi$ must belong to the atom $F_N$ associated with the point $p = (0, 1)$ of $S$. We have just shown that there is a successful computation of the algorithm.

We conclude the section by analyzing the complexity of the satisfiability problem for $\mathbb{A}\mathbb{B}\mathbb{B}$. In Section 7.3, we show that the satisfiability problem for $\mathbb{A}\mathbb{B}$ is EXPSPACE-complete. Here we prove that, quite surprisingly, the satisfiability problem for $\mathbb{A}\mathbb{B}\mathbb{B}$ (in fact, also that for the fragment $\mathbb{A}\mathbb{B}$) has much higher complexity, precisely, it is not primitive recursive. The proof is based on a reduction from the reachability problem for lossy counter machines, which is known to have strictly non-primitive recursive complexity [54], to the satisfiability problem for $\mathbb{A}\mathbb{B}\mathbb{B}$. In particular, it shows that there is an $\mathbb{A}\mathbb{B}$ formula that defines a set of encodings of all possible computations of a given lossy counter machine. The key ingredients of the proof are as follows. First, we represent the value $c(t)$ of each counter $c$, at each instant $t$ of a computation, by means of a set consisting of exactly $c(t)$ unit-length intervals labeled by $c$. Then, we require that suitable disequalities of the form $c(t+1) \leq c(t) + h$, with $h \in \{-1, 0, 1\}$, hold between the values of the counter $c$ at consecutive time instants. This can be done by enforcing the existence of a surjective partial function $q$ from the set of $c$-labeled unit-length intervals corresponding to the time instant $t$ to the set of $c$-labeled unit-length intervals corresponding to the next time instant $t+1$. Finally, we exploit the fact that surjective partial functions between sets of unit-length intervals can be specified in the logic $\mathbb{A}\mathbb{B}$.

**Theorem 8.2.4.** The satisfiability problem for $\mathbb{A}\mathbb{B}$, and hence that for $\mathbb{A}\mathbb{B}\mathbb{B}$, interpreted over finite linear orders, is not primitive recursive.

**Proof.** We first give a precise notion of lossy (Minsky) counter machine. This is a triple of the form $A = (Q, k, \delta)$, where $Q$ is a finite set of control states, $k$ is the number of counters (whose values range over $\mathbb{N}$), and $\delta$ is a function that maps each state $q \in Q$ to a transition rule having one of the following forms:

1. \( i \leftarrow i + 1; \ \text{goto} \ q', \) for some $1 \leq i \leq k$ and $q' \in Q$, meaning that, whenever $A$ is at state $q$, it increments the counter $i$ and it moves to state $q'$;
2. \( \text{if} \ i = 0 \ \text{then} \ \text{goto} \ q' \ \text{else} \ i \leftarrow i - 1; \ \text{goto} \ q'' ,\) for some $1 \leq i \leq k$ and $q', q'' \in Q$, meaning that, whenever $A$ is at state $q$ and the value of the counter $i$ is $0$ (resp., greater than $0$), it moves to state $q'$ (resp., it decrements the counter $i$ and it moves to state $q''$).

In addition, from each configuration $(q, z) \in Q \times \mathbb{N}^k$, a lossy counter machine $A$ can nondeterministically activate an internal (lossy) transition and move to a configuration $(q, z')$, with $z' \leq z$ (the relation $\leq$ is defined componentwise on the values of the counters, as in Lemma 8.2.2).
8.2. Decidability and complexity of the satisfiability problem for $\mathbb{A}\mathbb{B}\overline{\mathbb{B}}$ over finite linear orders

A computation of $\mathcal{A}$ is any sequence of configurations that respects the obvious semantics of the transition relation. The reachability problem for a lossy counter machine $\mathcal{A} = (Q, k, \delta)$ consists of deciding, given two configurations $(q_{\text{source}}, z_{\text{source}})$ and $(q_{\text{target}}, z_{\text{target}})$, whether or not there is a computation that takes $\mathcal{A}$ from $(q_{\text{source}}, z_{\text{source}})$ to $(q_{\text{target}}, z_{\text{target}})$. Below, we show how to reduce the reachability problem for lossy counter machines to the satisfiability problem for the logic $\mathbb{A}\mathbb{B}\overline{\mathbb{B}}$.

Let us fix a lossy counter machine $\mathcal{A} = (Q, k, \delta)$ together with a source configuration $(q_{\text{source}}, z_{\text{source}})$ and a target configuration $(q_{\text{target}}, z_{\text{target}})$. Without loss of generality, we can assume that $z_{\text{source}} - z_{\text{target}} = 0 = (0, \ldots, 0)$ (indeed, if this were not the case, we can modify $\mathcal{A}$ by introducing some fresh control states $p_0, p_1, \ldots, p'_0, p'_1, \ldots$, some increment-transitions that take the machine from $(p_0, 0)$ to $(q_{\text{source}}, z_{\text{source}})$, and some decrement-transitions that take the machine from $(q_{\text{target}}, z_{\text{target}})$ to $(p'_0, 0)$). Moreover, we can assume that $q_{\text{target}}$ is a sink state, namely, the only state accessible from $q_{\text{target}}$ is $q_{\text{target}}$ itself.

We first show how to encode a generic computation $(q_1, z_1) \ldots (q_n, z_n)$ of $\mathcal{A}$ into an interval structure $S = ([N, A, \overline{A}, B, \overline{B}, \sigma])$. To do that, we first introduce $|Q| + k$ propositional variables that label unit-length intervals (i.e., intervals of the form $[x, x + 1]$): the first $|Q|$ propositional variables will be identified with the control states of $\mathcal{A}$, while the last $k$ propositional variables, denoted $c_1, \ldots, c_k$, will be identified with the $k$ counters of $\mathcal{A}$. Then we divide the underlying domain $[0, \ldots, N]$ of the interval structure $S$ into exactly $n + 2$ intervals $I_0 = [0, x_1], I_1 = [x_1, x_2], \ldots, I_n = [x_n, x_{n+1}]$, $I_{n+1} = [x_{n+1}, N]$, with $1 = x_1 < \ldots < x_n < N - 1$ and $x_{n+1} - x_n = 1 + \sum_{1 \leq i \leq k} z_i(i)$ for all $1 \leq t \leq n$ (hence the length $N$ of the interval structure $S$ is exactly $2 + n + \sum_{1 \leq t \leq n} \sum_{1 \leq i \leq k} z_i(i)$). The intervals $I_0, \ldots, I_n$ will be used to encode, respectively, the configurations $(q_1, z_1), \ldots, (q_n, z_n)$ of the computation of $\mathcal{A}$, while the two additional intervals $I_0$ and $I_{n+1}$ will be used to correctly move between the various intervals via the modal operators $\langle A \rangle$ and $\langle \overline{A} \rangle$. Finally, we let the labeling function $\sigma$ associate a unique propositional variable in $Q \cup \{c_1, \ldots, c_k\}$ with each unit-length subinterval of $I_t$, for all $1 \leq t \leq n$, as follows:

i) the subinterval $[x_t, x_t + 1]$ is labeled by the control state $q_t$;

ii) for every $1 \leq i \leq k$, the number of $c_i$-labeled intervals of the form $[x, x + 1]$, with $x_t < x < x_{t+1}$, coincides with the value $z_i(i)$ of the counter $i$.

(note that there may exist different encodings of the same computation of $\mathcal{A}$).

As an example, Figure 8.4 represents part of an encoding of a computation for a lossy counter machine $\mathcal{A}$ with two control states, whose occurrences are represented by black-colored and white-colored intervals, and three counters, whose values are represented by the numbers of occurrences of intervals colored, respectively, by red, blue, and green (the meaning of the dashed arrows is explained below).

The next ingredient of the reduction is the specification of all encodings of all computations of $\mathcal{A}$ by means of a suitable $\mathbb{A}\mathbb{B}\overline{\mathbb{B}}$ formula. In particular, we are interested in enforcing disequalities between counters of the form $z_{i+1}(i) \leq z_i(i) + h$, with $h \in \{-1, 0, 1\}$. We first explain how this is done in the case $h = 0$. By definition, enforcing a constraint of the form $z_{i+1}(i) \leq z_i(i)$ is equivalent to enforcing the existence of a surjective partial function $g_i$ from the set of $c_i$-labeled subintervals of $I_t$ to the set of $c_i$-labeled subintervals of $I_{t+1}$. As an example, the dashed arrow

Figure 8.4: Encoding of part a computation of a lossy counter machine.
labeled by \( g_3 \) in Figure 8.4 represents one instance of a surjective partial function representing a constraint of the form \( z_{t+1}(3) \leq z_t(3) \). In its turn, each partial function \( g_t \) can be encoded by a set of \( g_t \)-labeled intervals of the form \([x, g(x)]\), where \( g_t \) is viewed as a fresh propositional variable, \( x_t < x < x_{t+1} < g(x) < x_{t+2} \), and \( \sigma[x, x+1] = \sigma[g(x), g(x)+1] = c_t \). The relevant properties of these \( g_t \)-labeled intervals are then translated into a suitable formula \( \varphi_{t}^{\leq} \) evaluated on the interval \( I_t \). Precisely, we let:

\[
\varphi_{t}^{\leq} = \left[ B][A](g_t \rightarrow \varphi_{Q}^{\geq} \land \langle B \rangle c_i \land \langle A \rangle c_i) \land \langle B \rangle[A](\varphi_{Q}^{\geq} \land \langle A \rangle c_i \rightarrow \langle A \rangle(A)g_t) \right]
\]

where \( \varphi_{Q}^{\geq} = \langle B \rangle(\langle A \rangle \bigvee_{a \in A} a \land \langle B \rangle(\langle a \rangle \bigvee_{a \in A} a \rightarrow \langle B \rangle[A] \land_{a \in a} a) \) for any given set \( A \) (e.g., \( A = Q \)) of propositional variables (note that \( \varphi_{Q}^{\geq} \) holds at a interval \( I \) iff \( I \) contains exactly one unit-length subinterval labeled by some propositional variable \( a \in A \)). Intuitively, the first line of the formula \( \varphi_{t}^{\leq} \) enforces the condition that the set of \( g_t \)-labeled intervals that start inside \( I_t \) represent a partial function from the \( c_1 \)-labeled subintervals of \( I_t \) to the \( c_2 \)-labeled subintervals of \( I_{t+1} \), while the second line guarantees that such a partial function is surjective.

In a similar way, one can specify the constraints of the form \( z_{t+1}(i) \leq z_t(i) - 1 \) (resp., \( z_{t+1}(i) \leq z_t(i) + 1 \)) by means of a formula \( \varphi_{t}^{\leq,-1} \) (resp., \( \varphi_{t}^{\geq,+1} \)): this is done by excluding from the domain (resp., from the range) of the surjective partial function \( g_t \) exactly one \( c_1 \)-labeled subinterval of \( I_t \) (resp., \( I_{t+1} \)), which is then distinguished by using an additional propositional variable \( \text{inc} \) (resp., \( \text{dec} \)). Precisely, we let:

\[
\begin{align*}
\varphi_{t}^{\leq,-1} &= \left[ B][A](g_t \rightarrow \varphi_{Q}^{\geq} \land \langle B \rangle c_i \land \langle A \rangle c_i) \land \langle B \rangle[A](\varphi_{Q}^{\geq} \land \langle A \rangle c_i \rightarrow \langle A \rangle(A)g_t) \right] \\
\varphi_{t}^{\geq,+1} &= \left[ B][A](g_t \rightarrow \varphi_{Q}^{\geq} \land \langle B \rangle c_i \land \langle A \rangle c_i \land \langle B \rangle g_t) \right) \land \left[ B][A](\varphi_{Q}^{\geq} \land \langle A \rangle c_i \land \langle B \rangle c_i) \rightarrow \langle A \rangle(A)g_t) \right) \land \langle A \rangle(A)g_t) \right)
\end{align*}
\]

Now, we rewrite each transition rule \( \delta(q) \) of \( A \) into a formula \( \varphi_{q,0}^{\leq} \), which is defined by case analysis as follows:

1. if \( \delta(q) \) is a rule of the form \( i \leftarrow i + 1 \); goto \( q' \), then we let \( \varphi_{q,0}^{\leq} \) be the formula \( \langle A \rangle q' \land \varphi_{t}^{\leq,-1} \land \bigwedge_{i \neq t} \varphi_{i}^{\leq} \);

2. if \( \delta(q) \) is a rule of the form \( if \ i = 0 \ then \ goto \ q' \ else \ i \leftarrow i - 1 \); goto \( q'' \), then we let \( \varphi_{q,0}^{\leq} \) be \( \left( [B][A] c_i \rightarrow \varphi_{q,0}^{\leq} \right) \land \left( [B] c_i \rightarrow \varphi_{q,1}^{\leq} \right) \), where \( \varphi_{q,0}^{\leq} = \langle A \rangle q' \land \bigwedge_{1 \leq i \leq k} \varphi_{i}^{\leq} \) and \( \varphi_{q,1}^{\leq} = \langle A \rangle q'' \land \varphi_{t}^{\leq,-1} \land \bigwedge_{i \neq t} \varphi_{i}^{\leq} \).

We can specify the set of all encodings of all computations of \( A \) by means of the following formula (here we shortly denote by \( C \) the set of \( k \) propositional variables \( c_1, \ldots, c_k \)):

\[
\varphi_{A} = \left[ U \right] \left( \langle A \rangle T \land \langle A \rangle T \land [B] \rightarrow \bigvee_{a \in Q \cup C} a \land \bigwedge_{a \neq b \in Q \cup C} a \land \langle B \rangle \rightarrow \bigwedge_{a \in Q \cup C \cup \text{inc,dec}} a \right) \land \left[ U \right] \left( \langle B \rangle q \rightarrow \langle A \rangle \varphi_{q,0}^{\leq} \right) \land \left[ U \right] \left( \langle B \rangle q \rightarrow \langle A \rangle \varphi_{q,0}^{\leq} \right) \land \left[ U \right] \left( \langle B \rangle q \rightarrow \langle A \rangle \varphi_{q,0}^{\leq} \right)
\]

where \([U] \alpha \) is a shorthand for \( \alpha \land \langle A \rangle \land \bigwedge_{a \in A} a \land \bigwedge_{a \in \bar{A}} \neg a \). Intuitively, the first two lines of the formula \( \varphi_{A} \) guarantees that all
8.3 Undecidability is the rule, decidability the exception

We conclude the paper by proving that \( \mathbb{A} \oplus \mathbb{B} \), interpreted over finite linear orders, is maximal with respect to decidability. The addition of a modality for any one of the remaining Allen’s relations, that is, of any modality in the set \( \{ \{D\}, \{\overline{D}\}, \{E\}, \{\overline{E}\}, \{O\}, \{\overline{O}\} \} \), indeed leads to undecidability.

**Theorem 8.3.1.** The satisfiability problem for the logic \( \mathbb{A} \oplus \mathbb{B} \oplus \mathbb{D} \) (resp., \( \mathbb{A} \oplus \mathbb{B} \oplus \mathbb{E} \), \( \mathbb{A} \oplus \mathbb{B} \oplus \mathbb{E} \), \( \mathbb{A} \oplus \mathbb{B} \oplus \mathbb{E} \), \( \mathbb{A} \oplus \mathbb{B} \oplus \mathbb{O} \), \( \mathbb{A} \oplus \mathbb{B} \oplus \mathbb{O} \)), interpreted over finite linear orders, is undecidable.

**Proof.** First of all, we recall the definitions for the Allen’s relations “contains” \( D \), “during” \( \overline{D} \), “ended by” \( E \), “ends” \( \overline{E} \), “overlaps” \( O \), and “overlapped by” \( \overline{O} \):

- “contains”: \([x, y] D [x’, y’]\) iff \( x < x' < y < y'\);
- “during”: \([x, y] \overline{D} [x’, y’]\) iff \( x < x' < y < y'\);
- “ended by ”: \([x, y] E [x’, y’]\) iff \( x < x' \) and \( y = y'\);
- “ends”: \([x, y] \overline{E} [x’, y’]\) iff \( x < x' \) and \( y = y'\);
- “overlaps”: \([x, y] O [x’, y’]\) iff \( x < x' < y \) and \( y < y'\);
- “overlapped by”\( ”: \([x, y] \overline{O} [x’, y’]\) iff \( x < x' < y \) and \( x < y' < y\).

The semantics of the corresponding formulas \( \langle D \rangle \alpha \) (resp., \( \langle \overline{D} \rangle \alpha \), \( \langle E \rangle \alpha \), \( \langle \overline{E} \rangle \alpha \), \( \langle O \rangle \alpha \), and \( \langle \overline{O} \rangle \alpha \) is defined, as usual, for a given interval structure \( S \) and a given interval \( I \) as follows: for any relation \( R \in \{D, \overline{D}, E, \overline{E}, O, \overline{O}\} \), we write \( S, I \models (R) \alpha \) iff there is an interval \( J \in \mathbb{N} \) such that \( I \subseteq J \) and \( S, J \models \alpha \).

Since Allen’s “contains” relation \( D \) (resp., Allen’s “during” relation \( \overline{D} \)) is definable in terms of Allen’s “begun by” and “ended by” relations \( B \) and \( E \) (resp., in terms of Allen’s “begins” and “ends” relations \( \overline{B} \) and \( \overline{E} \)), to prove the theorem it is sufficient to show that the extension of \( \mathbb{A} \oplus \mathbb{B} \) with any modal operator among \( \{D\}, \{\overline{D}\}, \{O\}, \) and \( \{\overline{O}\} \) has an undecidable satisfiability problem over finite linear orders. To do that, we will reduce the (undecidable) reachability problem for non-lossy (Minsky) counter machines to the satisfiability problem for each of the relevant extensions of \( \mathbb{A} \oplus \mathbb{B} \). One can think of these reductions as slight modifications of the proof of Theorem 8.2.4, where inequalities between counter values of the form \( z_{t+1}(i) \leq z_t(i) + h \) are replaced by more restrictive constraints of the form \( z_{t+1}(i) = z_t(i) + h \). Thus, from now on, we use the same notation as in the proof of Theorem 8.2.4. Replacing inequalities of the form \( z_{t+1}(i) \leq z_t(i) + h \) by corresponding equalities \( z_{t+1}(i) = z_t(i) + h \) amounts at enforcing all partial surjective functions \( g_i \) that match \( c_i \)-labeled subintervals of \( I_t \) with \( c_i \)-labeled subintervals of \( I_{t+1} \) to be in fact bijections. Therefore, given a counter machine \( A \), the set of possible encodings of the (unique, non-lossy) computation of \( A \) is specified in terms of a new formula \( \varphi_{\text{non-lossy}}^A \), which is obtained from \( \varphi^A \) (i.e., the formula introduced at the end of the proof of Theorem 8.2.4) by adding new conjuncts of the form \( \varphi^{-g_i} \) for all indices \( 1 \leq i \leq k \). Each of these conjuncts \( \varphi^{-g_i} \) precisely requires the partial surjective function \( g_i \) matching \( c_i \)-labeled subintervals of \( I_t \) with \( c_i \)-labeled subintervals of \( I_{t+1} \) to be, in addition, total and injective. In the sequel, we define the conjuncts \( \varphi^{-g_i} \) within the various logics \( \mathbb{A} \oplus \mathbb{B} \), \( \mathbb{A} \oplus \mathbb{B} \), \( \mathbb{A} \oplus \mathbb{B} \), \( \mathbb{A} \oplus \mathbb{B} \), and \( \mathbb{A} \oplus \mathbb{B} \), and we briefly discuss their semantics.

**Logic \( \mathbb{A} \oplus \mathbb{B} \).** For every \( 1 \leq i \leq k \), we define

\[
\varphi^{-g_i} = [\langle A \rangle c_i \land \neg \text{dec}] \rightarrow [\langle A \rangle g_i] \land [\langle B \rangle g_i \land [B] [\lnot g_i] \rightarrow [\langle D \rangle \neg g_i]].
\]
Intuitively, the first line of the formula $\varphi_1^{\omega}$ requires that every subinterval of $I_k$ which is labeled with $c_i$, but not with $\text{dec}$, is matched with a $c_i$-labeled subinterval of $I_{k+1}$, that is, the function $g_1$ is total. The second line of the formula tries to avoid the existence of pairs of $g_1$-labeled intervals that end in the same time point, that is, the function $g_1$ is injective; in fact, it enforces a stronger condition, namely, that there exist no intervals $I, J, K$ such that (i) both $I$ and $J$ are labeled by $g_1$, (ii) $I$ is the maximal interval that begins $K$, and (iii) $J$ is contained in, but does not begin or end, $K$. Even though the latter condition discards some valid encodings of the non-lossy computation of $A$ (precisely, those that feature $g_1$-labeled intervals contained one into each other), we can easily see that there exist equivalent encodings that guarantee that all $g_1$-labeled intervals are pairwise overlapping or non-intersecting. Under such an assumption, the second line of the formula $\varphi_1^{\omega}$ turns out to be equivalent to the condition that all $g_i$-labeled intervals end in pairwise distinct time points.

**Logic $A\bar{A}BB\bar{D}$**. The encoding of the equality constraints in the logic $A\bar{A}BB\bar{D}$ is analogous to that for the logic $A\bar{A}BB\bar{D}$. Precisely, for every $1 \leq i \leq k$, we define

$$
\varphi_i^{\omega} = \bigcup\left(\langle A \rangle c_i \land \neg \text{dec} \rightarrow \langle A \rangle g_1 \right) \land \\
\bigcup\left(\langle B \rangle g_1 \land \langle B \rangle [\neg g_1 \rightarrow [O] \neg g_i] \right).
$$

As a matter of fact, the above formula defines exactly the same models as those of the $A\bar{A}BB\bar{D}$ formula $\varphi_1^{\omega}$ introduced before.

**Logic $A\bar{A}B\bar{B}\bar{O}$**. For every $1 \leq i \leq k$, we define

$$
\varphi_i^{\omega} = \bigcup\left(\langle A \rangle c_i \land \neg \text{dec} \rightarrow \langle A \rangle g_1 \right) \land \\
\bigcup\left(\langle B \rangle g_i \land \langle B \rangle [\neg g_1 \rightarrow [O] \neg g_i] \right).
$$

The semantics of the above formula is similar to the previous one. The only difference now is that the second line avoids the existence of $g_1$-labeled intervals that are overlapping (rather than contained one into each other). By using arguments analogous to the previous cases, one can show that such an assumption is not too restrictive, since it still captures some valid encodings of the non-lossy computation of $A$.

**Logic $A\bar{A}B\bar{B}\bar{O}$**. The definitions and the arguments for the encoding in the logic $A\bar{A}B\bar{B}\bar{O}$ are symmetric to those for the logic $A\bar{A}B\bar{B}\bar{O}$:

$$
\varphi_i^{\omega} = \bigcup\left(\langle A \rangle c_i \land \neg \text{dec} \rightarrow \langle A \rangle g_1 \right) \land \\
\bigcup\left(\langle B \rangle g_i \land \langle B \rangle [\neg g_1 \rightarrow [O] \neg g_i] \right).
$$

\[\square\]

It is possible to show that the satisfiability problem for $A\bar{A}BB$ (in fact, this holds for its proper fragment $A\bar{A}B$) becomes undecidable if we interpret it over any class of linear orders that contains at least one linear order with an infinitely ascending sequence. It follows that, in particular, it is undecidable when $A\bar{A}BB$ is interpreted over natural time flows like $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$.

We first consider the satisfiability problem for $A\bar{A}B$ interpreted over $\mathbb{N}$. By definition, $\varphi$ is satisfiable over $\mathbb{N}$ if there exists an interval structure of the form $S = (I_w, \lambda, \bar{A}, \bar{B}, \sigma)$, with $I_w = \{[x, y] : 0 \leq x < y < \omega\}$ and $\sigma : I_w \rightarrow \mathcal{P}(\text{Prop})$, that satisfies it. A straightforward adaptation of the proof of Theorem 8.2.4 (see the proof of Theorem 8.3.2) shows that an undecidable variant of the universal reachability problem for lossy counter machines, called “structural termination” [43], is reducible to the satisfiability problem for $A\bar{A}B$ interpreted over interval structures of the form $S = (I_w, \lambda, \bar{A}, \bar{B}, \sigma)$. It immediately follows that the latter problem is undecidable as well. Such a negative result can be easily transferred to any class of linear orders that contains at least one linear order with an infinitely ascending sequence.

**Theorem 8.3.2.** The satisfiability problem for the logic $A\bar{A}B$, and hence that for the logic $A\bar{A}BB\bar{B}$, interpreted over any class of linear orders that contains at least one linear order with an infinitely ascending sequence is undecidable.
8.3. Undecidability is the rule, decidability the exception

Proof. We first reduce an undecidable variant of the universal reachability problem for lossy counter machines to the satisfiability problem for the logic $\mathbb{A} \mathbb{A} \mathbb{B}$ interpreted over structures of the form $S = (\mathbb{I}_w, A, \overline{A}, B, \sigma)$. The variant of the universal reachability problem we focus on is called structural termination and it consists of deciding, given a lossy counter machine $A = (Q, k, \delta)$ and a pair of control states $q_{source}$ and $q_{target}$, whether every computation of $A$ that starts at state $q_{source}$, with any arbitrary assignment for the counters, eventually reaches the state $q_{target}$, again with some arbitrary assignment for the counters. Given the results in [43], it is clear that the problem of structural termination is undecidable.

In order to reduce the above problem to a satisfiability problem for the logic $\mathbb{A} \mathbb{A} \mathbb{B}$, we use a technique similar to that of Theorem 8.2.4 and we encode an infinite computation $(q_1, z_1) (q_2, z_2) \ldots$ of $A$, with $q_1 = q_{source}$ and $q_t \neq q_{target}$ for all $t \geq 1$, into a suitable interval structure over the domain $([N, \prec])$. Precisely, we divide $([N, \prec])$ into an infinite sequence of intervals $I_0 = [0, x_1], I_1 = [x_1, x_2], \ldots$, where $1 = x_1 < x_2 \ldots$ and $x_{t+1} - x_t = 1 + \sum_{1 \leq i \leq k} z_i(t)$ for all $t \geq 1$. Then, we proceed exactly like in the proof of Theorem 8.2.4, by introducing suitable propositional variables and the auxiliary formulas $\varphi^A_I$, $\varphi^C_I$, etc. Finally, we represent the set of all infinite computations of $A$ that start in $q_{source}$ and avoid $q_{target}$ by means of the following formula:

$$
\varphi^A = \bigcup \left( (A) \top \land (\overline{A}) \top \land [B] \bot \to \bigvee_{a \in Q \cup C} a \land \bigwedge_{a \neq b \in Q \cup C} \neg (a \land b) \right) \land \\
\left( [A] \bot \lor [\overline{A}] \bot \lor (B) \top \to \bigwedge_{a \in Q \cup C \cup \{\text{inc, dec}\}} \neg a \right) \land \\
\left( [B] \land \langle q \rangle \to (\overline{A}) \langle A \rangle \varphi^A_q \right) \land \langle U \rangle q_{source} \land [U] \neg q_{target}.
$$

Clearly, $\varphi^A$ is satisfiable over a right-infinite interval structure of the form $S = (\mathbb{I}_w, A, \overline{A}, B, \sigma)$ iff there is an initial configuration of the form $(q_{source}, z)$, with $z \in \mathbb{N}^k$, for which the lossy counter machine $A$ never halts.

We conclude the proof by showing how the above undecidability result can be transferred to all right-infinite interval structures, that is, to all structures of the form $S = (\mathbb{I}, A, \overline{A}, B, \sigma)$, where $\mathbb{I}$ contains intervals over a fixed linear order $(\mathbb{I}, \prec)$ that embeds $(\mathbb{N}, \prec)$. To do that, we introduce a fresh propositional variable $\#$ and a suitable formula $\varphi_{\#}$ that enforces the following property: the set of all left-endpoints of $\#$-labeled intervals of $\mathbb{I}$ is an infinite and discrete subordering of $(\mathbb{I}, \prec)$ (hence, it embeds $(\mathbb{N}, \prec)$). Precisely, we let

$$
\varphi_{\#} = \bigcup \langle U \rangle \# \land \bigcup \left( \# \to (\overline{A})[A] \# \right) \land \left( \langle U \rangle (\# \to (\overline{A}) \langle A \rangle (\langle A \rangle \# \land [B][A] \neg \#)) \right)
$$

(intuitively, the above formula enforces that (i) there is at least one interval labeled by $\#$, (ii) for every pair of intervals $I, J$ that start at the same point, either both $I$ and $J$ are labeled by $\#$ or neither $I$ nor $J$ are labeled by $\#$, and (iii) the set of left-endpoints of $\#$-labeled intervals, equipped with the underlying order, is right-infinite and nowhere dense, namely, discrete). Finally, in order to correctly encode the set of relevant computations of $A$ inside the interval structure $S = ([I, A, \overline{A}, B, \sigma]$ it is sufficient to restrict the quantifications in the above formula $\varphi^A$ to range only over those intervals that satisfy $\# \land \langle A \rangle \#$. □
In the present dissertation we analyze the decidability of propositional interval temporal logics based on fragments of Halpern and Shoham’s propositional interval temporal logic (namely HS) over various classes of linear orders. By means of its operators HS can capture all the thirteen Allen’s interval relations. In previous years many undecidability results has been proved on very small fragments of HS under very weak assumptions on the linear orders on which these fragments are interpreted, it is the case of the begins,ends fragment ([39]). On the other hand the decidability results proposed use suitable reduction to well-studied point-based temporal logics ([34]) or are obtained by weakening the semantics of the interval temporal logics using homogeneity or locality properties ([41, 49]). The results proposed in the present work regards fragments of HS that are genuinely interval, in fact they cannot be reduced to point based temporal logics and all the proof are given without assumptions on the semantics.

Despite of the discouraging undecidability results that emerged when the study of HS has begun the present work consider very expressive fragments like the logic $\mathcal{ABB\bar{A}}$ and the logic $\mathcal{BB\bar{D}\bar{D}\bar{L}}$ that turns out to be decidable on finite linear orders and on dense linear orders respectively. It turns out from some recent undecidability results of HS ([10, 11]) that the extensions of these two logics with any other interval operators that cannot be captured by the logic itself leads to undecidability on every class of linear orders. Thanks to the various decidability results proposed in the recent years ([13, 45, 47, 56]) the decidability of HS fragments is more defined but it is far to be closed.

By our experience the hardest challenge it is represented by the decidability of the D fragment over finite linear orders. Very recent result ([15]) prove that the logic O is undecidable over discrete linear orders while over dense linear orders decidability is still open. The case of the D logic is completely symmetric to the case of the O logic, we know that it is decidable over dense orders but we have no conjecture on the decidability over the discrete case.

Moreover in the present dissertation are presented three applications of the decidability results obtained from the techniques applied in the study of HS fragments over linear orders. One application is represented by the extension to branching interval structures which, from the best of our knowledge, is a structure that has been poorly considered from the point of view of interval-based logics and, on the contrary, it is a structure widely studied in the case of point-based ones. Another area that should take benefit from the decidability results proposed for HS fragments is the spatial logics field. In the present work we see how to extend the results proposed for PNL in order to prove the decidability of a spatial version of it. However the most paradigmatic example of the strong connection between interval temporal logics and spatial logics is represented by the cone logic. This logic has been introduced in order to provide a decidable extension of the compass logic ([45]) and it turns out to be decidable by using an adaptation of a filtration technique well rehearsed in the field of interval and modal logics ([55]). Since the cone logic can capture the expressivity of the logic $\mathcal{BB\bar{D}\bar{D}\bar{L}}$ we obtain the decidability of this logic over $\mathbb{Q}$. However the decidability cone logic is open over all classes of dense grids, but, as in the case of the D logic, we consider the decidability of cone logic over discrete grids very challenging.

In figure 9.1 are depicted the current decidability results in interval temporal logics. For the sake of readability we omit the fragments generated from the inversion of all the relations (e.g.
from $A + B + \bar{B} + \bar{A}$ decidable over finite linear orders it turns out that $A + E + \bar{E} + \bar{A}$ is decidable over finite linear orders. All the decidability results are proved in a direct way or subsumed by some logic featured in the present work. The dark nodes are related to some undecidability results (obtained summing up the results proposed in [11, 15, 39, 42]) that blocks the introduction of new relations if we are interested in the research of new decidable fragments. For instance adding the $O$ (resp. $\bar{O}$ operator) to a logic that already contains the $D$ or the $B$ operator automatically leads to the undecidability on all classes of linear orders. The results obtained suggest that we have to focus our attention on completing the decidability of the fragments over all classes of linear orders or over different structure (like the branching interval structures introduced in Chapter 3) because the introduction of new modalities leads to indecidability almost in every case. In fact the only decidability results that featured some new operator may regard the case of the overlap relation $O$ interpreted over the classes of dense/general linear orders, since the case of discrete linear orders is proved to be undecidable [15].
decidable over all classes of linear orders.
decidable over $\mathbb{N}$ and its subsets.
decidable over finite linear orders.

Figure 9.1: The current classification of HS fragments
Bibliography


