The Horn Fragment of Branching Algebra

Alessandro Bertagnon
Department of Engineering, University of Ferrara, Italy
alessandro.bertagnon@unife.it

Marco Gavanelli
Department of Engineering, University of Ferrara, Italy
marco.gavanelli@unife.it

Alessandro Passantino
Department of Mathematics and Computer Science, University of Ferrara, Italy
alessandr.passantino@student.unife.it

Guido Sciavicco
Department of Mathematics and Computer Science, University of Ferrara, Italy
guido.sciavicco@unife.it

Stefano Trevisani
Department of Mathematics and Computer Science, University of Ferrara, Italy
stefan.trevisani@student.unife.it

Abstract

Branching Algebra is the natural branching-time generalization of Allen's Interval Algebra. As in the linear case, the consistency problem for Branching Algebra is NP-hard. Being relatively new, however, not much is known about the computational behaviour of the consistency problem of its sub-algebras, except in the case of the recently found subset of convex branching relations, for which the consistency of a network can be tested via path consistency and it is therefore deterministic polynomial. In this paper, following Nebel and Bürckert, we define the Horn fragment of Branching Algebra, and prove that it is a sub-algebra of the latter, being closed under inverse, intersection, and composition, that it strictly contains both the convex fragment of Branching Algebra and the Horn fragment of Interval Algebra, and that its consistency problem can be decided via path consistency. Finally, we experimentally prove that the Horn fragment of Branching Algebra can be used as an heuristic for checking the consistency of a generic network with a considerable improvement over the convex subset.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming

Keywords and phrases Constraint programming, Consistency, Branching time, Horn Fragment

Digital Object Identifier 10.4230/LIPIcs.TIME.2020.5

Funding *Guido Sciavicco*: G. Sciavicco acknowledges partial support by the Italian INDAM GNCS project *Strategic Reasoning and Automated Synthesis of Multi-Agent Systems*.

1 Introduction

In the context of temporal reasoning, Allen's Interval Algebra [1] (IA) is certainly one of the most important formalisms. Applications of the IA are widespread, and range from scheduling, to planning, database theory, and natural language processing, among others. In Allen's IA we consider the domain of all intervals on a linear order, and define thirteen basic relations (IA_{basic}) between pairs of intervals (such as, for example, *meets* or *before*); a constraint between two intervals is any disjunction of basic relations, and a network of constraints is defined as a set of variables plus a set of constraints between them, interpreted as a logical conjunction. Among the problems that emerge naturally in this field, checking the consistency of a network N of constraints is probably the most relevant one, and consists

of deciding whether N can be realized, that is, deciding if every variable can be instantiated to an interval without violating any constraint. The consistency problem is archetypical of the class of constraints satisfaction problems (CSP), because a network is a conjunction of constraints. The consistency problem for the IA is NP-complete, and classical approaches to efficient implementations are based either on clever brute-force enumerating algorithms (see, e.g. [8, 18]), or on tractable fragments of the algebra, which are interesting both on their own [9] and as heuristics to reduce the branching factor in branch-and-bound approaches for the full algebra [10, 13]. Two important fragments of the IA are the convex fragment (IA_{convex}), introduced by van Beek and Cohen [23], and encompassing 82 relations, and the more general ORD-Horn fragment (or, simply, the Horn fragment - IA_{Horn}), introduced by Nebel and Bürckert in [14], with 868 relations. In particular, to prove the tractability of the latter, Nebel and Bürckert identify a suitable point-based language that allows one to translate every relation of the Horn fragment of IA_{Horn} to a conjunction of Horn clauses; then, they prove that IA_{Horn} is closed under inverse, intersection, and composition, and that path consistency is complete for it.

In [15], the authors define a branching version of Allen's IA, which we refer to as Branching Algebra (BA), and introduce two possible sets of basic relations that may hold between two intervals on a tree-like partial order. One of these sets, composed of 24 mutually exclusive and jointly exhaustive basic relations, and also studied from the (first-order) expressive power point of view in [5], is characterized by basic relations whose semantics cannot be always written in the language of endpoints, therefore requiring quantification. By joining some of these relations via disjunction, one obtains a second set of 19, still mutually exclusive and jointly exhaustive, relations (BA_{basic}), each of which is translatable to the language of endpoints without using quantification. The consistency problem for a network of constraints in the algebra that emerges from these relations is, quite obviously, still NP-hard, and, in general, computationally more difficult than the one for IA. In [6], the authors presented the subset BA_{convex} of convex BA-relations, inspired by the convex fragment of the IA (IA_{convex}). The fragment BA_{convex}, that encompasses 91 relations, unlike its linear analogous, is not a subalgebra of BA, as it is not closed under composition; yet, it is closed on the (less restricting) operation of path consistency, which is also complete (w.r.t. deciding consistency) for it, making BA_{convex} the first non-trivial tractable fragment of BA. In this paper, we follow Nebel and Bürckert's approach, and define, first, a first order Horn theory (TORD-Horn), whose models can be interpreted as trees and in which BA-relations can be translated; then, we enumerate the subset of all and only BA-relations that can be translated in the language of TORD-Horn; finally, we prove, by enumeration, that such a subset (which we call BA_{Horn}) is closed under inverse, intersection, and composition, and it is therefore a subalgebra of BA. Finally, in the spirit of [6, 17], we implement a simple branch-and-bound algorithm for BA-networks to empirically study the expected improvement in computation time when the splitting is driven by BA_{Horn}-relations instead of basic relations.

2 Preliminaries

Notation. Let $(\mathcal{T}, <)$ be a partial order, whose elements are generally denoted by $a, b, \ldots$, and where $a||b$ (resp., a *lin* b) denotes that a and b are incomparable (resp., comparable) with respect to the ordering relation $<$. We use $x, y, \ldots$ to denote variables in the domain of points, and $x \leq y$ to denote $x < y \vee x = y$. A partial order $(\mathcal{T}, <)$, often denoted by $\mathcal{T}$, is a *future branching model of time* (or, simply, a *branching model*) if for all $a, b \in \mathcal{T}$ there is a greatest lower bound of a and b in $\mathcal{T}$, and, if $a||b$ then there exists no $c \in \mathcal{T}$ such that $c > a$

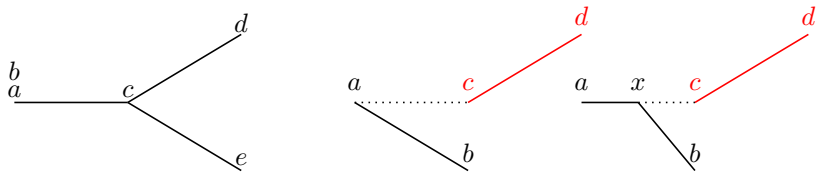

Figure 1 A pictorial representation of the four basic branching point relations, where $a = b$, $a < c$, $d > c$, and $d||e$ (left-hand side), and an example of two situations that require quantification to be distinguished in the language of endpoints (right-hand side).

Table 1 Composition of basic branching relations between points.

$\circ$	$<$	$>$	$=$	$\|\|$
$<$	$\{<\}$	*lin*	$\{<\}$	$\{\|\|, <\}$
$>$	?	$\{>\}$	$\{>\}$	$\{\|\|\}$
$=$	$\{<\}$	$\{>\}$	$\{=\}$	$\{\|\|\}$
$\|\|$	$\{\|\|\}$	$\{>, \|\|\}$	$\{\|\|\}$	?

and $c > b$ (that is, it is a tree). There are four basic relations that may hold between two points on a branching model: *equals* ($=$), *incomparable* ($\|\|$), *less than* ($<$), and *greater than* ($>$); the first two are symmetric, while the last two are inverse of each other. These relations are depicted in Figure 1 (left-hand side), and are called *basic branching point relations*. The set of basic branching point relations is denoted by BPA_{basic}. In the linear setting, the set of basic relations has only three elements, $<, =$, and $>$, and it is called PA_{basic} (*basic point relations*). An *interval* in $\mathcal{T}$ is a pair $[a, b]$ where $a < b$, and $[a, b] = \{x \in \mathcal{T} : a \leq x \leq b\}$. Intervals are generically denoted by $I, J, \ldots$. For an interval I (resp., X), we use I^-, I^+ to denote its endpoints. Following [5], one can describe 24 basic branching relations based on the possible relative position of two pairs of ordered points on a branching model, that is, by directly generalizing the universally known set of 13 *basic interval relations* [1] (IA_{basic}). While towards a precise study of the expressive power of branching relations in a first-order context this is an optimal choice, this is no longer true when studying the computational properties of the consistency problem. In particular, some of these relations require first-order quantification to be defined: for example, in Figure 1 (right-hand side) we see that, in order to distinguish the two situations, we need to quantify of the existence, or non-existence, of a point between a and c. To overcome this problem, that becomes relevant when we study the behaviour of branching relations in association with the behaviour of branching point relations (that is, by studying the properties of their point-based translations), Ragni and Wölfl [15] introduce a set of coarser relations, characterized by being translatable to point-based relations using only the language of endpoints, without quantification. These 19 relations are depicted in Figure 2, and form the set of *basic branching interval relations* (BA_{basic}); for each relation, the symbol in parentheses corresponds to its inverse, if the relation is not symmetric. A relation in the set BA_{basic} is either a linear relation, or the relation u (*unrelated*), or it corresponds to the disjunction between a pair of finer relations from the set of 24 [5]. For example, the relation *ib* is the disjunction of the two relations in Figure 1.

Operations and algebras. In general, given the basic relations $r_1, \ldots, r_l$, we denote by $R = \{r_1, \ldots, r_l\}$ the disjunctive relation $r_1 \vee \ldots \vee r_l$; thus, a relation is seen as a set, and a basic relation as a singleton. As the set IA_{basic} contains 13 elements, the set IA of all

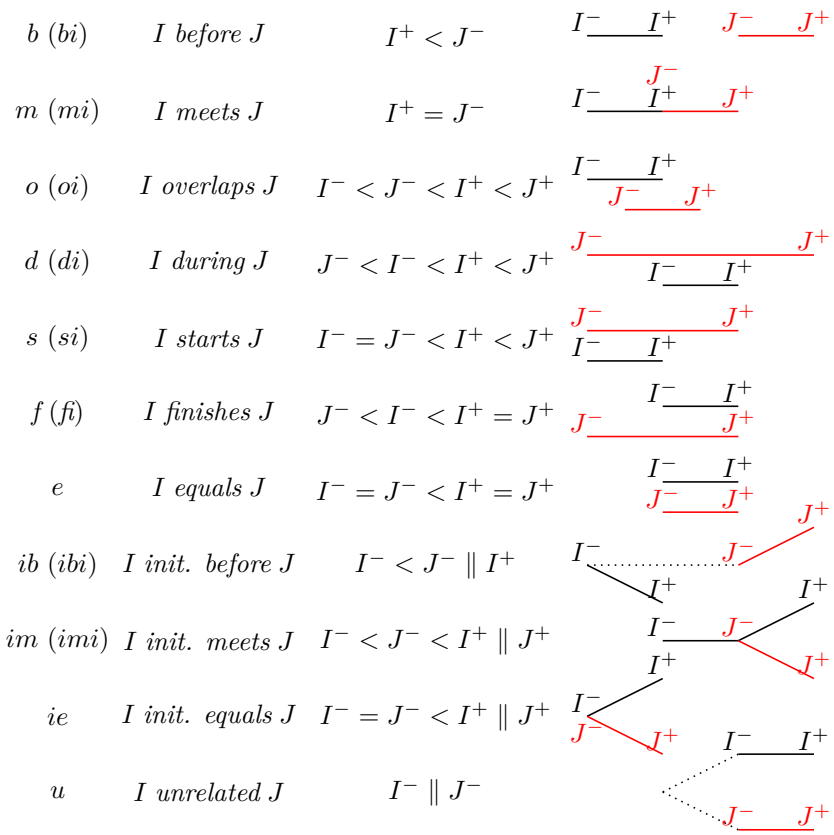

Figure 2 A pictorial representation of the nineteen basic branching interval relations. In this picture, in which $I = [I^-, I^+]$ and $J = [J^-, J^+]$, we assume $I^- < I^+$ and $J^- < J^+$. Solid lines are actual intervals, dashed lines complete the underlying tree structure. We use aR_1bR_2c as a shorthand for aR_1b and bR_2c.

interval relations in the linear setting encompasses $2^{13} - 1$ elements; similarly, the set BA_{basic} of 19 basic relations entails $2^{19} - 1$ interval relations in the branching setting. A *constraint* is an object of the type xRy, where x, y are point variables and R is a relation. There are three basic operations with relations: (Boolean) intersection, inverse, and composition. The *inverse* of a relation $R = \{r_1, \ldots, r_l\}$ is the relation $R^{-1} = \{r_1^{-1}, \ldots, r_l^{-1}\}$, where, for each basic relation r, r^{-1} is its inverse. In our notation, for example, *bi* (*later*) denotes the inverse of the basic relation *b* (*before*). The *composition* of two basic relations r_1, r_2 is defined as follows: for variables s, t, z, we say that s is in the *composed relation* $r_1 \circ r_2$ with t, denoted $s(r_1 \circ r_2)t$, if there exists z such that sr_1z and zr_2t. The composition of two relations R_1, R_2 is defined component-wise: $R_1 \circ R_2 = \{r \mid \exists r_1 \in R_1 \exists r_2 \in R_2 (r = r_1 \circ r_2)\}$. When a set of relations A is closed under inverse, intersection, and composition, we call it an *algebra*. Clearly, to compute the composition of two non-basic relations we base ourselves on the composition between basic relations, and to compute the latter in the interval ontology, both in the linear and the branching setting, we use the composition between basic relations in the point ontology. The latter can be easily computed "by hand" (see Table 1 for the branching case). The entire composition table between two intervals in the branching case is fully reported in [16] (and in [2] in the linear case). Given a set A of relations, an *A-network* is a directed graph $N = (V, E)$, where V is a set of variables and $E \subseteq V \times V$ is a set of

A-constraints between pairs of variables. To denote a constraint between the variables s and t in a network, we use indistinctly the notation (s, t) or the infix notation sRt (when we want to specify the relation). Given a network $N = (V, E)$, we say that N' is a sub-network of N if $N' = (V', E')$, $V' \subseteq V$, and E' is the projection of E on the variables in V'. Given a network, we say that it is *consistent* if there exists a model such that each variable can be mapped (*realized*) to a concrete element so that every constraint is respected; establishing if an A-network is consistent is the A-*consistency* problem.

Tractability and local consistency. Consistency problems such as those for IA, BA, and their fragments are often approached via popular heuristics such as constraint propagation and local consistency. A network N is said to be k-*consistent* if, given any consistent realization of $k-1$ variables, there exists an instantiation of any k-th variable such that the constraints between the subset of k variables can be satisfied together. Because of the particular nature of networks of constraints in temporal algebras, they are always 1-consistent (also called *node consistent*) and 2-consistent (also called *arc consistent*), by definition. Enforcing *path consistency*, that is, 3-consistency, in a network N, corresponds to apply the following simple algorithm: for every triple (s, t, z) of variables in $N = (V, E)$ such that $sRt, sR_1z, tR_2z \in E$, replace sRt by $s(R \cap (R_1 \circ R_2))t$. Clearly, if enforcing path consistency results in at least one empty constraint, the entire network N is not consistent. But, in general, enforcing path consistency (in fact, k-consistency for any constant $k < |V|$) does not imply consistency, and, indeed checking the consistency of a IA-network is a NP-hard problem [9]. Much effort has been devoted to identify, and classify, the relevant fragments of the IA for which the consistency problem becomes tractable. Besides the fragment of basic relations only, IA_{basic}, which is trivially tractable, two important tractable fragments are the *convex* fragment (IA_{convex}), introduced by van Beek and Cohen [23], and encompassing 82 relations, and the more general ORD-Horn fragment (or, simply, the Horn fragment - IA_{Horn}), introduced by Nebel and Bürckert in [14], encompassing 868 relations. Both IA_{convex} and IA_{Horn} are subalgebras of the IA, and in both cases checking path consistency is a complete method for checking the consistency.

In analogy with the linear case, Ragni and Wölfl [15] proved that also checking the consistency of a BA-network is at least NP-hard, and observed that the set of basic BA-relations only constitutes a tractable fragment (although not an algebra); also, in [6], the authors presented the branching version of the fragment IA_{convex}, called BA_{convex}, which is tractable, but, unlike its linear homologous, not closed under composition, and therefore not an algebra. Tractable fragments are not only important per se. As a matter of fact, the consistency problem for the full IA and BA alike is NP-complete, thanks to the fact that it can be decided by a simple branch-and-bound algorithm based on basic relations, and the completeness of path consistency for a fragment has another interesting consequence: improving the performances of such an algorithm. A branch-and-bound consistency checking algorithm is a backtracking algorithm that enforces path-consistency in each node of the search tree (more detail is in Section 5). At each step, the algorithm tries one basic relation for each relation. If at any step one relation results in the empty relation, it backtracks to the last choice; otherwise it proceeds to the next relation in the network. Fragments of the full algebra, both in the linear and the branching case, whose consistency can be decided via path consistency can be used to drive the splitting in such an algorithm, as a heuristics to speed up the branch-and-bound process: if, at any step, one ends up with a network whose labels are all contained in any such a fragment, that particular branch can be decided by simply enforcing path consistency. This has been done with both IA_{convex} and IA_{Horn} in the linear case, and with BA_{convex} in the branching case.

3 Applying Branching Algebra

Interval algebra has been known for 30 years, and its role in planning and database theory is universally accepted. Temporal reasoning in a branching setting is also a very well-established research area, at least at the logical level. Therefore, studying interval algebra in the branching setting is very natural. Possible application fields include the following ones.

Planning with errors. The use of IA, and in particular of IA-networks of constraints, to model planning problems is ubiquitous in the literature (see, e.g. [11, 12, 24]). A typical modelling exercise involves a set of *tasks* to be executed in order for a *goal* to be reached. Plans that are modelled with linear time, however, allow no margin for error: once the plan is being followed, every task must be executed. Using branching time we can develop plans that have alternative routes that can be taken in case some action fails. While we are following an (initial) part of the plan with actions that have no possibility of failing (in our abstract model), the underlying temporal model is linear; as soon as we encounter a task that may fail, the underlying model becomes branching, and, from that moment onward, different plans may be followed. In this sense, different branches will never join again; so, the underlying model is in fact tree-like, and a network of constraints that takes mistakes or obstacles into account is naturally modelled in BA.

Automatic generation of narrative. Generation of narrative is a modern application of artificial intelligence, and, more specifically, of natural language processing [22]. While the classical applications of automatically generated narratives include weather reports, instructions, descriptions of museum artifacts, narratives can be also used as the basis of automatic storytelling and plot generation [19]. Many modern and classic science-fiction stories, movies, and even video games make substantial use of parallel, incomparable timelines. To keep an adequate cause-effect consistency, however, in presence of non-trivial literary *escamotage* (such as time travel, for example), modelling the basic elements with BA may be a solution. The generated narrative can be checked for consistency to ensure that, while being possibly non-linear, it is internally coherent.

Verification of parallel programs. Some techniques for program verification make use of IA (see, e.g. [20]). Verifying parallel programs is a challenging task [4] which may take advantage from a branching interval algebra such as BA, in which the typical *fork* constructs can be modelled in a natural way. Consistency, in this case, can be interpreted as the absence of temporal contradictions in the executions of (sub)routines.

4 The Horn Fragment of BA

Horn branching relations. Every basic relation of IA, interpreted on a linear model $(\mathcal{T}, <)$ can be translated into a conjunction of formulas of the language of endpoints. Every non-basic relation, obviously, gives rise to a disjunction of such conjunctions, which, in turn, can be re-written into a conjunction of disjunctions, that is, of *clauses*. Thus, a network of constraints can be translated into a conjunction of clauses. Let us denote by $\Pi(r)$ (resp., $\Pi(R)$, $\Pi(N)$) the translation of a basic relation (resp., non-basic relation, network), and by $C, D, \ldots$ (resp., $\mathcal{C}, \mathcal{D}$) a generic clause (resp., set of clauses). As observed in [14], some translations of relations have the additional property that their corresponding set of clauses are *Horn*, that is, each clause has at most one positive literal; these are called IA_{Horn}-relations. By associating

such a translation to a first-order Horn theory, called ORD-HORN, whose models can be interpreted as linear orders, one obtains that:

(i) for a network N of IA_{Horn}-constraint, N is consistent if and only if $\Pi(N) \wedge$ ORD-HORN is satisfiable, and

(ii) checking the satisfiability of $\Pi(N) \wedge$ ORD-HORN is a tractable problem, for example via *positive unit resolution* [7].

In order to define the branching equivalent of IA_{Horn}, we need to construct the branching equivalent of ORD-HORN, which we denote by TORD-HORN. First, we need to define the language of TORD-HORN; then, we shall specify its axioms, and prove that every model of TORD-HORN can be interpreted as future branching models of time; finally, we can check which subset of relations of BA can be translated to the language of TORD-HORN, and that such subset forms an algebra.

▶ **Definition 1.** *The* language *of TORD-HORN encompasses an enumerable set of variables* $X, Y, \ldots$ *and the binary relations* $\doteq$ *(*equality*),* $\preceq$ *(*less or equal*),* $\sim$ *(*linear*),* $\shortparallel$ *(*incomparable*), and* $\prec\!\!\shortparallel$ *(*less or incomparable*).*

In this context, the theory of future branching models of time cannot be (fully) axiomatized in the standard way, because some of the necessary properties are not in form of Horn formulas (e.g., $X \sim Y$ defined as $X \preceq Y \vee Y \preceq X$). However, to our purposes it suffices to have models that can be *extended to* tree-like orderings.

▶ **Definition 2.** *The* theory *TORD-HORN is characterized by the following axioms:*

1. $X \doteq X$ *(reflexivity of* $\doteq$*);*
2. $X \doteq Y \rightarrow Y \doteq X$ *(symmetry of* $\doteq$*);*
3. $X \doteq Y \wedge Y \doteq Z \rightarrow X \doteq Z$ *(transitivity of* $\doteq$*);*
4. $X \preceq X$ *(reflexivity of* $\preceq$*);*
5. $X \preceq Y \wedge Y \preceq X \rightarrow X \doteq Y$ *(antisymmetry of* $\preceq$*);*
6. $X \preceq Y \wedge Y \preceq Z \rightarrow X \preceq Z$ *(transitivity of* $\preceq$*);*
7. $X \not\shortparallel X$ *(irreflexivity of* $\shortparallel$*);*
8. $X \shortparallel Y \rightarrow Y \shortparallel X$ *(symmetry of* $\shortparallel$*);*
9. $X \sim X$ *(reflexivity of* $\sim$*);*
10. $X \sim Y \rightarrow Y \sim X$ *(symmetry of* $\sim$*);*
11. $X \not\prec\!\!\shortparallel X$ *(irreflexivity of* $\prec\!\!\shortparallel$*);*
12. $X \prec\!\!\shortparallel Y \wedge Y \prec\!\!\shortparallel X \rightarrow X \shortparallel Y$ *(antisymmetry of* $\prec\!\!\shortparallel$*);*
13. $X \doteq Y \rightarrow X \preceq Y \wedge Y \preceq X \wedge X \sim Y$ *(weakening of* $\doteq$*);*
14. $X \preceq Y \rightarrow X \sim Y$ *(weakening of* $\preceq$*);*
15. $X \shortparallel Y \rightarrow X \prec\!\!\shortparallel Y \wedge Y \prec\!\!\shortparallel X$ *(weakening of* $\shortparallel$*);*
16. $X \shortparallel Y \rightarrow X \not\preceq Y \wedge Y \not\preceq X$ *(compatibility of* $\shortparallel$ *and* $\preceq$*);*
17. $X \sim Y \rightarrow X \not\shortparallel Y$ *(compatibility of* $\sim$ *and* $\shortparallel$*);*
18. $X \prec\!\!\shortparallel Y \rightarrow Y \not\preceq X$ *(compatibility of* $\prec\!\!\shortparallel$ *and* $\preceq$*);*
19. $X \shortparallel Y \wedge Y \preceq Z \rightarrow X \shortparallel Z$ *(tree-likeness).*

In the following, we denote by TORD-HORN the set of axioms 1-19[1]; observe that TORD-HORN is a Horn theory. We use the language of TORD-HORN to translate certain relations of BA; as we have recalled, such a translation is correct if and only if the resulting model can

[1] We do not claim these axiom are minimal; having a minimal set of axioms, however possible, would probably hid some of the underlying structure.

be interpreted as a future branching model of time. It turns out that, in order to guarantee that this is possible, we need to further limit the use of the language of TORD-Horn in translations, and, in particular, we say that C is an *admissible clause* if it uses only literals with the positive relations $\doteq, \preceq, \sim, \mid\mid, \prec\!\!\mid$, and the negative relation $\not\doteq$. Observe that limiting the use of certain relations does not decrease the (semantic) expressive power of the language, as $X \not\preceq Y$ (resp., $X \not\sim Y$, $X \not\mid Y$, $X \not\prec\!\!\mid Y$) can be written as $Y \prec\!\!\mid X$ (resp., $X \mid\mid Y$, $X \sim Y$, $Y \preceq X$).

▶ **Theorem 3.** *Every model* $(\mathcal{M}, \doteq, \preceq, \sim, \mid\mid, \prec\!\!\mid)$ *of TORD-Horn* $\cup\, \mathcal{C}$*, where* $\mathcal{C}$ *is a set of admissible clauses, can be represented as a branching model of time.*

It is important to remark that the use of an extended signature to specify the properties of a tree-like model is justified by the need of such a specification to be Horn. Admissible Horn clauses, as it can be proved by computer-assisted enumeration, are expressive enough to translate a subset of BA-relations that form an algebra, and allowing any of the forbidden symbols would require some non-Horn axiom.

▶ **Definition 4.** *The set* BA_{Horn} *is the subset of* BA *of all and only the relations that can be translated to the language of TORD-Horn using only admissible Horn clauses.*

▶ **Theorem 5.** BA_{Horn} *is an algebra, that is, it is closed under inverse, intersection, and composition.*

The set BA_{Horn} can be computed automatically, and it consists of 4510 relations. Although it covers less than 1% of the entire algebra, it is about 50 times more extended than BA_{convex}.

Completeness of path consistency. Let us consider a network N of BA_{Horn}-constraints. By the above results, we know that N is consistent if and only if $\Pi(N) \wedge$ TORD-Horn is satisfiable. Now, we ask ourselves if the consistency of N can also be checked by path consistency, in the same way in which the consistency of a network of IA_{Horn}-constraints can. Again following [14], proving that path consistency is complete for BA_{Horn} boils down to proving that, given a path consistent network N, the empty clause cannot be derived from $\Pi(N) \wedge$ TORD-Horn; to show the latter, one can restrict the attention to derivations that use positive unit resolution, which is complete for Horn clauses [7].

Let N be a path consistent BA_{Horn}-network. Let $\Pi(N) = \{\varphi_1, \varphi_2, \dots, \}$ be the Horn formulas of the signature TORD-Horn that are the result of translating the BA_{Horn}-constraints of $N = \{IR_1J, KR_2Z, \dots\}$; each φ_i is a conjunction of Horn clauses. The following observation will be relevant for us: by exhaustive exploration of all clauses that can be obtained from translating BA_{Horn}-relations, we realize that they either are unary or of the type:

$$(X \diamond Y \vee X \not\doteq Y) \text{ or } (Y \diamond X \vee X \not\doteq Y),$$

where $\diamond \in \{\preceq, \mid\mid, \prec\!\!\mid\}$. In the following we assume that each formula φ_i is *explicit*, that is, it explicitly contains all consequences of every axiom of TORD-Horn, and that each clause $C_j \in \varphi_i$ is *minimal*, that is, it contains no redundant literal; a set $\Pi(N)$ in which every formula is explicit, and every clause in every formula is minimal will be called *explicit* and *clause-minimal*. We want to prove that if N is path consistent and contains no empty relation, then positive unit resolution cannot deduce the empty clause from $\Pi(N) \wedge$ TORD-Horn.

▶ **Theorem 6.** *Let* N *be a path consistent* BA_{Horn}*-network. Then, if* $\Pi(N)$ *is explicit and clause-minimal, then the empty clause cannot be obtained from* $\Pi(N) \wedge$ *TORD-Horn by positive unit resolution.*

Algorithm 1 Backtracking algorithm.

```
1: function CONSISTENT(P, Split)
2:     enforce generalized arc consistency on P
3:     if there is a variable ν_XY such that 𝔇_XY = ∅ then
4:         return false
5:     else
6:         choose an unprocessed variable ν_XY such that 𝔇_XY ∉ Split
7:         if there is no such variable then
8:             return true
9:         {𝔇_1, ..., 𝔇_p}=PARTITION(𝔇_XY, Split)
10:        for all 𝔇_i ∈ {𝔇_1, ..., 𝔇_p} do
11:            P' = P_{𝔇_XY/𝔇_i}
12:            if CONSISTENT(P', Split) then
13:                return true
14:        return false
```

▶ **Corollary 7.** *Path consistency is complete for checking the consistency of a network of BA_{Horn}-relations.*

5 Experiments

In order to assess the usefulness of the fragment BA_{Horn} to improve the experimental computation time for checking the consistency of a network of BA-constraints, we devised a series of tests.

Constraint satisfaction problems. We designed a simple algorithm based on encoding the temporal network into a *constraint satisfaction problem (CSP)* using the classical *dual CSP* approach by Condotta et al. [3], based on the fact that enforcing path consistency on the original qualitative temporal network corresponds to enforcing *generalized arc consistency* on the corresponding dual CSP.

▶ **Definition 8.** *Given a BA-network $N = (V, E)$, its* dual CSP *is a triple $P = (\mathfrak{V}, \mathfrak{D}, \mathfrak{C})$, where $\mathfrak{V}$ is a set of* variables*, $\mathfrak{D}$ is a set of* variable domains*, and $\mathfrak{C}$ is a set* constraints*, such that:*

- **(i)** *$\mathfrak{V}$ contains a variable ν_{XY} for each pair of nodes $X, Y \in V$;*
- **(ii)** *$\mathfrak{D}$ contains a domain $\mathfrak{D}_{XY}$ for each variable in $\mathfrak{V}$, which corresponds to the constraint $XRY \in E$, and*
- **(iii)** *$\mathfrak{C}$ contains a binary constraint $inverse(\nu_{XY}, \nu_{YX})$ for each pair of nodes $X, Y \in V$, satisfied by all pairs (r, r^{-1}), where $r \in BA_{basic}$, and a ternary constraint $composition(\nu_{XY}, \nu_{YZ}, \nu_{XZ})$ for each triple of nodes $X, Y, Z \in V$, which encodes the composition table and is satisfied by all triples (r_1, r_2, r_3) such that $r_3 = r_2 \circ r_1$.*

Since path consistency is not complete for consistency checking of general networks, it is typically associated to a search algorithm, such as the one depicted in Algorithm 1 [6, 13]. Algorithm 1 checks the consistency of a general network; moreover, when there is a known fragment which is tractable through path consistency, Algorithm 1 can exploit it to speedup the search. The family of sets *Split* represents exactly such a tractable fragment. If no tractable fragment is known, the set *Split* contains just basic relations (as singleton sets),

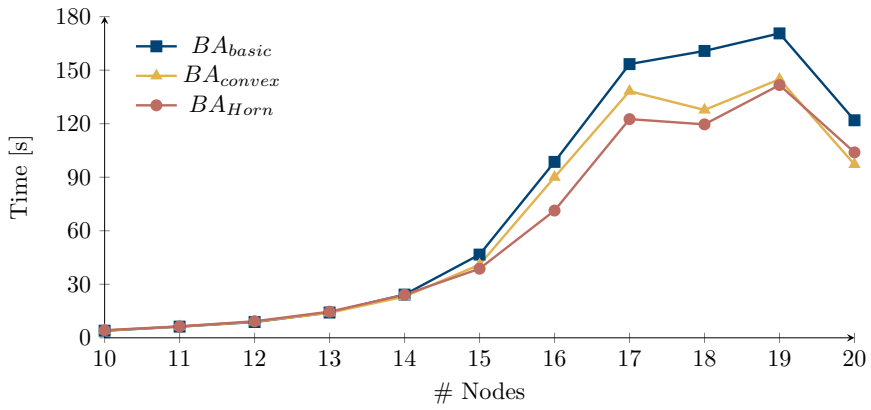

Figure 3 Running time of the backtracking algorithm varying the number of nodes n of the network. Each point represents the geometric mean of 100 instances, with density $d = 70\%$. Different lines represent different fragments as *Split* set.

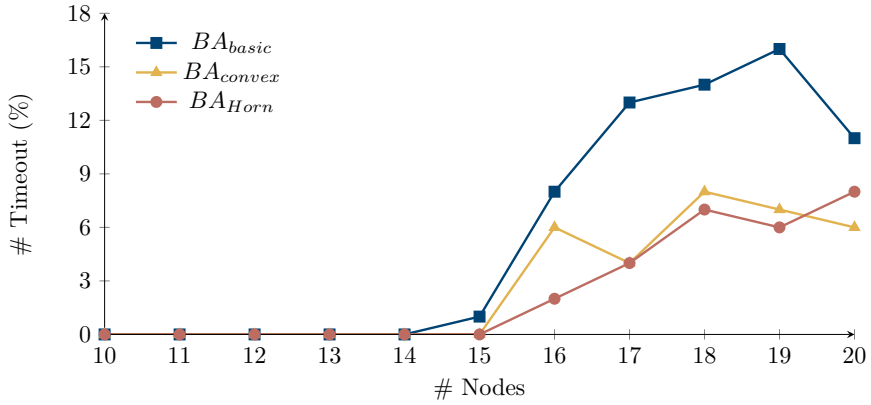

Figure 4 Fraction of instances that incurred in a timeout varying the number of nodes n of the network and fixing the density to $d = 70\%$. Different lines represent different fragments as *Split* set.

and the algorithm amounts to selecting a variable of the CSP, then splitting its domain into the basic relations (function Partition), nondeterministically assigning it one of the basic relations in its domain, enforcing path consistency on the obtained network, and recursively solving the remaining part of the CSP. On backtracking, another basic relation is selected, and so on; the search stops when all the variables of the CSP are assigned a basic relation in their domain. This is sound in the case of BA, since path consistency is complete for consistency for the set of basic BA-relations [15]. In case a larger fragment (e.g., BA_{convex} [6], or BA_{Horn}) is known to be solvable by path consistency, such fragment can be effectively used. Again, a variable of the CSP is selected, and its domain is partitioned into subsets, each belonging to the family *Split*. Since the subsets are no longer required to be singleton, the branching factor can be reduced; in general, the larger the fragment, the better the algorithm is expected to behave. Function Partition requires the solution of a set-partitioning problem, which is itself NP-hard, in the general case. In our case the trivial solution that splits a domain into its singletons is always feasible, and it can be computed in polynomial time; however such solution is useless, as Algorithm 1 would not exploit the tractable fragment. As in [6], we used a *trie* to store the tractable fragment, and a greedy

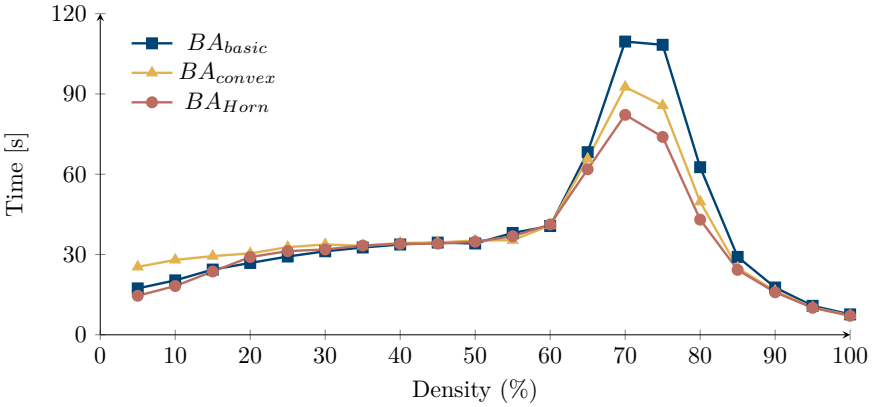

Figure 5 Running time of the backtracking algorithm varying the density d of the network. Each point represents the geometric mean of 50 instances, with number of nodes $n = 16$. Different lines represent different fragments as *Split* set.

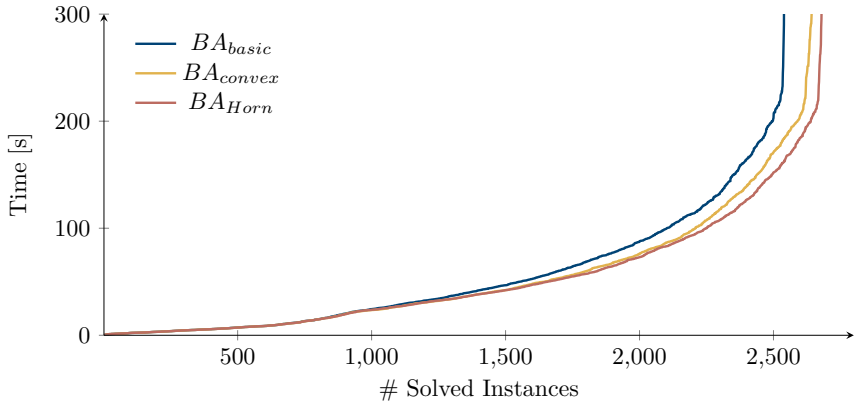

Figure 6 Cactus plot showing the number of solved instances varying the solving time. Instances have been generated with a number n of nodes varying from 15 to 20 and a constraint density d varying from 55% to 100%. Different lines represent different fragments as *Split* set in backtracking algorithm.

algorithm to quickly find a partition of the domain. Algorithm 1 was implemented in the Constraint Logic Programming environment ECL^{i}PS^e [21], that is a declarative language with built-in libraries for constraint satisfaction problems.

Experimental setting and results. In this experiment, random instances are generated as in [6], with a technique derived from [17]. Each instance is characterized by three parameters: the number of nodes n, the network density d, and the probability of a constraint p. Given the three parameters, for each given cardinality n, we generate a graph with n nodes, then we select $d\frac{n(n-1)}{2}$ edges at random. For each selected edge, we generate its domain by choosing with probability p each of the basic relations in BA_{basic}. Edges not selected are associated with the universal relation. Our experiments aim to assess the improvement of the backtracking algorithm when the BA_{Horn} fragment is used as *Split* heuristics as opposed to use the BA_{convex} fragment or using basic relations only.

In Figure 3 each point represents the geometric mean of 100 problem instances with density of the network fixed to 70%. The results suggest that using the BA_{Horn} fragment positively influences the computation time, not only with respect to not using it but also with respect to using the BA_{convex} fragment. Figure 4 where the fraction of instances that incurred in a timeout is plotted, confirm this observation. Figure 5 shows the running time of the backtracking algorithm varying the density of the network, while fixing the number of nodes to 16. Each point represents the geometric mean of 50 instances. The shape of the curves shows the *phase transition* as shown by BA_{convex} fragment in [6]: low density networks are easily satisfiable, while in high density networks the unsatisfiability is easily provable. Note that the new fragment improves in particular in the hardest region, at a density between 70% and 80%, in which both satisfiability and unsatisfiability are hard to prove. Finally, we generated 3000 random instances varying the number of nodes n from 15 to 20 and varying the constraint density d from 55% to 100%; also the cactus plot in Figure 6 shows that exploiting the BA_{Horn} fragment leads to an improvement in computation time. All experiments were run on ECL^{i}PS^e v. 7.0, build #54, with a time limit of 600s on Intel® Xeon® E5-2630 v3 CPUs running at 2.4GHz on CentOS Linux 7, using only one core and with 1GB of reserved memory.

6 Conclusions

Branching Algebra is the natural branching-time generalization of Allen's Interval Algebra. Being relatively new, not much is known about the computational behaviour of the consistency problem of its sub-algebras. Branching Algebra has been introduced in [15], where it has been proven that the consistency problem for the subset that includes only basic relations is tractable. Later, in [6], the subset of convex branching relations was introduced, showing that path consistency is complete for consistency in that case as well. In this paper, following Nebel and Bürckert [13], we further extended the convex fragment to obtain the Horn fragment of the Branching Algebra. We proved that it is a subalgebra, being closed under inverse, intersection, and composition, and that its consistency problem is treatable; we also proved that path consistency is complete for consistency in this case as well. Finally, we designed and conducted a series of experiments on randomly generated networks of constraints in the full algebra, to evaluate the improvement in computation time that comes from using the Horn fragment as heuristics.

This paper constitutes yet another step towards the complete classification between tractable/intractable fragments of Branching Algebra. At the moment, the Horn fragment is the biggest tractable known fragment, and our initial investigation points towards its maximality w.r.t. the tractability of the consistency problem. Yet, other, incomparable fragments may exist. The algebra of intervals is traditionally applied to task scheduling. In the branching case, applications are more difficult to visualize; yet, the Branching Algebra can be applied to a variety of situations in which multiple, incomparable timelines co-exist. In this paper, we have suggested a series of possible application scopes, but our list can be certainly extended and further explored.

References

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

2 J.F. Allen and P. J. Hayes. Short time periods. In *Proc. of IJCAI 1987: 10th International Joint Conference on Artificial Intelligence*, pages 981–983, 1987.

3 J.F. Condotta, D. D'Almeida, C. Lecoutre, and L. Saïs. From qualitative to discrete constraint networks. In *Proc. of KI 2006: Workshop on Qualitative Constraint Calculi*, pages 54–64, 2006.

4 S. Darabi, S.C.C. Blom, and M. Huisman. A verification technique for deterministic parallel programs. In *Proc. of NFM 2017: 9th International Symposium on NASA Formal Methods*, volume 10227 of *Lecture Notes in Computer Science*, pages 247–264. Springer, 2017.

5 S. Durhan and G. Sciavicco. Allen-like theory of time for tree-like structures. *Information and Computation*, 259(3):375–389, 2018.

6 M. Gavanelli, A. Passantino, and G. Sciavicco. Deciding the consistency of branching time interval networks. In *Proc. of TIME 2018: 25th International Symposium on Temporal Representation and Reasoning*, volume 120 of *LIPIcs*, pages 12:1–12:15, 2018.

7 L. Henshen and L. Wos. Unit refutation and Horn sets. *Journal of the ACM*, 21:590–605, 1974.

8 P. Jonsson and V. Lagerkvist. An initial study of time complexity in infinite-domain constraint satisfaction. *Artificial Intelligence*, 245:115–133, 2017.

9 A. Krokhin, P. Jeavons, and P. Jonsson. Reasoning about temporal relations: The tractable subalgebras of Allen's interval algebra. *Journal of the ACM*, 50(5):591–640, 2003.

10 P.B. Ladkin and A. Reinefeld. Fast algebraic methods for interval constraint problems. *Annals of Mathematics and Artificial Intelligence*, 19(3-4):383–411, 1997.

11 M. Mantle, S. Batsakis, and G. Antoniou. Large scale reasoning using Allen's Interval Algebra. In *Proc. of the 15th Mexican International Conference on Artificial Intelligence*, volume 11062 of *Lecture Notes in Computer Science*, pages 29–41. Springer, 2017.

12 L. Mudrová and N. Hawes. Task scheduling for mobile robots using interval algebra. In *Proc. of ICRA 2015: International Conference on Robotics and Automation*, pages 383–388. IEEE, 2015.

13 B. Nebel. Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of using the ORD-Horn class. *Constraints*, 1(3):175–190, 1997.

14 B. Nebel and H.J. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of allen's interval algebra. *Journal of the ACM*, 42(1):43–66, 1995.

15 M. Ragni and S. Wölfl. Branching Allen. In *Proc. of ISCS 2004: 4th International Conference on Spatial Cognition*, volume 3343 of *Lecture Notes in Computer Science*, pages 323–343. Springer, 2004.

16 A.J. Reich. Intervals, points, and branching time. In *Proc. of TIME 1994: 9th International Symposium on Temporal Representation and Reasoning*, pages 121–133. IEEE, 1994.

17 J. Renz and B. Nebel. Efficient methods for qualitative spatial reasoning. *Journal of Artificial Intelligence Resoning*, 15:289–318, 2001.

18 J. Renz and B. Nebel. Qualitative spatial reasoning using constraint calculi. In *Handbook of Spatial Logic*, pages 161–215. Springer, 2007.

19 E. Rishes, S.M. Lukin, D.K. Elson, and M.A. Walker. Generating different story tellings from semantic representations of narrative. In *Proc. of ICIDS 2013: 6th International Conference on Interactive Storytelling*, volume 8230 of *Lecture Notes in Computer Science*, pages 192–204. Springer, 2013.

20 G. Rosu and S. Bensalem. Allen linear (interval) temporal logic - translation to LTL and monitor synthesis. In *Proc. of CAV 2006: 18th International Conference on Computer Aided Verification*, volume 4144 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 2006.

21 J. Schimpf and K. Shen. Ecl$^{\text{i}}$ps$^{\text{e}}$ - from LP to CLP. *Theory and Practice of Logic Programming*, 12(1-2):127–156, 2012.

22 M. Theune, K. Meijs, D. Heylen, and R.Ordelman. Generating expressive speech for storytelling applications. *IEEE Transactions on Audio, Speech & Language Processing*, 14(4):1137–1144, 2006.

23 P. van Beek and R. Cohen. Exact and approximate reasoning about temporal relations. *Computational Intelligence*, 6:132–144, 1990.
24 A.K. Zaidi and L.W. Wagenhals. Planning temporal events using point-interval logic. *Mathematical and Computer Modelling*, 43(9):1229–1253, 2006.

A Appendix

Proof. (of Theorem 3) Since $\doteq$ is an equivalence relation, we can take the quotient $\mathcal{M}/_{\doteq}$, denoted $\mathcal{T}$, and equipped with the canonical equivalence $=$. In the following, we denote by $x, y, \ldots$, rather than $[X]_{/\doteq}, [Y]_{/\doteq}$, the elements of $\mathcal{T}$. We define the binary relation $\leq$ between classes:

$$x \leq y =^{def} \exists X, Y(X \in x \wedge Y \in y \wedge X \preceq Y),$$

and, consequently, $x < y$ as $x \leq y \wedge x \neq y$. We want to prove that $(\mathcal{T}, <)$ can be extended to a branching model of time.

- $\leq$ is an ordering relation. Clearly, $\leq$ is reflexive and antisymmetric because so is $\preceq$. Moreover, assume that $x \leq y$ and $y \leq z$ for some x, y, z. This means that $X \preceq Y$ and $Y' \preceq Z$ for some X, Y, Y', Z such that $X \in x$, $Y, Y' \in y$, and $Z \in z$. But since $Y, Y' \in y$, we have that $Y \doteq Y'$, and by axiom 13 we know that $Y \preceq Y'$. Since $\preceq$ is transitive, we obtain that $X \preceq Z$, implying that $x \leq z$. So, $\leq$ is also transitive. This also implies that $<$ is a strict pre-order, as it is irreflexive (because $\doteq$ is reflexive).
- $\leq$ can be extended to a tree-like order. To see this, observe that tree-likeness could be violated by having $x \not\leq y$, $y \not\leq x$, $y \leq z$, and either $x \leq z$ or $z \leq x$ for some x, y, z, but $\not\leq$ simply cannot be generated by the set $\mathcal{C}$, since it contains only admissible clauses. Because we need to interpret every symbol of the language of TORD-Horn, let us define the *incomparable* relation between classes, as:

  $$x \parallel y =^{def} \exists X, Y(X \in x \wedge Y \in y \wedge X \shortparallel Y),$$

  which is well-defined thanks to axiom 7 and axiom 8. To ensure that $(\mathcal{T}, <)$ can be extended to a tree-like ordering, we also have to guarantee that the introduction of $\parallel$ does not generate any contradictions. So, suppose that $x \parallel y, x \leq z$, and $y \leq t$ for some x, y, z, t. By definition, for some $X \in x$ and $Y \in y$ we have that $X \shortparallel Y$. Moreover, since $x \leq z$, for some $X' \in x$ and $Z \in z$ we have that $X' \preceq Z$. But this implies, by axiom 13, that $X \preceq Z$. So, axiom 19 applies, implying that $Y \shortparallel Z$. The same argument can be re-applied, leading us the conclusion that $Z \shortparallel T$. By definition, this implies that $z \parallel t$. By contradiction, assume now that $x \parallel y$ and $x \leq y$ for some x, y. This means that $X \shortparallel Y$ and $X' \preceq Y'$ for some $X, X' \in x$ and $Y, Y' \in y$. By axiom 13 and axiom 6, this implies that $X \preceq Y$, which is in contradiction with axiom 16. As a consequence of these two facts we have that $x \parallel y \leftrightarrow (x \not\leq y \wedge y \not\leq x)$ is realizable in $(\mathcal{T}, <)$. Now define:

  $$x \mathit{lin}\ y =^{def} \exists X, Y(X \in x \wedge Y \in y \wedge X \sim Y),$$

  Clearly *lin* is reflexive and symmetric because so is $\sim$, which implies that it is well-defined. Once again, we need to make sure that introducing *lin* does not generate contradictions. So, suppose, by contradiction, that $x \mathit{lin}\ y$ and $x \parallel y$ hold for some x, y. This means that $X \sim Y$ and $X' \shortparallel Y'$ for some $X, X' \in x$ and $Y, Y' \in y$. By axiom 13, this implies that $X \shortparallel Y$, which is in contradiction with axiom 17. Similarly, assume that $x \leq y$ for some x, y (the case in which $y \leq x$ or $x = y$ are similar). This means that $X \preceq Y$ for some

$X \in x$ and $Y \in y$. By axiom 14, this implies that $X \sim Y$, leading us to conclude that $x \; lin \; y$. Finally, since $\mathcal{C}$ is admissible, $x \; \not{lin} \; y \wedge x \not\parallel y$ cannot occur. As a consequence, we have that $x \; lin \; y \leftrightarrow (x \leq y \vee y \leq x \vee x = y)$ is realizable in $(\mathcal{T}, <)$. Finally, let us define:

$$x <\!\!\parallel y =^{def} \exists X, Y (X \in x \wedge Y \in y \wedge X \prec\!\!\!\!\!\!| \; Y),$$

which is well-defined thanks to axiom 11, 12, and 15. Suppose that, for some x, y it is the case that $x <\!\!\parallel y$ and $y \leq x$. This means that $X \prec\!\!\!\!\!\!| \; Y$ and $Y' \preceq X'$ for some $X, X' \in x$ and $Y, Y' \in y$. But since $X, X' \in x$ and $Y, Y' \in y$, we have that $X \doteq X'$ and $Y \doteq Y'$, and by axiom 13 and axiom 6, we know that $X \preceq Y$, which is in contradiction with axiom 18. Moreover, since $\mathcal{C}$ is admissible, $x \not<\!\!\parallel y \wedge y \not\leq x$ cannot occur. As a consequence, we have that $x <\!\!\parallel y \leftrightarrow x < y \vee x \parallel y$ is realizable in $(\mathcal{T}, <)$.

In conclusion, the structure $(\mathcal{T}, <)$ can be extended to a branching model of time, as we wanted. ◀

Proof. (of Theorem 6) We prove a stronger claim, that is, we prove that if N is a path consistent BA_{Horn}-network, and $\Pi(N)$ is explicit and clause-minimal, then no new positive unit clauses at all can be deduced by positive unit resolution from $\Pi(N) \wedge$ TORD-HORN. As a matter of fact, to deduce a new unit clause, it must be the case that $\Pi(N) \wedge$ TORD-HORN contains one clause $C = \neg L_1 \vee \neg L_2 \vee \ldots \vee \neg L_q \vee L$ (where $L_1, L_2, \ldots$ are propositional atoms), and a sequence of positive unit clauses $C_1 = L_1, C_2 = L_2, \ldots, C_q = L_q$, but does not contain the clause $C = L$. Moreover, it must also be the case that $q \leq 2$, as we have observed that clauses of $\Pi(N)$ are at most binary, and instances of axioms are at most ternary. We proceed by case analysis.

- Suppose, first, that C and $C_1, \ldots, C_q$ belong to $\Pi(N)$. If their variables are endpoints of different interval variables, then no resolution step can be applied. Suppose, then, that they contain the same endpoint variables; therefore, they also belong to the same formula φ_i. So, it must be the case that $C = X \diamond Y \vee X \not\doteq Y$, $C_1 = X \doteq Y$, and $q = 1$ (because, as we have observed, $\diamond$ must be positive). But, as it turns out, $\diamond \notin \{\doteq, \preceq, \sim\}$, otherwise $\Pi(N)$ could not be explicit, and $\diamond \notin \{\parallel, \prec\!\!\!\!\!\!| \}$, otherwise $\Pi(N)$ could not be clause-minimal. Therefore, $C, C_1, \ldots, C_q$ cannot all belong to $\Pi(N)$.
- Suppose, then, that C is an instance of some transitivity axiom (3 or 6). Then, no C_j can be an instance of some irreflexivity axiom (7 or 11), because it would not be positive, neither can be an instance of any other axiom except 1,4, and 9, because it would not be unitary; no resolution step can be carried on with 9, because $\sim$ is not transitive, and the only possible resolution steps that could be completed with the reflexivity of $\doteq$ and $\preceq$ would lead to tautologies. Therefore, every C_j must belong to $\Pi(N)$. If they are all clauses of the same formula φ, then either $C_1 = X \preceq Y$, $C_2 = Y \preceq Z$, and $q = 2$, in which case $C = X \preceq Z \in \varphi$ as well because φ is explicit, or $C_1 = X \doteq Y$, $C_2 = Y \doteq Z$, and $q = 2$, in which case $C = X \doteq Z \in \varphi$ for the same reason. Therefore, C_1 belongs to φ_1, which translates some constraint IR_1J, and C_2 belongs to φ_2, which translates some constraint JR_2K (if the constraints referred to completely different interval variables, then the endpoints variables would be different, and no resolution step could be performed). As before, either $C_1 = X \preceq Y$ and $C_2 = Y \preceq Z$, or $C_1 = X \doteq Y$ and $C_2 = Y \doteq Z$, and in both cases $q = 2$. Because N is path consistent, the constraint IR_3K exists, and $R_3 \subseteq R_1 \circ R_2$. Thus, $\Pi(N)$ also contains its translation φ_3. Since

(i) R_3 is stronger than $R_1 \circ R_2$,

(ii) composition is the systematic application of the transitivity axiom(s) and the tree-likeness axiom (see Table 1), and

(iii) L (which is either $X \preceq Z$ or $X \doteq Z$) can be deduced from C, C_1, and C_2,

it must be the case that $L \in \varphi_3$, so, also in this case, no new deduction can be performed.

- Assume, therefore, that C is an instance of the tree-likeness axiom. Then no C_j can be an instance of some reflexivity axiom (1 or 4), because L would not be new, nor can it be an instance of some irreflexivity axiom (7 or 11), because it would not be positive. Also, C_j can never be the instance of any other axiom because it would not be unitary. Therefore, every C_j must belong to $\Pi(N)$. If they are all clauses of the same formula φ, then $C_1 = X \shortparallel Y$, $C_2 = Y \preceq Z$, and $q = 2$, in which case we have that $X \shortparallel Z \in \varphi$ as well, because φ is explicit. Therefore, C_1 belongs to φ_1, which translates some constraint IR_1J, and C_2 belongs to φ_2, which translates some constraint JR_2K (if the constraints referred to completely different interval variables, then the endpoints variables would be different, and no resolution step could be performed). As above, $C_1 = X \shortparallel Y$ and $C_2 = Y \preceq Z$, and $q = 2$. Because N is path consistent, there exists a constraint IR_3K, and $R_3 \subseteq R_1 \circ R_2$. Thus, $\Pi(N)$ also contains its translation φ_3. Since

  (i) R_3 is stronger than $R_1 \circ R_2$,

  (ii) composition is the systematic application of the transitivity axiom(s) and the tree-likeness axiom, and

  (iii) L (which is $X \preceq Z$)) can be deduced from C, C_1, C_2,

  it must be the case that $L \in \varphi_3$, so, also in this case, no new deduction can be performed.
- Finally, suppose that C is any other axiom. C cannot be an instance of a reflexivity axiom (1 or 4), because it would be unitary and positive, and it cannot be an instance of any irreflexivity axiom (7 or 11) because C_1 would not be admissible. C cannot be the instance of any symmetry axiom (2, 8, or 10), because this would entail $q = 1$, which is to say C_1 would suffice for a deduction, but if C_1 belongs to some formula φ, then the latter must also contain L because it is explicit. Finally, if C is an instance of some antisymmetry axiom (5 or 12), then both C_1 and C_2 must refer to the same two endpoints as C, that is, they must belong to the same formula φ; therefore, since φ is explicit, L already belongs to φ, and if it is the instance of some weakening axiom (13, 14, or 15), or the instance of some compatibility axiom (16, 17, or 18), then C_1 must refer to the same two endpoints as C, and the same argument applies.

Therefore, no deduction can be performed on the translation of a path consistent network. ◀