

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.2020.8

Acknowledgements 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. Bertagnon and M. Gavanelli and A. Passantino and G. Sciavicco and S. Trevisani; licensed under Creative Commons License CC-BY

27th International Symposium on Temporal Representation and Reasoning (TIME 2020).

Editors: Emilio Muñoz-Velasco, Ana Ozaki, and Martin Theobald; Article No. 8; pp. 8:1–8:17

Leibniz International Proceedings in Informatics



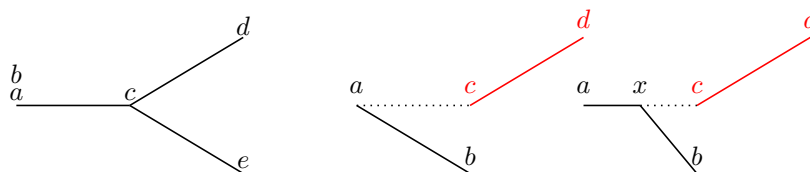
LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

46 a constraint between two intervals is any disjunction of basic relations, and a network of
 47 constraints is defined as a set of variables plus a set of constraints between them, interpreted
 48 as a logical conjunction. Among the problems that emerge naturally in this field, checking
 49 the consistency of a network N of constraints is probably the most relevant one, and consists
 50 of deciding whether N can be realized, that is, deciding if every variable can be instantiated
 51 to an interval without violating any constraint. The consistency problem is archetypical of
 52 the class of constraints satisfaction problems (CSP), because a network is a conjunction of
 53 constraints. The consistency problem for the IA is NP-complete, and classical approaches to
 54 efficient implementations are based either on clever brute-force enumerating algorithms (see,
 55 e.g. [8, 18]), or on tractable fragments of the algebra, which are interesting both on their
 56 own [9] and as heuristics to reduce the branching factor in branch-and-bound approaches
 57 for the full algebra [10, 13]. Two important fragments of the IA are the convex fragment
 58 (IA_{convex}), introduced by van Beek and Cohen [23], and encompassing 82 relations, and the
 59 more general ORD-HORN fragment (or, simply, the Horn fragment - IA_{Horn}), introduced
 60 by Nebel and Bürckert in [14], with 868 relations. In particular, to prove the tractability of
 61 the latter, Nebel and Bürckert identify a suitable point-based language that allows one to
 62 translate every relation of the Horn fragment of IA_{Horn} to a conjunction of Horn clauses;
 63 then, they prove that IA_{Horn} is closed under inverse, intersection, and composition, and
 64 that path consistency is complete for it.

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

89 2 Preliminaries

90 **Notation.** Let $(\mathcal{T}, <)$ be a partial order, whose elements are generally denoted by a, b, \dots ,
 91 and where $a||b$ (resp., $a \text{ lin } b$) denotes that a and b are incomparable (resp., comparable)

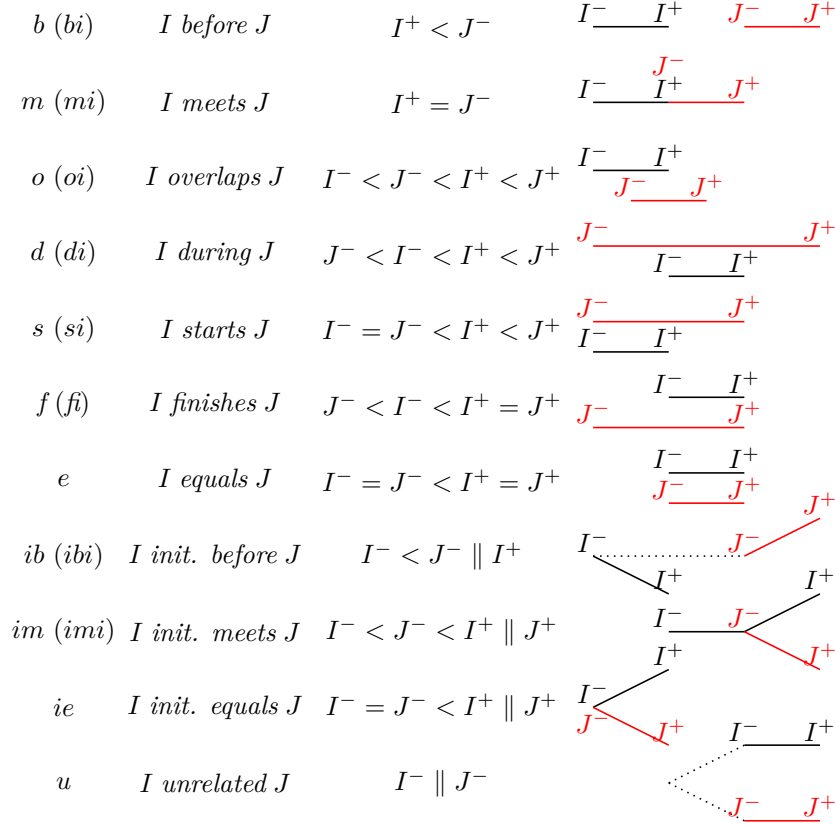


■ **Figure 1** A pictorial representation of the four basic branching point relations, where $a = b$, $a < c$, $d > c$, and $d \parallel e$ (left-hand side), and an example of two situations that require quantification to be distinguished in the language of endpoints (right-hand side).

92 with respect to the ordering relation $<$. We use x, y, \dots to denote variables in the domain of
 93 points, and $x \leq y$ to denote $x < y \vee x = y$. A partial order $(\mathcal{T}, <)$, often denoted by \mathcal{T} , is a
 94 *future branching model of time* (or, simply, a *branching model*) if for all $a, b \in \mathcal{T}$ there is a
 95 greatest lower bound of a and b in \mathcal{T} , and, if $a \parallel b$ then there exists no $c \in \mathcal{T}$ such that $c > a$
 96 and $c > b$ (that is, it is a tree). There are four basic relations that may hold between two
 97 points on a branching model: *equals* ($=$), *incomparable* (\parallel), *less than* ($<$), and *greater than*
 98 ($>$); the first two are symmetric, while the last two are inverse of each other. These relations
 99 are depicted in Figure 1 (left-hand side), and are called *basic branching point relations*. The
 100 set of basic branching point relations is denoted by BPA_{basic} . In the linear setting, the set
 101 of basic relations has only three elements, $<$, $=$, and $>$, and it is called PA_{basic} (*basic point*
 102 *relations*). An *interval* in \mathcal{T} is a pair $[a, b]$ where $a < b$, and $[a, b] = \{x \in \mathcal{T} : a \leq x \leq b\}$.
 103 Intervals are generically denoted by I, J, \dots . For an interval I (resp., X), we use I^-, I^+ to
 104 denote its endpoints. Following [5], one can describe 24 basic branching relations based on
 105 the possible relative position of two pairs of ordered points on a branching model, that is,
 106 by directly generalizing the universally known set of 13 *basic interval relations* [1] (IA_{basic}).
 107 While towards a precise study of the expressive power of branching relations in a first-order
 108 context this is an optimal choice, this is no longer true when studying the computational
 109 properties of the consistency problem. In particular, some of these relations require first-order
 110 quantification to be defined: for example, in Figure 1 (right-hand side) we see that, in order
 111 to distinguish the two situations, we need to quantify of the existence, or non-existence,
 112 of a point between a and c . To overcome this problem, that becomes relevant when we
 113 study the behaviour of branching relations in association with the behaviour of branching
 114 point relations (that is, by studying the properties of their point-based translations), Ragni
 115 and Wölf [15] introduce a set of coarser relations, characterized by being translatable to
 116 point-based relations using only the language of endpoints, without quantification. These
 117 19 relations are depicted in Figure 2, and form the set of *basic branching interval relations*
 118 (BA_{basic}); for each relation, the symbol in parentheses corresponds to its inverse, if the
 119 relation is not symmetric. A relation in the set BA_{basic} is either a linear relation, or the
 120 relation u (*unrelated*), or it corresponds to the disjunction between a pair of finer relations
 121 from the set of 24 [5]. For example, the relation ib is the disjunction of the two relations in
 122 Figure 1.

123 **Operations and algebras.** In general, given the basic relations r_1, \dots, r_l , we denote by
 124 $R = \{r_1, \dots, r_l\}$ the disjunctive relation $r_1 \vee \dots \vee r_l$; thus, a relation is seen as a set, and
 125 a basic relation as a singleton. As the set IA_{basic} contains 13 elements, the set IA of all
 126 interval relations in the linear setting encompasses $2^{13} - 1$ elements; similarly, the set BA_{basic}
 127 of 19 basic relations entails $2^{19} - 1$ interval relations in the branching setting. A *constraint*
 128 is an object of the type xRy , where x, y are point variables and R is a relation. There are

8:4 The Horn Fragment of Branching Algebra



■ **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 .

129 three basic operations with relations: (Boolean) intersection, inverse, and composition. The
 130 *inverse* of a relation $R = \{r_1, \dots, r_l\}$ is the relation $R^{-1} = \{r_1^{-1}, \dots, r_l^{-1}\}$, where, for each
 131 basic relation r , r^{-1} is its inverse. In our notation, for example, bi (*later*) denotes the inverse
 132 of the basic relation b (*before*). The *composition* of two basic relations r_1, r_2 is defined as
 133 follows: for variables s, t, z , we say that s is in the *composed relation* $r_1 \circ r_2$ with t , denoted
 134 $s(r_1 \circ r_2)t$, if there exists z such that sr_1z and zr_2t . The composition of two relations R_1, R_2
 135 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
 136 of relations A is closed under inverse, intersection, and composition, we call it an *algebra*.
 137 Clearly, to compute the composition of two non-basic relations we base ourselves on the
 138 composition between basic relations, and to compute the latter in the interval ontology, both
 139 in the linear and the branching setting, we use the composition between basic relations in the
 140 point ontology. The latter can be easily computed ‘by hand’ (see Table 1 for the branching
 141 case). The entire composition table between two intervals in the branching case is fully
 142 reported in [16] (and in [2] in the linear case). Given a set A of relations, an A -*network*
 143 is a directed graph $N = (V, E)$, where V is a set of variables and $E \subseteq V \times V$ is a set of
 144 A -constraints between pairs of variables. To denote a constraint between the variables s and
 145 t in a network, we use indistinctly the notation (s, t) or the infix notation sRt (when we
 146 want to specify the relation). Given a network $N = (V, E)$, we say that N' is a sub-network

\circ	$<$	$>$	$=$	\parallel
$<$	$\{<\}$	<i>lin</i>	$\{<\}$	$\{\parallel, <\}$
$>$	$\{?\}$	$\{>\}$	$\{>\}$	$\{\parallel\}$
$=$	$\{<\}$	$\{>\}$	$\{=\}$	$\{\parallel\}$
\parallel	$\{\parallel\}$	$\{>, \parallel\}$	$\{\parallel\}$	$\{?\}$

■ **Table 1** Composition of basic branching relations between points.

147 of N if $N' = (V', E')$, $V' \subseteq V$, and E' is the projection of E on the variables in V' . Given a
 148 network, we say that it is *consistent* if there exists a model such that each variable can be
 149 mapped (*realized*) to a concrete element so that every constraint is respected; establishing if
 150 an A -network is consistent is the A -consistency problem.

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

172 In analogy with the linear case, Ragni and Wöfl [15] proved that also checking the
 173 consistency of a BA -network is at least NP-hard, and observed that the set of basic BA -
 174 relations only constitutes a tractable fragment (although not an algebra); also, in [6], the
 175 authors presented the branching version of the fragment IA_{convex} , called BA_{convex} , which
 176 is tractable, but, unlike its linear homologous, not closed under composition, and therefore
 177 not an algebra. Tractable fragments are not only important per se. As a matter of fact, the
 178 consistency problem for the full IA and BA alike is NP-complete, thanks to the fact that
 179 it can be decided by a simple branch-and-bound algorithm based on basic relations, and
 180 the completeness of path consistency for a fragment has another interesting consequence:
 181 improving the performances of such an algorithm. A branch-and-bound consistency checking
 182 algorithm is a backtracking algorithm that enforces path-consistency in each node of the
 183 search tree (more detail is in Section 5). At each step, the algorithm tries one basic relation
 184 for each relation. If at any step one relation results in the empty relation, it backtracks to

185 the last choice; otherwise it proceeds to the next relation in the network. Fragments of the
 186 full algebra, both in the linear and the branching case, whose consistency can be decided via
 187 path consistency can be used to drive the splitting in such an algorithm, as a heuristics to
 188 speed up the branch-and-bound process: if, at any step, one ends up with a network whose
 189 labels are all contained in any such a fragment, that particular branch can be decided by
 190 simply enforcing path consistency. This has been done with both IA_{convex} and IA_{Horn} in
 191 the linear case, and with BA_{convex} in the branching case.

192 **3** Applying Branching Algebra

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

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

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

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

224 **4** The Horn Fragment of BA

225 **Horn branching relations.** Every basic relation of IA , interpreted on a linear model
 226 $(\mathcal{T}, <)$ can be translated into a conjunction of formulas of the language of endpoints. Every
 227 non-basic relation, obviously, gives rise to a disjunction of such conjunctions, which, in turn,

228 can be re-written into a conjunction of disjunctions, that is, of *clauses*. Thus, a network of
 229 constraints can be translated into a conjunction of clauses. Let us denote by $\Pi(r)$ (resp.,
 230 $\Pi(R)$, $\Pi(N)$) the translation of a basic relation (resp., non-basic relation, network), and
 231 by C, D, \dots (resp., \mathcal{C}, \mathcal{D}) a generic clause (resp., set of clauses). As observed in [14], some
 232 translations of relations have the additional property that their corresponding set of clauses
 233 are *Horn*, that is, each clause has at most one positive literal; these are called IA_{Horn} -
 234 relations. By associating such a translation to a first-order Horn theory, called ORD-HORN,
 235 whose models can be interpreted as linear orders, one obtains that: (i) for a network N
 236 of IA_{Horn} -constraint, N is consistent if and only if $\Pi(N) \wedge \text{ORD-HORN}$ is satisfiable, and
 237 (ii) checking the satisfiability of $\Pi(N) \wedge \text{ORD-HORN}$ is a tractable problem, for example via
 238 *positive unit resolution* [7].

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

245 ► **Definition 1.** *The language of TORD-HORN encompasses an enumerable set of variables*
 246 *X, Y, \dots and the binary relations \doteq (equality), \preceq (less or equal), \sim (linear), \parallel (incomparable),*
 247 *and \prec_{\parallel} (less or incomparable).*

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

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

- 253 1. $X \doteq X$ (*reflexivity of \doteq*);
- 254 2. $X \doteq Y \rightarrow Y \doteq X$ (*symmetry of \doteq*);
- 255 3. $X \doteq Y \wedge Y \doteq Z \rightarrow X \doteq Z$ (*transitivity of \doteq*);
- 256 4. $X \preceq X$ (*reflexivity of \preceq*);
- 257 5. $X \preceq Y \wedge Y \preceq X \rightarrow X \doteq Y$ (*antisymmetry of \preceq*);
- 258 6. $X \preceq Y \wedge Y \preceq Z \rightarrow X \preceq Z$ (*transitivity of \preceq*);
- 259 7. $X \not\parallel X$ (*irreflexivity of \parallel*);
- 260 8. $X \parallel Y \rightarrow Y \parallel X$ (*symmetry of \parallel*);
- 261 9. $X \sim X$ (*reflexivity of \sim*);
- 262 10. $X \sim Y \rightarrow Y \sim X$ (*symmetry of \sim*);
- 263 11. $X \not\prec_{\parallel} X$ (*irreflexivity of \prec_{\parallel}*);
- 264 12. $X \prec_{\parallel} Y \wedge Y \prec_{\parallel} X \rightarrow X \parallel Y$ (*antisymmetry of \prec_{\parallel}*);
- 265 13. $X \doteq Y \rightarrow X \preceq Y \wedge Y \preceq X \wedge X \sim Y$ (*weakening of \doteq*);
- 266 14. $X \preceq Y \rightarrow X \sim Y$ (*weakening of \preceq*);
- 267 15. $X \parallel Y \rightarrow X \prec_{\parallel} Y \wedge Y \prec_{\parallel} X$ (*weakening of \parallel*);
- 268 16. $X \parallel Y \rightarrow X \not\preceq Y \wedge Y \not\preceq X$ (*compatibility of \parallel and \preceq*);
- 269 17. $X \sim Y \rightarrow X \not\parallel Y$ (*compatibility of \sim and \parallel*);
- 270 18. $X \prec_{\parallel} Y \rightarrow Y \not\preceq X$ (*compatibility of \prec_{\parallel} and \preceq*);
- 271 19. $X \parallel Y \wedge Y \preceq Z \rightarrow X \parallel Z$ (*tree-likeness*).

272 In the following, we denote by TORD-HORN the set of axioms 1-19¹; observe that TORD-
 273 HORN is a Horn theory. We use the language of TORD-HORN to translate certain relations
 274 of BA ; as we have recalled, such a translation is correct if and only if the resulting model can
 275 be interpreted as a future branching model of time. It turns out that, in order to guarantee
 276 that this is possible, we need to further limit the use of the language of TORD-HORN in
 277 translations, and, in particular, we say that C is an *admissible clause* if it uses only literals
 278 with the positive relations $\dot{=}$, \preceq , \sim , \parallel , $\prec\parallel$, and the negative relation \neq . Observe that limiting
 279 the use of certain relations does not decrease the (semantic) expressive power of the language,
 280 as $X \not\preceq Y$ (resp., $X \not\sim Y$, $X \not\parallel Y$, $X \not\prec\parallel Y$) can be written as $Y \prec\parallel X$ (resp., $X \parallel Y$, $X \sim Y$,
 281 $Y \preceq X$).

282 ► **Theorem 3.** *Every model $(\mathcal{M}, \dot{=}, \preceq, \sim, \parallel, \prec\parallel)$ of $\text{TORD-HORN} \cup \mathcal{C}$, where \mathcal{C} is a set of*
 283 *admissible clauses, can be represented as a branching model of time.*

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

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

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

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

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

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

$$(X \diamond Y \vee X \neq Y) \text{ or } (Y \diamond X \vee X \neq Y),$$

309 where $\diamond \in \{\preceq, \parallel, \prec\parallel\}$. In the following we assume that each formula φ_i is *explicit*, that is, it
 310 explicitly contains all consequences of every axiom of TORD-HORN, and that each clause

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.

■ **Algorithm 1** Backtracking algorithm.

```

1: function CONSISTENT( $P, Split$ )
2:   enforce generalized arc consistency on  $P$ 
3:   if there is a variable  $\nu_{XY}$  such that  $\mathfrak{D}_{XY} = \emptyset$  then
4:     return false
5:   else
6:     choose an unprocessed variable  $\nu_{XY}$  such that  $\mathfrak{D}_{XY} \notin Split$ 
7:     if there is no such variable then
8:       return true
9:      $\{\mathfrak{D}_1, \dots, \mathfrak{D}_p\} = \text{PARTITION}(\mathfrak{D}_{XY}, Split)$ 
10:    for all  $\mathfrak{D}_i \in \{\mathfrak{D}_1, \dots, \mathfrak{D}_p\}$  do
11:       $P' = P_{\mathfrak{D}_{XY}/\mathfrak{D}_i}$ 
12:      if CONSISTENT( $P', Split$ ) then
13:        return true
14:    return false

```

311 $C_j \in \varphi_i$ is *minimal*, that is, it contains no redundant literal; a set $\Pi(N)$ in which every
312 formula is explicit, and every clause in every formula is minimal will be called *explicit* and
313 *clause-minimal*. We want to prove that if N is path consistent and contains no empty relation,
314 then positive unit resolution cannot deduce the empty clause from $\Pi(N) \wedge \text{TORD-HORN}$.

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

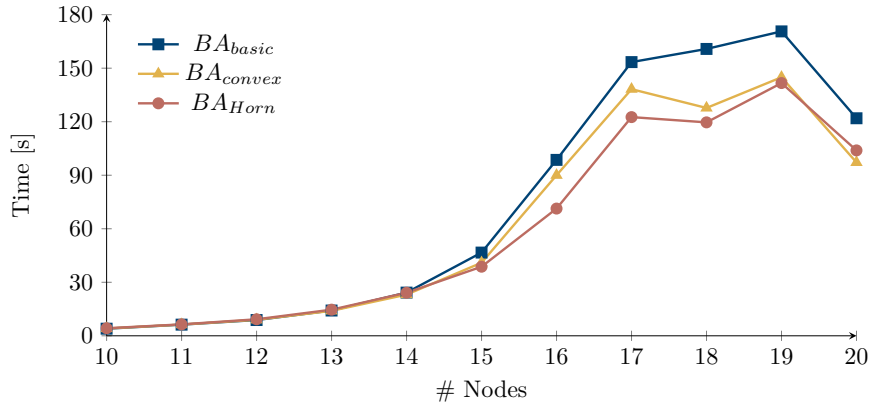
318 ► **Corollary 7.** *Path consistency is complete for checking the consistency of a network of*
319 *BA_{Horn} -relations.*

320 5 Experiments

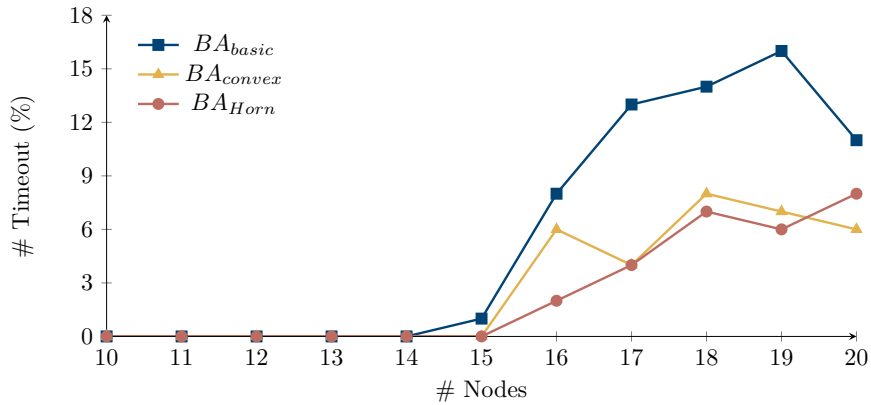
321 In order to assess the usefulness of the fragment BA_{Horn} to improve the experimental
322 computation time for checking the consistency of a network of BA -constraints, we devised a
323 series of tests.

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

329 ► **Definition 8.** *Given a BA -network $N = (V, E)$, its dual CSP is a triple $P = (\mathfrak{V}, \mathfrak{D}, \mathfrak{C})$,*
330 *where \mathfrak{V} is a set of variables, \mathfrak{D} is a set of variable domains, and \mathfrak{C} is a set constraints, such*
331 *that: (i) \mathfrak{V} contains a variable ν_{XY} for each pair of nodes $X, Y \in V$; (ii) \mathfrak{D} contains a domain*
332 *\mathfrak{D}_{XY} for each variable in \mathfrak{V} , which corresponds to the constraint $XRY \in E$, and (iii) \mathfrak{C}*
333 *contains a binary constraint $\text{inverse}(\nu_{XY}, \nu_{YX})$ for each pair of nodes $X, Y \in V$, satisfied by*
334 *all pairs (r, r^{-1}) , where $r \in BA_{basic}$, and a ternary constraint $\text{composition}(\nu_{XY}, \nu_{YZ}, \nu_{XZ})$*
335 *for each triple of nodes $X, Y, Z \in V$, which encodes the composition table and is satisfied by*
336 *all triples (r_1, r_2, r_3) such that $r_3 = r_2 \circ r_1$.*

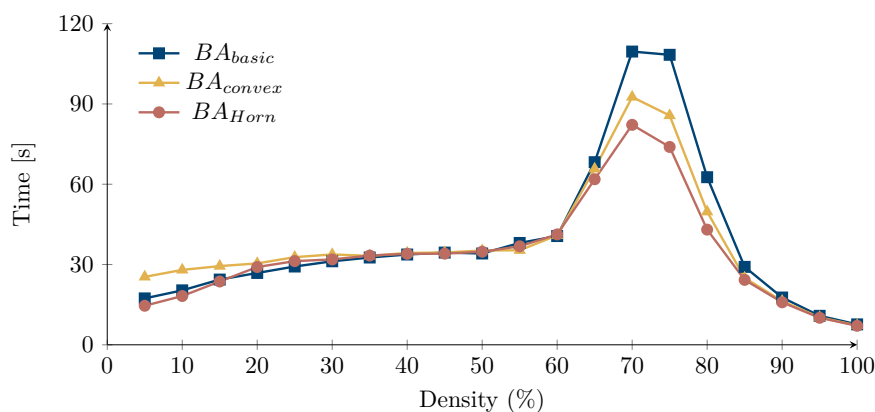


■ **Figure 3** Running time of the backtracking algorithm varying the number of nodes n of the network. Each point represents the geometric mean of 100 instances, with density $d = 70\%$. Different lines represent different fragments as *Split* set.

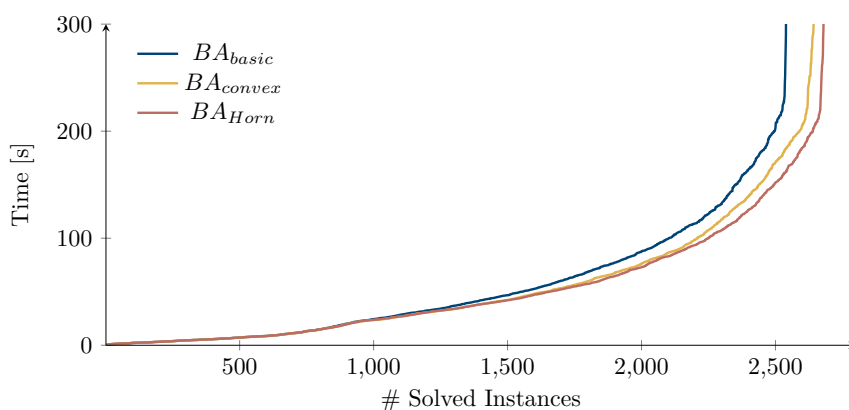


■ **Figure 4** Fraction of instances that incurred a timeout varying the number of nodes n of the network and fixing the density to $d = 70\%$. Different lines represent different fragments as *Split* set.

337 Since path consistency is not complete for consistency checking of general networks, it is
 338 typically associated to a search algorithm, such as the one depicted in Algorithm 1 [6, 13].
 339 Algorithm 1 checks the consistency of a general network; moreover, when there is a known
 340 fragment which is tractable through path consistency, Algorithm 1 can exploit it to speedup
 341 the search. The family of sets *Split* represents exactly such a tractable fragment. If no
 342 tractable fragment is known, the set *Split* contains just basic relations (as singleton sets),
 343 and the algorithm amounts to selecting a variable of the CSP, then splitting its domain
 344 into the basic relations (function PARTITION), nondeterministically assigning it one of the
 345 basic relations in its domain, enforcing path consistency on the obtained network, and
 346 recursively solving the remaining part of the CSP. On backtracking, another basic relation
 347 is selected, and so on; the search stops when all the variables of the CSP are assigned a
 348 basic relation in their domain. This is sound in the case of BA , since path consistency is
 349 complete for consistency for the set of basic BA -relations [15]. In case a larger fragment (e.g.,
 350 BA_{convex} [6], or BA_{Horn}) is known to be solvable by path consistency, such fragment can
 351 be effectively used. Again, a variable of the CSP is selected, and its domain is partitioned
 352 into subsets, each belonging to the family *Split*. Since the subsets are no longer required to



■ **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.

353 be singleton, the branching factor can be reduced; in general, the larger the fragment, the
 354 better the algorithm is expected to behave. Function PARTITION requires the solution of a
 355 set-partitioning problem, which is itself NP-hard, in the general case. In our case the trivial
 356 solution that splits a domain into its singletons is always feasible, and it can be computed
 357 in polynomial time; however such solution is useless, as Algorithm 1 would not exploit the
 358 tractable fragment. As in [6], we used a *trie* to store the tractable fragment, and a greedy
 359 algorithm to quickly find a partition of the domain. Algorithm 1 was implemented in the
 360 Constraint Logic Programming environment ECLⁱPS^e [21], that is a declarative language
 361 with built-in libraries for constraint satisfaction problems.

362 **Experimental setting and results.** In this experiment, random instances are generated as
 363 in [6], with a technique derived from [17]. Each instance is characterized by three parameters:
 364 the number of nodes n , the network density d , and the probability of a constraint p . Given
 365 the three parameters, for each given cardinality n , we generate a graph with n nodes, then
 366 we select $d \frac{n(n-1)}{2}$ edges at random. For each selected edge, we generate its domain by

367 choosing with probability p each of the basic relations in BA_{basic} . Edges not selected are
 368 associated with the universal relation. Our experiments aim to assess the improvement of the
 369 backtracking algorithm when the BA_{Horn} fragment is used as *Split* heuristics as opposed to
 370 use the BA_{convex} fragment or using basic relations only.

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

388 6 Conclusions

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

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

411 — References —

- 412 1 J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*,
413 26(11):832–843, 1983.
- 414 2 J.F. Allen and P. J. Hayes. Short time periods. In *Proc. of IJCAI 1987: 10th International*
415 *Joint Conference on Artificial Intelligence*, pages 981–983, 1987.
- 416 3 J.F. Condotta, D. D’Almeida, C. Lecoutre, and L. Saïs. From qualitative to discrete constraint
417 networks. In *Proc. of KI 2006: Workshop on Qualitative Constraint Calculi*, pages 54–64,
418 2006.
- 419 4 S. Darabi, S.C.C. Blom, and M. Huisman. A verification technique for deterministic parallel
420 programs. In *Proc. of NFM 2017: 9th International Symposium on NASA Formal Methods*,
421 volume 10227 of *Lecture Notes in Computer Science*, pages 247–264. Springer, 2017.
- 422 5 S. Durhan and G. Sciavicco. Allen-like theory of time for tree-like structures. *Information*
423 *and Computation*, 259(3):375–389, 2018.
- 424 6 M. Gavanelli, A. Passantino, and G. Sciavicco. Deciding the consistency of branching time
425 interval networks. In *Proc. of TIME 2018: 25th International Symposium on Temporal*
426 *Representation and Reasoning*, volume 120 of *LIPICs*, pages 12:1–12:15, 2018.
- 427 7 L. Henshen and L. Wos. Unit refutation and Horn sets. *Journal of the ACM*, 21:590–605,
428 1974.
- 429 8 P. Jonsson and V. Lagerkvist. An initial study of time complexity in infinite-domain constraint
430 satisfaction. *Artificial Intelligence*, 245:115–133, 2017.
- 431 9 A. Krokhin, P. Jeavons, and P. Jonsson. Reasoning about temporal relations: The tractable
432 subalgebras of Allen’s interval algebra. *Journal of the ACM*, 50(5):591–640, 2003.
- 433 10 P.B. Ladkin and A. Reinefeld. Fast algebraic methods for interval constraint problems. *Annals*
434 *of Mathematics and Artificial Intelligence*, 19(3-4):383–411, 1997.
- 435 11 M. Mantle, S. Batsakis, and G. Antoniou. Large scale reasoning using Allen’s Interval Algebra.
436 In *Proc. of the 15th Mexican International Conference on Artificial Intelligence*, volume 11062
437 of *Lecture Notes in Computer Science*, pages 29–41. Springer, 2017.
- 438 12 L. Mudrová and N. Hawes. Task scheduling for mobile robots using interval algebra. In *Proc.*
439 *of ICRA 2015: International Conference on Robotics and Automation*, pages 383–388. IEEE,
440 2015.
- 441 13 B. Nebel. Solving hard qualitative temporal reasoning problems: Evaluating the efficiency of
442 using the ORD-Horn class. *Constraints*, 1(3):175–190, 1997.
- 443 14 B. Nebel and H.J. Bürckert. Reasoning about temporal relations: A maximal tractable subclass
444 of allen’s interval algebra. *Journal of the ACM*, 42(1):43–66, 1995.
- 445 15 M. Ragni and S. Wöfl. Branching Allen. In *Proc. of ISCS 2004: 4th International Conference*
446 *on Spatial Cognition*, volume 3343 of *Lecture Notes in Computer Science*, pages 323–343.
447 Springer, 2004.
- 448 16 A.J. Reich. Intervals, points, and branching time. In *Proc. of TIME 1994: 9th International*
449 *Symposium on Temporal Representation and Reasoning*, pages 121–133. IEEE, 1994.
- 450 17 J. Renz and B. Nebel. Efficient methods for qualitative spatial reasoning. *Journal of Artificial*
451 *Intelligence Reasoning*, 15:289–318, 2001.
- 452 18 J. Renz and B. Nebel. Qualitative spatial reasoning using constraint calculi. In *Handbook of*
453 *Spatial Logic*, pages 161–215. Springer, 2007.
- 454 19 E. Rishes, S.M. Lukin, D.K. Elson, and M.A. Walker. Generating different story tellings from
455 semantic representations of narrative. In *Proc. of ICIDS 2013: 6th International Conference*
456 *on Interactive Storytelling*, volume 8230 of *Lecture Notes in Computer Science*, pages 192–204.
457 Springer, 2013.
- 458 20 G. Rosu and S. Bensalem. Allen linear (interval) temporal logic - translation to LTL and
459 monitor synthesis. In *Proc. of CAV 2006: 18th International Conference on Computer Aided*
460 *Verification*, volume 4144 of *Lecture Notes in Computer Science*, pages 263–277. Springer,
461 2006.

8:14 The Horn Fragment of Branching Algebra

- 462 21 J. Schimpf and K. Shen. Eclⁱps^e - from LP to CLP. *Theory and Practice of Logic Programming*,
463 12(1-2):127–156, 2012.
- 464 22 M. Theune, K. Meijs, D. Heylen, and R.Ordelman. Generating expressive speech for storytelling
465 applications. *IEEE Transactions on Audio, Speech & Language Processing*, 14(4):1137–1144,
466 2006.
- 467 23 P. van Beek and R. Cohen. Exact and approximate reasoning about temporal relations.
468 *Computational Intelligence*, 6:132–144, 1990.
- 469 24 A.K. Zaidi and L.W. Wagenhals. Planning temporal events using point-interval logic. *Math-*
470 *ematical and Computer Modelling*, 43(9):1229 – 1253, 2006.

471 **Appendix**

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

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

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

- 478 ■ \leq is an ordering relation. Clearly, \leq is reflexive and antisymmetric because so is \preceq .
 479 Moreover, assume that $x \leq y$ and $y \leq z$ for some x, y, z . This means that $X \preceq Y$ and
 480 $Y' \preceq Z$ for some X, Y, Y', Z such that $X \in x, Y, Y' \in y$, and $Z \in z$. But since $Y, Y' \in y$,
 481 we have that $Y \doteq Y'$, and by axiom 13 we know that $Y \preceq Y'$. Since \preceq is transitive, we
 482 obtain that $X \preceq Z$, implying that $x \leq z$. So, \leq is also transitive. This also implies that
 483 $<$ is a strict pre-order, as it is irreflexive (because \doteq is reflexive).
- 484 ■ \leq can be extended to a tree-like order. To see this, observe that tree-likeness could be
 485 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
 486 $\not\leq$ simply cannot be generated by the set \mathcal{C} , since it contains only admissible clauses.
 487 Because we need to interpret every symbol of the language of TORD-HORN, let us define
 488 the *incomparable* relation between classes, as:

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

489 which is well-defined thanks to axiom 7 and axiom 8. To ensure that $(\mathcal{T}, <)$ can be
 490 extended to a tree-like ordering, we also have to guarantee that the introduction of \parallel
 491 does not generate any contradictions. So, suppose that $x \parallel y, x \leq z$, and $y \leq t$ for some
 492 x, y, z, t . By definition, for some $X \in x$ and $Y \in y$ we have that $X \parallel Y$. Moreover, since
 493 $x \leq z$, for some $X' \in x$ and $Z \in z$ we have that $X' \preceq Z$. But this implies, by axiom 13,
 494 that $X \preceq Z$. So, axiom 19 applies, implying that $Y \parallel Z$. The same argument can be
 495 re-applied, leading us the conclusion that $Z \parallel T$. By definition, this implies that $z \parallel t$. By
 496 contradiction, assume now that $x \parallel y$ and $x \leq y$ for some x, y . This means that $X \parallel Y$
 497 and $X' \preceq Y'$ for some $X, X' \in x$ and $Y, Y' \in y$. By axiom 13 and axiom 6, this implies
 498 that $X \preceq Y$, which is in contradiction with axiom 16. As a consequence of these two
 499 facts we have that $x \parallel y \leftrightarrow (x \not\leq y \wedge y \not\leq x)$ is realizable in $(\mathcal{T}, <)$. Now define:

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

500 Clearly *lin* is reflexive and symmetric because so is \sim , which implies that it is well-defined.
 501 Once again, we need to make sure that introducing *lin* does not generate contradictions.
 502 So, suppose, by contradiction, that $x \text{ lin } y$ and $x \parallel y$ hold for some x, y . This means that
 503 $X \sim Y$ and $X' \parallel Y'$ for some $X, X' \in x$ and $Y, Y' \in y$. By axiom 13, this implies that
 504 $X \parallel Y$, which is in contradiction with axiom 17. Similarly, assume that $x \leq y$ for some
 505 x, y (the case in which $y \leq x$ or $x = y$ are similar). This means that $X \preceq Y$ for some
 506 $X \in x$ and $Y \in y$. By axiom 14, this implies that $X \sim Y$, leading us to conclude that
 507 $x \text{ lin } y$. Finally, since \mathcal{C} is admissible, $x \text{ lin } y \wedge x \parallel y$ cannot occur. As a consequence, we
 508 have that $x \text{ lin } y \leftrightarrow (x \leq y \vee y \leq x \vee x = y)$ is realizable in $(\mathcal{T}, <)$. Finally, let us define:

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

8:16 The Horn Fragment of Branching Algebra

509 which is well-defined thanks to axiom 11, 12, and 15. Suppose that, for some x, y it is the
 510 case that $x <|| y$ and $y \leq x$. This means that $X <_{||} Y$ and $Y' \preceq X'$ for some $X, X' \in x$
 511 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
 512 by axiom 13 and axiom 6, we know that $X \preceq Y$, which is in contradiction with axiom 18.
 513 Moreover, since \mathcal{C} is admissible, $x <|| y \wedge y \not\leq x$ cannot occur. As a consequence, we have
 514 that $x <|| y \leftrightarrow x < y \vee x || y$ is realizable in $(\mathcal{T}, <)$.

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

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

- 526 ■ Suppose, first, that C and C_1, \dots, C_q belong to $\Pi(N)$. If their variables are endpoints of
 527 different interval variables, then no resolution step can be applied. Suppose, then, that
 528 they contain the same endpoint variables; therefore, they also belong to the same formula
 529 φ_i . So, it must be the case that $C = X \diamond Y \vee X \neq Y$, $C_1 = X \doteq Y$, and $q = 1$ (because,
 530 as we have observed, \diamond must be positive). But, as it turns out, $\diamond \notin \{\dot{=}, \preceq, \sim\}$, otherwise
 531 $\Pi(N)$ could not be explicit, and $\diamond \notin \{||, <_{||}\}$, otherwise $\Pi(N)$ could not be clause-minimal.
 532 Therefore, C, C_1, \dots, C_q cannot all belong to $\Pi(N)$.
- 533 ■ Suppose, then, that C is an instance of some transitivity axiom (3 or 6). Then, no C_j
 534 can be an instance of some irreflexivity axiom (7 or 11), because it would not be positive,
 535 neither can be an instance of any other axiom except 1,4, and 9, because it would not be
 536 unitary; no resolution step can be carried on with 9, because \sim is not transitive, and the
 537 only possible resolution steps that could be completed with the reflexivity of $\dot{=}$ and \preceq
 538 would lead to tautologies. Therefore, every C_j must belong to $\Pi(N)$. If they are all clauses
 539 of the same formula φ , then either $C_1 = X \preceq Y$, $C_2 = Y \preceq Z$, and $q = 2$, in which case
 540 $C = X \preceq Z \in \varphi$ as well because φ is explicit, or $C_1 = X \doteq Y$, $C_2 = Y \doteq Z$, and $q = 2$,
 541 in which case $C = X \doteq Z \in \varphi$ for the same reason. Therefore, C_1 belongs to φ_1 , which
 542 translates some constraint IR_1J , and C_2 belongs to φ_2 , which translates some constraint
 543 JR_2K (if the constraints referred to completely different interval variables, then the
 544 endpoints variables would be different, and no resolution step could be performed). As
 545 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
 546 in both cases $q = 2$. Because N is path consistent, the constraint IR_3K exists, and
 547 $R_3 \subseteq R_1 \circ R_2$. Thus, $\Pi(N)$ also contains its translation φ_3 . Since (i) R_3 is stronger than
 548 $R_1 \circ R_2$, (ii) composition is the systematic application of the transitivity axiom(s) and
 549 the tree-likeness axiom (see Table 1), and (iii) L (which is either $X \preceq Z$ or $X \doteq Z$) can
 550 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
 551 new deduction can be performed.
- 552 ■ Assume, therefore, that C is an instance of the tree-likeness axiom. Then no C_j can be
 553 an instance of some reflexivity axiom (1 or 4), because L would not be new, nor can it
 554 be an instance of some irreflexivity axiom (7 or 11), because it would not be positive.

555 Also, C_j can never be the instance of any other axiom because it would not be unitary.
 556 Therefore, every C_j must belong to $\Pi(N)$. If they are all clauses of the same formula
 557 φ , then $C_1 = X \parallel Y$, $C_2 = Y \preceq Z$, and $q = 2$, in which case we have that $X \parallel Z \in \varphi$ as
 558 well, because φ is explicit. Therefore, C_1 belongs to φ_1 , which translates some constraint
 559 IR_1J , and C_2 belongs to φ_2 , which translates some constraint JR_2K (if the constraints
 560 referred to completely different interval variables, then the endpoints variables would
 561 be different, and no resolution step could be performed). As above, $C_1 = X \parallel Y$ and
 562 $C_2 = Y \preceq Z$, and $q = 2$. Because N is path consistent, there exists a constraint IR_3K ,
 563 and $R_3 \subseteq R_1 \circ R_2$. Thus, $\Pi(N)$ also contains its translation φ_3 . Since (i) R_3 is stronger
 564 than $R_1 \circ R_2$, (ii) composition is the systematic application of the transitivity axiom(s)
 565 and the tree-likeness axiom, and (iii) L (which is $X \preceq Z$) can be deduced from C, C_1, C_2 ,
 566 it must be the case that $L \in \varphi_3$, so, also in this case, no new deduction can be performed.
 567 ■ Finally, suppose that C is any other axiom. C cannot be an instance of a reflexivity
 568 axiom (1 or 4), because it would be unitary and positive, and it cannot be an instance
 569 of any irreflexivity axiom (7 or 11) because C_1 would not be admissible. C cannot be
 570 the instance of any symmetry axiom (2, 8, or 10), because this would entail $q = 1$, which
 571 is to say C_1 would suffice for a deduction, but if C_1 belongs to some formula φ , then
 572 the latter must also contain L because it is explicit. Finally, if C is an instance of some
 573 antisymmetry axiom (5 or 12), then both C_1 and C_2 must refer to the same two endpoints
 574 as C , that is, they must belong to the same formula φ ; therefore, since φ is explicit, L
 575 already belongs to φ , and if it is the instance of some weakening axiom (13, 14, or 15), or
 576 the instance of some compatibility axiom (16, 17, or 18), then C_1 must refer to the same
 577 two endpoints as C , and the same argument applies.

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

579 ◀