Deciding the Consistency of Branching Time Interval Networks

Marco Gavanelli
Department of Engineering
University of Ferrara, Italy
marco.gavanelli@unife.it
https://orcid.org/0000-0001-7433-5899

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
https://orcid.org/0000-0002-9221-879X

Abstract

Allen's Interval Algebra (IA) is one of the most prominent formalisms in the area of qualitative temporal reasoning; however, its applications are naturally restricted to linear flows of time. When dealing with nonlinear time, Allen's algebra can be extended in several ways, and, as suggested by Ragni and Wölfl [20], a possible solution consists in defining the Branching Algebra (BA) as a set of 19 basic relations (13 basic linear relations plus 6 new basic nonlinear ones) in such a way that each basic relation between two intervals is completely defined by the relative position of the endpoints on a tree-like partial order. While the problem of deciding the consistency of a network of IA-constraints is well-studied, and every subset of the IA has been classified with respect to the tractability of its consistency problem, the fragments of the BA have received less attention. In this paper, we first define the notion of convex BA-relation, and, then, we prove that the consistency of a network of convex BA-relations can be decided via path consistency, and is therefore a polynomial problem. This is the first non-trivial tractable fragment of the BA; given the clear parallel with the linear case, our contribution poses the bases for a deeper study of fragments of BA towards their complete classification.

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

Keywords and phrases Constraint programming, Consistency, Branching time

Digital Object Identifier 10.4230/LIPIcs.TIME.2018.12

Acknowledgements G. Sciavicco acknowledges partial support by the Italian INDAM GNCS project *Formal methods for verification and synthesis of discrete and hybrid systems.*

1 Introduction

Allen's Interval Algebra [1] (IA) is one of the most prominent formalisms in the area of qualitative temporal, and also spatial, reasoning. However, its applications are naturally restricted to linear flows of time. Allen's algebra is considered one of the most influential formalisms in qualitative reasoning, and it has found application in a wide range of contexts, such as scheduling, planning, database theory, 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 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. The most relevant problem in IA is deciding whether a network can be satisfied, that is, deciding if every variable of a network can be instantiated to an interval without violating any constraint. The consistency of a network of constraints is archetypical of the class of constraints satisfaction problems (CSP), because a network is a conjunction of constraints; other consistency problems, even in temporal algebras, are more general, and allow some form of disjunction. A very wide range of programming techniques, strategies, and heuristics have been and are being studied to devise efficient implementations; in the particular case of the IA, whose consistency problem is NP-complete, two main strategies have been mainly adopted, that are based either on clever brute-force enumerating algorithms (see, e.g. [13, 23]), or on tractable fragments of the algebra, which are interesting on their own [14] as well as heuristics, aimed to reduce the branching factor in branch-and-bound approaches [15, 19]. Among the several tractable fragments of the IA, a particularly interesting one is known as IA_{convex}, introduced in the early years of this line of research, and encompassing 44 basic and non-basic relations [25]. To obtain IA_{convex}, van Beek and Cohen [25] study, first, the simpler Point Algebra (PA), which has only three basic relations, and define the notion of convexity for a PA-relation; having identified the set of PA_{convex}-relations, they define the set IA_{convex} as the maximal subset of IA-relations with the following property: every network of IA_{convex}-constraints can be translated into an equi-satisfiable network of PA_{convex}-constraints. Then, using the properties of such a translation, they prove that the path consistency algorithm is complete for deciding the consistency of a IA_{convex}-network, obtaining the same result for PA_{convex}-networks as a corollary. Since then, many other tractable fragments of the IA have been discovered, and, in fact, we know the status of every fragment of the IA with respect to the tractability/intractability of the corresponding consistency problem [14].

Branching time has been less studied from the algebraic point of view, in sharp contrast with the huge amount of research on point-based and interval-based temporal logics, such as CTL, CTL*, ATL, or branching PNL [3, 4, 11]. The Branching Point Algebra (BPA) has been studied in [9, 12], and the computational behaviour of the consistency problem of the BPA and its fragments has been analyzed in [5], where, in particular, a polynomial algorithm to decide the consistency of a network of BPA-constraints is given. In [20], the authors define a branching version of Allen's IA, which we refer to as BA (Branching Algebra) 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 basic relations, and also studied from the (first-order) expressive power point of view in [10], is characterized by basic relations whose semantics cannot be always written in the language of endpoints, therefore requiring quantification; on the other hand, the basic relations of the second set are coarser, still jointly exhaustive and mutually exclusive, and their point-based semantics depends only on the relative position of the endpoints. The latter set (BA_{basic}), composed of 19 basic relations, is therefore preferable in many aspects. As expected, the consistency problem for the full BA is NP-complete, and we know only one tractable fragment of it that includes at least one nonlinear relation, that is, the set BA_{basic} itself, since a network of BA_{basic}-constraints can be fully translated into a network of BPA-constraints. In this paper:

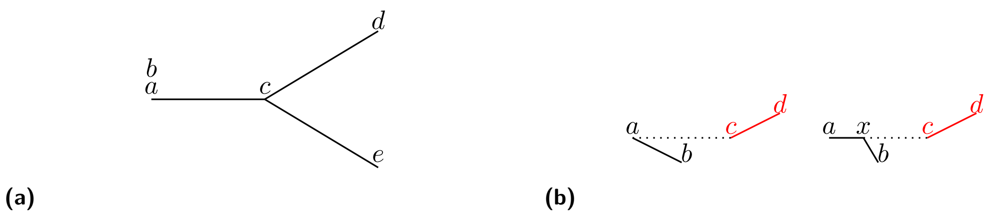


Figure 1 A pictorial representation of the four basic branching point relations, in which $a = b$, $a < c$, $d > c$, and $d||e$ (left-hand side), and two branching relations that need quantification (right-hand side).

(i) in the the spirit of [25], we define the notion of convex branching point relation and the notion of convex branching interval relation, and prove that, as in the linear case, the consistency of a network of convex branching interval (and therefore point) relations can be decided enforcing its path consistency, and

(ii) following [22], 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 convex relations instead of basic relations.

This paper is organized as follows. First we give some necessary preliminaries and notation. In Section 3 we give the main result of this paper, that is, we define convex branching relations and show that the consistency problem can be decided by enforcing path consistency in convex networks. Then, in Section 4, we give some experimental evidence that convex branching relations can be used to speed up the process of deciding branching relation networks, before concluding.

2 Preliminaries

Notation. Let $(\mathcal{T}, <)$ be a partial order, whose elements are generally denoted by $a, b, \ldots$, and where $a||b$ denotes that a and b are incomparable with respect to the ordering relation $<$. We use $x, y, \ldots$ to denote variables in the domain of points. 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 $c > a$ and $c > b$ (that is, it is a tree). In a branching model $(\mathcal{T}, <)$, any maximal linearly ordered subset $\mathcal{B}$ of $\mathcal{T}$ will be called *branch*. 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 Fig. 1a, 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 $I, J, \ldots$, and we use $X, Y, \ldots$ to indicate variables in the domain of intervals. Following Allen [1], we adopt the so-called *strict interpretation* by asking that intervals with coincident endpoints are excluded. In the case of linear time, a theory that encompasses both intervals and *points* has been presented in [7]. There are several ways to define basic relations between intervals on a branching order. Following [10], 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

b (bi)	I $before$ J	$b < c$
m (mi)	I $meets$ J	$b = c$
o (oi)	I $overlaps$ J	$a < c < b < d$
d (di)	I $during$ J	$c < a < b < d$
s (si)	I $starts$ J	$a = c < b < d$
f (fi)	I $finishes$ J	$c < a < b = d$
e	I $equals$ J	$a = c < b = d$
ib (ibi)	I $init.before$ J	$a < c \| \| b$
im (imi)	I $init.meets$ J	$a < c < b \| \| d$
ie	I $init.equals$ J	$a = c < b \| \| d$
u	I $unrelated$ J	$a \| \| c$

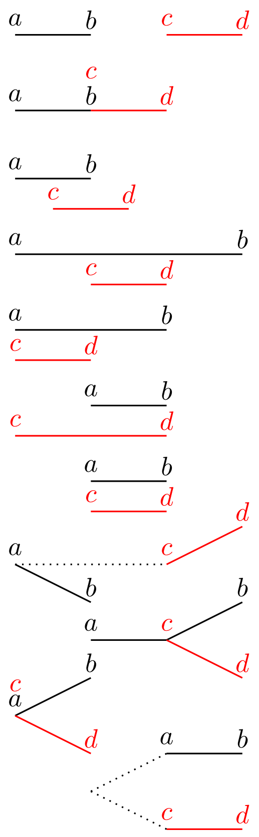


Figure 2 A pictorial representation of the nineteen basic branching interval relations. In this picture, we assume $a < b$ and $c < d$. Solid lines are actual intervals, dashed lines complete the underlying tree structure.

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 Fig. 1b we see that, in order to distinguish the two situations, we need to quantify of the existence, or non-existence, of points, comparable with c, 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 [20] introduce a set of coarser relations, characterized by being translatable to point-based relations using only the language of endpoints, that is, without quantification. These 19 relations are depicted in Fig. 2, and form the set of *basic branching interval relations* BA_{basic}; for each relation, the symbol in brackets corresponds to its inverse one, 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 fine relations. For example, the relation im is the disjunction of the two relations in Fig. 1b.

Operations and algebras. The set BPA of *branching point relations* is the set of all possible non-empty disjunctions of basic branching point relations, and it encompasses $2^4 - 1 = 15$ relations. Similarly, the set BA of *branching interval relations* is the set of all possible disjunctions of basic branching interval relations, and it encompasses $2^{19} - 1$

Table 1 Composition of basic BPA-relations (left-hand side), and of basic PA-relations (right-hand side).

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

$\circ$	$<$	$>$	$=$
$<$	$\{<\}$	lin	$\{<\}$
$>$	$?$	$\{>\}$	$\{>\}$
$=$	$\{<\}$	$\{>\}$	$\{=\}$

relations. 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. A BPA-*constraint* is an object of the type xRy, where x, y are point variables and $R \in BPA$; analogously, a BA-*constraint* is an object of the type XRY, where X, Y are interval variables and $R \in BA$. 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 i, r_i^{-1} is the inverse basic relation of the basic relation r_i. In our notation, for example, bi (*later*) is 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: $R_1 \circ R_2 = \{r \mid \exists r_1 \in R_1 \exists r_2 \in R_2(r = r_1 \circ r_2)\}$. Clearly, to compute the composition of two non-basic relations we base ourselves on the composition between basic relations. As for the set BPA_{basic} (resp., the set PA_{basic}), the composition table can be easily computed 'by hand', as in Tab. 1, left-hand side (resp., right-hand side), where we use the abbreviations $lin = \{<, =, >\}$ and $? = lin \cup \{||\}$. The result of composing two relations in the set BA_{basic} (resp., IA_{basic}) can be computed automatically from Tab. 1, and it is fully reported in [21] (resp., [2]).

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, and two networks N and M are said to be *equi-satisfiable* if it happens that N is consistent if and only if M is consistent. Given a constraint (s, t) in a network N, we say that $r \in (s, t)$ is *feasible* if there exists a model of N such that s and t are realized respecting r, and a constraint (s, t) is said to be *minimal* if every $r \in (s, t)$ is feasible and (s, t) cannot be extended with other feasible relations; establishing the minimal constraints for every constraint in a network is called the *minimal labels* problem. Enforcing the minimal label in a network implies deciding its consistency, but, in general, we may have a consistent network with non-minimal labels. The operations of inverse, intersection, and composition can be used to design a constraint satisfaction problem (CSP) technique to decide the consistency of a A-network. The sets BPA and BA are called, respectively, the *branching point algebra* and the *branching interval algebra*, and they extend, respectively, the *interval algebra* IA and the *point algebra* PA. Since the consistency problem for the IA is NP-complete [26], the problem of finding tractable fragments of it is interesting, and it has been largely studied in the recent literature [1, 25, 14]. The branching setting presents a similar situation, as the consistency problem for the BA is NP-complete as well, but, in contrast with the linear case, only one tractable fragment is known, that is, BA_{basic} [20]; the consistency of a network of basic branching interval constraints can be decided by translating it into an equi-satisfiable network of BPA-constraints, for which a deterministic polynomial consistency algorithm exists [5].

Local consistency. On the one hand, the BA-consistency problem is in NP because there exists a simple non-deterministic algorithm that solves it, which, given a BA-network $N = (V, E)$, guesses the relative position of $2 \cdot |V|$ points and checks if every constraint is respected. On the other hand, these problems are often solved 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; it is said to be *strongly* k-consistent if it is k'-consistent for every $k' \leq k$ (see [17]); if a network is strongly k-consistent, then it must also have minimal labels. 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) does not imply consistency; this is true for BA-networks as well as for IA-networks. In [25], however, it is proven that enforcing path consistency is equivalent to computing the minimal labels of a IA_{basic}-network, which, in turn, allows one to check the existence of a model. This property of path consistency is shown for a more general set of relations, called *convex interval relations*, which are defined starting from the set PA_{convex} of *convex point relations*, and, in particular, it is proved that the set IA_{convex} (the *convex interval algebra*) includes IA_{basic}, and that its consistency problem (and, as a corollary, the consistency of a PA_{convex}-constraints) can be decided by path consistency. This result, particularly interesting for us, has the following consequences. First, IA_{convex} is a fragment of the IA with a tractable (in fact, cubic time) consistency problem. Second, one can implement a simple branch-and-bound algorithm to decide the consistency of any IA-network, based on IA_{basic}: at each step, the algorithm tries one basic relation for each relation, and then forces the path consistency of the resulting network; if at any step the network is path consistent, it returns *true*, and if every combination has been tried and enforcing path consistency has always resulted in an empty relation, it returns *false*. Third, the set IA_{convex} 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 IA_{convex}-network, that particular branch can be decided by simply enforcing path consistency.

In the following, we shall define the algebra BA_{convex} of convex branching relations. We shall prove that, since BA_{convex} extends BA_{basic}, and since enforcing path consistency can be used to decide the consistency of a BA_{convex}-network, one can apply the same schema, effectively lifting all above results to the setting of branching time.

A motivating scenario. *Scheduling* is the problem of distributing computing resources (such as processor time, bandwidth, or memory) to various processes, threads, data flows, and applications that need them. In robotics, scheduling is used to organize tasks to be assigned to robots of various kinds, in such a way that all physical and subjective constraints are met. A recent application of Allen's Interval Algebra to the scheduling of tasks for a robot has been proposed by Mudrová and Hawes [18]. The authors propose a scheduling technique that takes into account a series of constraints, including *deadlines* and *processing time* for each one of a series of tasks that a robot is asked to complete. They propose to apply a consistency checking algorithm to the network of qualitative constraints that underlies the scheduling problem, in order to prune any ordering of the tasks that does not meet the qualitative constraints, and to be able to select, systematically, possible models of the problem. To each of the models, then, a successive phase of quantitative constraint checking is applied.

Mudrová and Hawes solve the scheduling problem of a *single robot*, which encompasses assuming the time to be linear and introducing the additional constraint that no two tasks can overlap. Using Branching Algebra instead of Interval Algebra as part of the scheduling algorithm, and using the relation $||$ to relax, when possible, the non-overlapping constraint, one may obtain, instead, branching models among the solutions. A branching model of the scheduling can be interpreted as a scheduling in which *more than one robot* is involved, that is, in which every new branch implicitly refers to a new robot being activated (at the branching point), and, if we assume that *bootstrapping* a robot has some fixed cost (higher than maintaining a robot active), then it makes sense to look for a branching scheduling as we have defined it, that is, tree-like (in which branches never join again). Thus, Branching Algebra allows one to generalize this scheduling problem to a more complex scenario, which cannot be easily handled in the original formulation.

3 Convex Branching Interval Relations

We start by defining the concept of convex relation in the branching setting. In this section, we operate with translations from interval constraints to point constraints; when necessary, for an interval variable X, we use the symbols X^-, X^+ to denote the point variables that correspond to its endpoints.

▶ **Definition 1.** The *convex branching point algebra* is the set of relations:

$$BPA_{convex} = \{\{=\}, \{<\}, \{<, =\}, \{<, =, >\}, \{=, >\}, \{>\}, \{||\}\}.$$

Each relation of set BPA_{convex} is called *convex branching point relation.*

Observe that BPA_{convex} extends the convex point algebra PA_{convex} as defined in [25] by adding the relation $\{||\}$. While the set PA_{convex} is closed under composition, inverse, and intersection, the set BPA_{convex} is closed under under inverse and intersection only; this, however, does not prevent us from applying constraint propagation algorithms such as path consistency enforcing.

▶ **Definition 2.** The *convex branching interval algebra* BA_{convex} is the set of all (and only) BA-relations R such that the constraint XRY can be translated to an equi-satisfiable conjunction of constraints between the endpoints of X and Y using BPA_{convex}-relations only. A branching relation with such a property is called *convex branching interval relation.*

For example, in the linear setting, $\{b, m\}$ is convex because, if $X = [X^-, X^+]$ and $Y = [Y^-, Y^+]$, the constraint $X\{b, m\}Y$ is equivalent to the conjunction of the constraints $X^- < X^+$, $Y^- < Y^+$, and $X_1^+\{<, =\}Y^-$; conversely, the relation $\{b, bi\}$ is not convex, because translating it results in a disjunction of point-based constraints. Clearly, the set BA_{convex} extends the set IA_{convex} of convex interval relations as defined in [25]. The

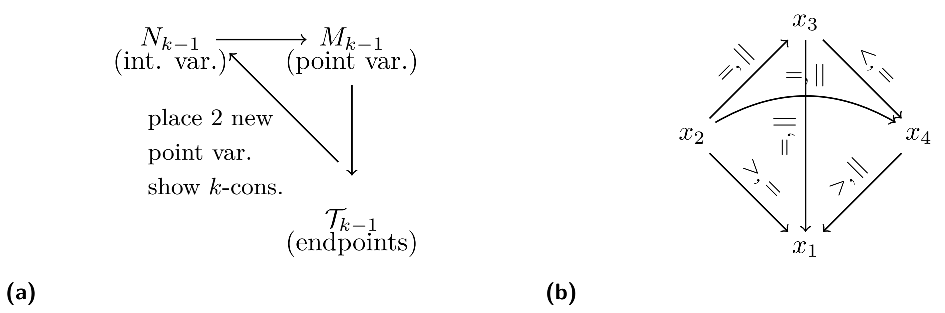


Figure 3 A general view of the strategy for the inductive case of Theorem 5 (left-hand side), and a path-consistent network with non-minimal labels (right-hand side).

crucial property of a BA_{convex}-network N is that it can be translated to an equi-satisfiable BPA_{convex}-network M (notice that this is not true for a generic BA-network: non-convex constraints may result in disjunctions that cannot be represented in the language of full BPA-networks), such that, if N is k-consistent for any k, then M is $2 \cdot k$-consistent. As it can be easily checked, there are precisely 91 convex branching interval relations. In the linear case, the following results hold [25].

▶ **Theorem 3.** *Enforcing path consistency in a IA_{convex}-network is sufficient to compute its minimal labels.*

▶ **Corollary 4.** *Enforcing path consistency in a PA_{convex}-network is sufficient to compute its minimal labels.*

Our purpose in this section is to generalize the above theorem to the branching case.

We want to prove that we can decide the consistency of a BA_{convex}-network by enforcing its path consistency. To this end, we prove that enforcing path consistency of a BA_{convex}-network actually enforces the minimal labels on each constraint by proving that, in fact, enforcing path consistency of a network entails enforcing its strongly k-consistency for every k. As we have already observed, this allows us to check the consistency of a network.

▶ **Theorem 5.** *Enforcing path consistency in a BA_{convex}-network is sufficient to compute its minimal labels.*

Proof. Let N be a BA_{convex}-network, and let M be the BPA_{convex}-network that results from translating N in the language of endpoints. We assume that path consistency has been forced on N, and we want to show that N is strongly k-consistent for every k; since a strongly k-consistent network must have minimal labels, we have the result. Let us proceed by induction. As base case, we know that N is k-consistent for $k \leq 3$. As for the inductive case, we suppose now that N is $k-1$-consistent and we prove that it is also k-consistent. Consider a subset S of $k-1$ interval variables in N. Let us call N_{k-1} the sub-network, with the $k-1$ interval variables $X_1, \ldots, X_{k-1}$, corresponding to the projection of N over set S, and let us call M_{k-1} the corresponding BPA_{convex}-network whose variables are precisely the $2 \cdot (k-1)$ endpoints of $X_1, \ldots, X_{k-1}$. Our strategy, as sketched in Fig. 3a, can be summarized as follows: since N_{k-1} is consistent by hypothesis, M_{k-1} must be consistent as well, that is, it must realized in a branching model $\mathcal{T}_{k-1}$; if we pick the point variables corresponding to the endpoints of any k-th interval variable and accommodate them in $\mathcal{T}$ showing that every constraint is respected, then we obtain a branching model for k interval variables, proving that N_k is also consistent. Let X be any interval variable in N different from $X_1, \ldots, X_{k-1}$, and let XR_iX_i the BA_{convex}-relation between the variables X and X_i, for each i. Let M_k the BPA_{convex}-network obtained by adding to M_{k-1} the point variables X^-, X^+, the constraint $X^- < X^+$, and every constraint between the endpoints of X and the endpoints of $X_1, \ldots, X_{k-1}$ that results from translating the constraints of the type XR_iX_i. On $\mathcal{T}_{k-1}$ we can identify the set $\mathcal{R} = \{a_1, \ldots, a_n\}$ with the following characteristics: for each i, a_i is the realization of some point variable y in M_{k-1} (that is, a_i is the realization of some endpoint of the interval variables $X_1, \ldots, X_{k-1}$) and that, for every point variable $y \in M_{k-1}$, realized in some point $a \in \mathcal{T}_{k-1}$, it is not the case that $a < a_i$. Indeed, consider the branching model $\mathcal{T}_{k-1}$: since it must be a tree, it may be the case that, in order to realize two variables that are constrained to be incomparable to each other, a greatest common predecessor must be added; therefore, if projected to the points that realize some point variable in N_{k-1}, $\mathcal{T}_{k-1}$ is a *forest of trees*, rather then a tree. Every point in $\mathcal{R}$ is the *root* of one of the trees in $\mathcal{T}_{k-1}$; let us call c their greatest common predecessor. Now, let $x_1, \ldots, x_m$ be the point variables that have been realized in $a_1, \ldots a_n$ (observe that $n \leq m \leq k-1$: two variables may have been realized in the same root, and m cannot exceed k_1 because, at most, every interval variable has its left endpoint realized in a root). We want to show, first, that the point variable X^- can be successfully realized on $\mathcal{T}_{k-1}$, and we proceed case by case.

- Suppose, first, that $(x_l, X^-) = \{||\}$ for every point variable x_l realized in some root. In this case, we realize X^- with a new point a such that $a||a_i$ for each root a_i, and that $c < a$. To prove that this is a consistent choice, consider any point variable y of M_{k-1} realized at some point $b \geq a_i$ for some root a_i. Suppose that a_i is the realization of some point variable x_l, which means that $(y, x_l) \subseteq lin$. If $<\in (y, x_l)$, then $(y, x_l) \circ (x_l, X^-) = \{<, ||\}$. By intersection with BA_{convex}, then either $(y, X^-) = \{<\}$ or $(y, X^-) = \{||\}$; in the first case, however, we obtain, by path consistency, that $(X^-, x_l) \in lin$, which is a contradiction. Therefore, $(y, X^-) = \{||\}$. If , on the other hand, $<\notin (y, x_l)$, then, $(y, x_l) \subseteq \{>, =\}$, and, since $\{>, =\} \circ \{||\} = \{||\}$, it must be the case that $(y, X^-) = \{||\}$.
- Suppose, now, that $(x_l, X^-) \subseteq lin$ for some point variable x_l realized in some root a_i. Observe, first, that if $(X^-, x_l) = \{<\}$, then we can select the subset $\mathcal{R}' \subseteq \mathcal{R}$ such that, for each $x'_l \in \mathcal{R}'$, we have that $(X^-, x'_l) = \{<\}$; in this case, by the argument in the above case, for each $x''_l \in \mathcal{R} \setminus \mathcal{R}'$, we have that $(X^-, x'_l) = \{||\}$. Consider each a_j that is the realization of some variable in $\mathcal{R}'$: we realize X^- in a point $a > c$, such that a is less than every such a_j, and incomparable with every other root in $\mathcal{R} \setminus \mathcal{R}'$. If, otherwise, $(X^-, x_l) \subseteq \{>, =\}$, then, for each x'_l realized in some root $a_j \neq a_i$, we must have $(X^-, x'_l) = \{||\}$. In this case, we can say that a_i is the root of the tree in which we have to realize X^-; let us call it $\mathcal{T}_{a_i}$. Observe that, by the same argument as in the above case, wherever we realize X^- in $\mathcal{T}_{a_i}$, this realization is consistent with any point that belongs to some $\mathcal{T}_{a_j}$ with $a_j \neq a_i$. Now, we consider the point $b \in \mathcal{T}_{a_i}$ which is the least point (greater than or equal to a_i) with at least two immediate successors b_1, b_2 such that $b_1||b_2$, if it exists. We have the following cases.
  - Suppose that b does not exists. This means that T_{a_i} is linearly ordered. Let b' be the least point (greater than a_i), such that is the realization of some variable y such that $(y, X^-) = \{||\}$. If there is no such b', then, by Theorem 3, we can find a realization for X^- consistent with $\mathcal{T}_{a_i}$; since we already know that such a realization is consistent with every other tree, we conclude that it is consistent. If b' exists, then we realize X^- in a point a such that $a||b'$ and that $a > d$ where d is the immediate predecessor of b'. By the argument used in the first case, this choice must be consistent with $\mathcal{T}_{a_i}$, and therefore it must be consistent.

- Suppose, now, that b exists. If y is realized in b, and $\{<, =, ||\} \cap (X^-, y) \neq \emptyset$, then we proceed as in the previous case. If, on the other hand, $(X^-, y) = \{>\}$, we have the following two cases. First, if $<\in (X^-, y)$ for every y_j realized in some point b_j such that b_j is immediate successor of b, then realize X^- in a point a such that $b < a$ and that $a < b_j$ for every immediate successor b_j of b, which must be a consistent choice, given that, by path consistency, the relation between X^- and every variable realized in a point greater than b must contain $<$. If, for some immediate successor b_j of b, which is the realization of some variable y, it is the case that $<\notin (X^-, y)$, then we can treat every immediate successor b_j of b as the root of some sub-tree of T_{a_i}, and therefore we can apply the same entire argument, recursively.

Having realized the variable X^-, the network M_{k-1} enriched with X^- (and all relative constraints) must be consistent. By reapplying the entire argument, we can show that any other point variable can be consistently realized in the resulting network; if we choose X^+ among these, we prove that the original network N is, in fact k-consistent, completing the induction. ◀

▶ **Corollary 6.** *Enforcing path consistency in a BPA_{convex}-network is sufficient to compute its minimal labels.*

It would be natural, at this point, to ask whether the set BPA_{convex} can be enriched while retaining the above nice properties. A natural candidate in this perspective would be the set of all BPA-relations such that, for each R, it is the case that $R \cap lin$ (which is a linear relation) is convex. This set is closed under composition, inverse, and intersection, allowing one to approach its consistency problem in the same way as we did in this work; in particular, it would be possible to define the set of BA-relations that can be translated in this language, obtaining, in fact, a greater set of branching relation whose minimal labels could be enforced by path consistency. Unfortunately, there is an easy counter-example to this claim, shown in Fig. 3b, in which we have a path consistent network but with non-minimal labels. Therefore, if there exists any extension of BPA_{convex} whose minimal labels can be enforced by path-consistency, it cannot include at least one of the relations in Fig. 3b.

Observe that enforcing the minimal labels via path consistency is not the only way to prove that the consistency of a network can be decided via path consistency, and it is certainly not the only way to prove that a fragment of relations is tractable. For example, the tractability of the consistency problem for a network of IA-relations in the ORD-$Horn$ fragment is proven via embedding into the Horn fragment of propositional logic [14]; as an another example, the tractability of the consistency problem for a network of full BPA-relations is proven with a specialized algorithm in [5], and it is the basis for Ragni and Wölfl's result about the tractability of the consistency problem for a network of BA_{basic}-relations.

4 Experiments

In order to evaluate the usefulness of the convex fragment as a heuristics for the task of checking the consistency of a BA-network, we devised a set of experiments, following the classical methodologies in the literature.

To generate a random set of instances, we used a (modification of) a technique suggested by Renz and Nebel [22] that consists of the following steps. Given a number n of nodes, an average density d and a probability p, we generate a random instance as follows:

Algorithm 1 Backtracking algorithm.

```
function Consistent($P$, $Split$)
    enforce generalized arc consistency on $P$
    if there is a variable $\nu_{XY}$ such that $\mathcal{D}_{XY} = \emptyset$ then
        return $false$
    else
        choose an unprocessed variable $\nu_{XY}$ such that $\mathcal{D}_{XY} \notin Split$
        if there is no such variable then
            return $true$
        $\{\mathcal{D}_1, \ldots, \mathcal{D}_p\}$=Partition($\mathcal{D}_{XY}$, $Split$)
        for all $\mathcal{D}_i \in \{\mathcal{D}_1, \ldots, \mathcal{D}_p\}$ do
            $P' = P_{\mathcal{D}_{XY}/\mathcal{D}_i}$
            if Consistent($P'$, $Split$) then
                return $true$
        return $false$
```

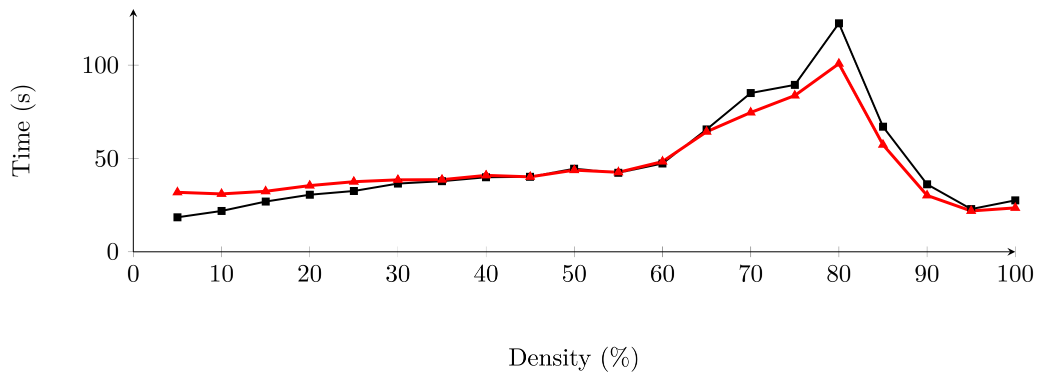
(i) we generate a graph with n nodes, and select $\frac{d \cdot n \cdot (n-1)}{2}$ edges at random;
(ii) for each selected edge (s, t), we generate a BA-relation R by selecting, with probability p, each BA_{basic}-relation to be inserted in R, and
(iii) to each non-selected edge (s, t), we assign the universal relation.

As much as the solver is concerned, we based ourselves on the general strategy of *constraint logic programming on finite domains*, by means of which we are able to check the satisfiability of a temporal network. The solver itself is based on the dual CSP, encoded with translation into ternary constraints, as proposed by Condotta et al. [6]. In particular, given an A-network $N = (V, E)$, we define a CSP P as a triple $\langle \mathcal{V}, \mathcal{D}, \mathcal{C} \rangle$, such that:

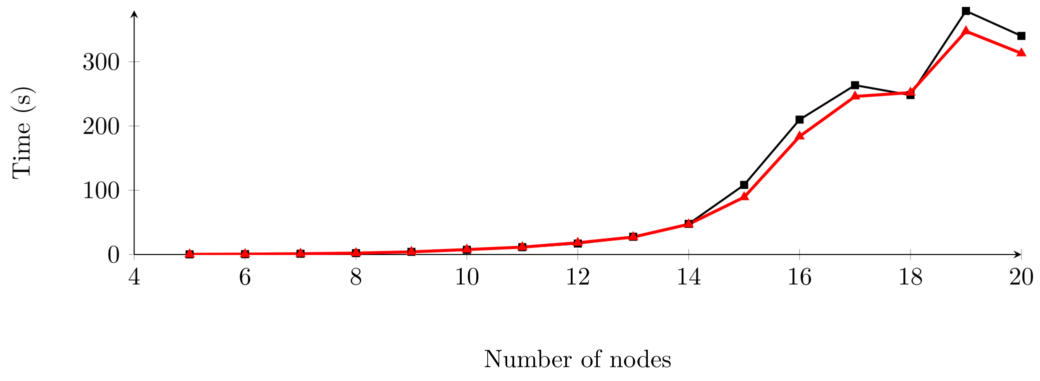
- the set $\mathcal{V}$ contains a CSP variable ν_{XY} for each pair of variables X, Y in V;
- the domain $\mathcal{D}_{XY}$ of each variable ν_{XY} is precisely the constraint (X, Y), and
- for each triple of variables, $\mathcal{C}$ contains a ternary constraint, so that each ternary constraint will be satisfied by a triple $(r, r_1, r_2) \in BA^3_{basic}$ if and only if $r \in r_1 \circ r_2$.

As noted by Condotta et al. [6], enforcing the (generalized) arc consistency on the problem P is equivalent to enforcing path consistency on the original A-network. Since, as we know, both path consistency and (generalized) arc consistency are incomplete algorithms for BA-networks, a backtracking search is applied, and to each node of the search tree, (generalized) arc consistency is enforced. The generic schema of a backtracking algorithm can be described as in Alg. 1 [19].

In Alg. 1, the family of sets *Split* plays a key role. When solving the general CSP, without exploiting any tractable fragment of the BA, *Split* can be thought as containing all singletons, each one of them corresponding to a single BA_{basic}-relation. Therefore, the technique to solve a problem P consists of simply choosing a variable, substitute its domain with one of its components creating a new problem P', and recursively solve P'. This algorithm is correct because the search terminates with a node with basic relations only, for which path consistency is a complete method. When a bigger fragment for which enforcing path consistency is known to be complete for consistency, we can take advantage from it by setting *Split* to be the family of its relations; in such a case Alg. 1 stops the search even if the domain of some of the CSP variables is not a singleton, obtaining, *de facto*, a smaller branching factor of the search tree. Unfortunately, establishing if a set can be partitioned into smaller sets taken from some family of sets corresponds to the *set partitioning*



🟨 **Figure 4** Geometric mean of the computation time to solve the instances varying the density d of the network. Number of nodes $n = 15$, 20 instances solved per point. The black squares show the running time of the backtracking algorithm with $\mathit{Split} = BA_{basic}$, while the red triangles represent the curve with $\mathit{Split} = BA_{convex}$.



🟨 **Figure 5** Geometric mean of the computation time to solve the instances varying the number of nodes n of the network. Density $d = 70\%$, 20 instances solved per point. The black squares show the running time of the backtracking algorithm with $\mathit{Split} = BA_{basic}$, while the red triangles represent the curve with $\mathit{Split} = BA_{convex}$.

problem, which is NP-complete in general (in Alg. 1 this step is encoded into a function $\textsc{Partition}(\mathcal{D}, \mathit{Split})$). In our case, the domain can be partitioned into basic relations (so a polynomial solution exists), but such a solution is inconvenient, as it does not exploit the tractability of the fragment. Since this problem should be solved in every node of the search tree, a quick, although non-optimal, method is mandatory to obtain reasonable efficiency. A first solution would be to pre-compute the (possibly, optimal) solutions of the set-partitioning for each possible subset of the domains; this would generate a table of size $2^{|BA_{basic}|}$, which may fit into the main memory of modern computers, but poses the problem of efficiently accessing to the data structure that contains it. Another option is to use a greedy method to quickly provide a possibly non-optimal partitioning (note, also, that it is not necessary for $\textsc{Partition}(\mathcal{D}, \mathit{Split})$ to return the complete partitioning, as each of the sets $\mathcal{D}_1, \ldots, \mathcal{D}_p$ can be generated on demand). We decided to follow the latter approach and to store the set of convex relations into a *trie*, which is a data structure whose access time depends on the order in which the elements of a set are stored, but that resulted efficient in practice.

The algorithm was implemented in the constraint logic programming system ECL^{i}PS^e [24], using the CLP(FD) library. To implement the ternary constraints we used the `propia`

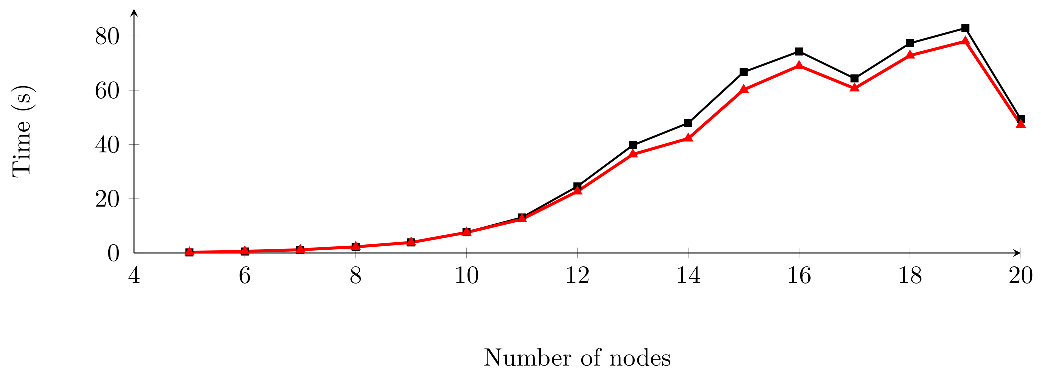


Figure 6 Geometric mean of the computation time to solve the instances varying the number of nodes n of the network. Each point is the geometric average of 140 instances, obtained with density varying from 70% to 100%. The black squares show the running time of the backtracking algorithm with $Split = BA_{basic}$, while the red triangles represent the curve with $Split = BA_{convex}$.

library [16] that provides a general and very declarative way to implement new constraints, although we are aware that more efficient implementations could be possible. The objective of the experimentation was discussing the relative improvement given by the exploitation of the convex fragment, rather then evaluating the absolute performances of our implementation. All experiments were run on a Intel Core i7-3720QM CPU @ 2.60GHz running ECL^{i}PS^e Version 6.1 #224 on Linux Mint 18.1 Serena 64 bits, and using only one core. Timeout was fixed to 10 minutes.

In Fig. 4, we fixed the number of vertices of the A-network to $n = 15$, the probability $p = 1/2$ and varied the constraint density d from 5% to 100%. Each point in the curve represents the geometric mean obtained running 20 instances. Statistically speaking, a small number of *very difficult* networks may be produced in a set of 20 random instances; the neat effect on the computation time of such instances can be softened using the geometric mean rather then the, more common, arithmetic mean [8]. The shape of the curve shows the expected *phase transition*: when the density is low, most of the instances are easily satisfiable, while to high density correspond networks for which proving unsatisfiability is easy. The phase transition occurs at a density around $d = 80\%$, in which both satisfiability and unsatisfiability are hard to prove. The curves in Fig. 4 (in which the red curve represents the performance of the algorithm when the convex fragment is taken into account) show that exploiting the convex fragment is particularly convenient for hard problems near the phase transition, while the overhead that, implicitly, is introduced in such a solution makes it not worth for easily satisfiable problems. In Fig. 5 we fixed the density to a point close to phase transition ($d = 70\%$), and varied the number of nodes in the graph, from 3 to 20. Each point is the geometric mean of 20 runs. At 70% density, most problems under 20 nodes are difficult, and, again, exploiting the convex fragment is convenient with respect to the expected performance. Finally, we investigated the computation time of the two solutions varying the number of nodes (from 5 to 20) independently of the density. To this end, we generated, for each number of nodes, 140 instances with densities varying from 70% to 100%, and considered the geometric mean of the time needed to solve them. In Fig. 6 we show the result of such an analysis, that proves that a certain improvement in computation time exists when the convex fragment is taken into account.

5 Conclusions

Allen's Interval Algebra is one of the most prominent formalisms in the area of qualitative temporal reasoning. However, its applications are naturally restricted to linear flows of time, raising the question of whether one can reason about branching (tree-like) flows of time in a similar manner. We considered, in this paper, the set of 19 branching relations suggested by Ragni and Wölfl, which enjoy the desirable characteristics of being expressible in the language of endpoints without quantification. Ragni and Wölfl have shown that the consistency problem for a network of branching relations is intractable (as expected), while the consistency problem for a network of basic branching relations is polynomial. In clear parallelism with the linear case, we defined the set of convex branching relations, which extends the set of basic branching relations, and we proved that enforcing path consistency of a network of convex relations is sufficient to decide its consistency, effectively providing the first non-trivial tractable (via path consistency) fragment of the branching algebra. As another consequence of this work, we made it possible to treat the consistency problem of a network of constraints as a constraint propagation problem, allowing not only the possibility of quick implementations using well-known libraries, but, also, the possibility of implementing a clever branch-and-bound algorithm for a generic network that exploits the tractability of convex relations as an heuristics. Finally, we tested such a solution, giving experimental evidence of the expected improvement.

The most interesting open problems at the moment include, among other, the question of whether the convex branching algebra is maximal with respect to tractability of the network consistency problem (which seems unlikely) and/or with respect to the possibility of enforcing the minimal labels of a network via path consistency, the question of whether other popular and well-behaved fragments of the interval algebra in the linear case can be generalized to the branching setting preserving their computational behaviour, and the question of whether efficient enumerating algorithms can be devised for the branching case as it has been done in the linear case. We already know that there is a set of branching point relations which could be considered a natural generalization of convex branching point relations and whose minimal labels problem cannot be solved by path consistency; however, fragments of the branching point algebra that strictly include BA_{convex} and for which the minimal labels of a network can be enforced by path consistency are still possible.

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 R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

4 D. Bresolin, A. Montanari, and P. Sala. An optimal tableau for right propositional neighborhood logic over trees. In *Proc. of TIME 2008: 15th International Symposium on Temporal Representation and Reasoning*, pages 110–117. IEEE, 2008.

5 M. Broxvall. The point algebra for branching time revisited. In *Proc. of KI2001: Advances in Artificial Intelligence*, volume 2174 of *Lecture Notes in Artificial Intelligence*, pages 106–121. Springer, 2001.

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

7 W. Conradie, S. Durhan, and G. Sciavicco. An integrated first-order theory of points and intervals: Expressive power in the class of all linear orders. In *Proc. of TIME 2012: 19th International Symposium on Temporal Representation and Reasoning*, pages 47–51. IEEE, 2012.

8 M.J. Dent and R.E. Mercer. A new model of hard binary constraint satisfaction problems. In *Proc. of AI 96: 11th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, volume 1081 of *Lecture Notes in Computer Science*, pages 14–25. Springer, 1996.

9 I. Düntsch, H. Wang, and S. McCloskey. Relations algebras in qualitative spatial reasoning. *Fundamenta Informaticae*, 39(3):229–248, 1999.

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

11 E. Allen Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 995–1072. MIT Press, 1990.

12 R. Hirsch. Expressive power and complexity in algebraic logic. *Journal of Logic and Computation*, 7(3):309–351, 1997.

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

14 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.

15 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.

16 T. Le Provost and M. Wallace. Generalized constraint propagation over the CLP scheme. *Journal of Logic Programming*, 16(3):319–359, 1993.

17 A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

18 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.

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

20 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.

21 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.

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

23 J. Renz and B. Nebel. Qualitative spatial reasoning using constraint calculi. In M. Aiello, I. Pratt-Hartmann, and J.F.A.K. van Benthem, editors, *Handbook of Spatial Logic*, pages 161–215. Springer, 2007.

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

25 P. van Beek and R. Cohen. Exact and approximate reasoning about temporal relations. *Computational Intelligence*, 6:132–144, 1990.

26 M. B. Vilain and H.A. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proc. of AAAI 1986: 5th National Conference on Artificial Intelligence*, pages 377–382, 1986.