On Binary Constraint Networks

Peter B. Ladkin*
Kestrel Institute
1801 Page Mill Road
Palo Alto, Ca 94304-1216

Roger D. Maddux
Department of Mathematics
Iowa State University
Ames, Iowa 50011

Abstract

It is well-known that general constraint satisfaction problems (CSPs) may be reduced to the binary case (BCSPs) *[Pei92]*. CSPs may be represented by binary constraint networks (BCNs), which can be represented by a graph with nodes for variables for which values are to be found in the domain of interest, and edges labelled with binary relations between the values, which constrain the choice of solutions to those which satisfy the relations (e.g. *[Mac77]*). We formulate networks and algorithms in a general algebraic setting, that of Tarski's relation algebra *[JonTar52]*, and obtain a parallel $O(n^2 log n)$ upper bound for path-consistency, and give a class of examples on which reduction-type algorithms (which include the standard serial algorithms *[Mac77, MacFre85, MohHen86]* and all possible parallelisations of them) are $O(n^2)$. We then consider BCNs over various classes of relations that arise from an underlying linearly ordered set, the most well-known being the interval algebra *[All83, LadMad88.1]*. There are three main consequences of the algebraic approach. Firstly, it puts the theory of BCNs on a firm (and classical) theoretical footing, enabling, for example, the complexity results. Secondly, we can apply techniques from relation algebra to show that consistency checking for a large class of relations on intervals (*[All83]*) is serial cubic, or parallel log time, significantly extending previous results (the problem is NP-hard in general *[VilKau86]*). Thirdly, results are obtained via a new construction of relation algebras from other algebras which is of independent mathematical interest.

*The first author was partially supported by DARPA contract N00039-88-C-0099, administered by the U.S. Navy. The views and conclusions expressed in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of Kestrel Institute, or any agency of the United States Government.

1 Introduction

We are concerned with Boolean constraint satisfaction problems *[Mac87]* with binary relations, which we call *BCSP*s. It is well-known *[Pei92]* that binary relations are logically sufficient to represent n-ary relations for $n \geq 2$, and hence techniques for binary constraint satisfaction have been emphasised in the literature. A number of algorithms have been proposed for constraint satisfaction in exact domains *[Mac77, Fre78, All83, MohHen86]*. We consider an algebraic approach to BCSPs, by considering *binary constraint networks*, or *BCN*s, in algebras of relations. We argue that a class of relations over which a particular BCSP may be framed defines a *relation algebra* in the sense of Tarski *[JonTar52, Mad82]*. This relation algebra may have many different representations, and the BCN associated with the BCSP may thus be interpretable over different domains than the one in which the BCSP is formulated. Thus we distinguish algebraic properties of a BCN from solvability of the associated BCSP, which may be solvable or not depending on the domain of interpretation. We formulate standard consistency and satisfiability properties, and algorithms for these, in this general algebraic framework. For example, we show there exists a most general path-consistent reduction of a given BCN A (the $MGR_3(\mathcal{A})$), and a parallel $O(n^2\ log\ n)$ upper bound for computing it. Further, we exhibit a class of examples on which all the reduction-type algorithms for $MGR_3(\mathcal{A})$ are $O(n^2)$. Reduction-type algorithms include the well-known serial algorithms *[Mac77, MacFre85, MohHen86]* and parallelisations of these with varying search topologies. Reduction-type algorithms are those which, loosely speaking, calculate $MGR_3(\mathcal{A})$ by calculating $MGR_3(\mathcal{A}')$ on subnetworks $\mathcal{A}'$, and iterating until a fixpoint is reached. (This seems to be the most common way to calculate fixpoints of anything). The lower bound for reduction-type algorithms of $O(n^2)$ is aymptotically not much better than serial ($O(n^3)$), the known upper limit *[Mac77, MacFre85, MohHen86]*, but the practical computation of $MGR_3(\mathcal{A})$ must gain great advantage from using parallel techniques, since the computation over different subnetworks may be spread over different processors in sundry clever ways. We shall show that the computation of $MGR_3(\mathcal{A})$ is similar to matrix multiplication, by representing a constraint network with n nodes as a matrix. Thus parallel techniques for matrix multiplication, a fairly well-understood domain, may be used for the computation of $MGR_3(\mathcal{A})$, yielding, one would imagine, good speedup over serial algorithms. Our matrix representation is different from that of *[Mac77]* who uses adjacency matrices to represent relations, which can only be done for finite domains. In particular, the size of the adjacency matrices is that of the domain of solution. In our representation, the size of a matrix is the size of the constraint network it represents, and the size of the domain of solution plays no part.

It should be noted that, in the presence of a suitable class of primitive (*atomic*) relations, namely a class forming a partition of the universe of pairs, the contrary

of each relation exists, and therefore arbitrary quantifier-free formulas over such a domain may be represented as disjoint unions (disjunctions) of CSPs. Solution or consistency algorithms may theoretically be run in parallel on such unions, and so the problem of solving arbitrary quantifier-free formulas is reduced to that of solving BCSPs. In practice, it might be that the representation of an arbitrary quantifier-free formula as a union of BCSPs itself requires significant computational resources, since it is roughly equivalent to putting a formula in disjunctive normal form, which in general is computationally infeasible. Most domains of interest have appropriate classes of primitives. It follows from our characterisation of constraint networks that if there is no suitable class of primitives then the standard tools for solving BCSPs aren't applicable, although some of our techniques are.

There is one important class of BCSPs which has appeared in the literature on temporal reasoning, namely the Interval Calculus of Allen *[All83, VilKau86, Sch86, Val87]*. The domain of BCSPs in this framework is that of intervals (pairs of ordered unequal points) over a dense, unbounded linear order *[LadMad88.1]*. The relations in *[All83]* form an algebra which we call the *Interval Algebra* **IA**. The **IA** contains as a subalgebra the *Point Algebra* **PA**. These algebras have the uncommon property that they both have only one countable representation as algebras of binary relations (that is, sets of pairs of elements) *[LadMad88.1]*. In the case of **PA**, that representation is as the algebra of $<$, $>$, and $=$ over the rational numbers. For **IA**, it is the algebra of intervals over the rational numbers with the 13 basic relations, and all unions of these, defined in *[vBen83, All83]*. (We shall call these 13 basic relations *atomic*, for reasons which we introduce later).

Why be concerned with the particular classes of relations that form **IA** and **PA**? For a specific answer, we show below that any CSP over an arbitrary finite or countable linear order may be solved by using the **PA**, and a CSP over intervals on a finite or countable linear order which has as labels sums of the thirteen basic relations (i.e. labels representing relations which are nonzero in the **IA**) may be solved by using the **IA** alone. This obviates the need to use the weaker formulation of Allen and Hayes *[AllHay85, AllHay87, AllHay88, Ladkin87.3]*, for quantifier-free constraint satisfaction for this (finite) class of relations. (The differences appear with quantified formulae, and with relations that are zero in the **IA** but nonzero in some other interval structure). For a more general answer, the Interval Calculus has more general application than temporal reasoning, since it is basically a formulation of one-dimensional interval reasoning. Reasoning with intervals in higher dimensions allows one to formulate constraint satisfaction problems in the domain of user interfaces (optimal placement of windows on a screen over time), and robot motion (three dimensional rectangular constraint spaces), and to solve such problems in general by lifting algorithms from the simpler one-dimensional cases. From the point of view of the general study of constraint satisfaction problems, the categoricity (that is, having only one countable model up to isomorphism) of the relation algebras

involved allows us to prove specific results about satisfiability, and use algorithms which do not work in the general case. By studying these cases, we can show how to extend these properties and algorithms to a large class of BCSPs. We are currently working on higher-dimensional analogues to the **IA**, and it appears that many results will extend to these algebras.

Allen *[All83]* proposed a heuristic consistency algorithm, which ran in low polynomial time to check a network for path-consistency *[Mon74, Mac77]*. As he noted (by an example due to Kautz), path consistency of an arbitrary BCN is insufficient to determine satisfiability in his calculus. We show that there are natural subalgebras of the Interval Algebra, for example the Containment Algebra, which just considers the relations of containment, disjointness and overlapping, in which even a path-consistent network with atomic relations (i.e. not sums of relations) on the edges is not necessarily satisfiable. The algorithms suggested for constraint satisfaction in this context thus cannot always be used for that purpose, since they rely on the satisfiability of atomically-labelled BCNs. We show that there is a large class of relation algebras (including the **IA**) for which a large class of path-consistent BCNs (including the ones with only atomic labels) is satisfiable. This immediately implies that consistency-checking is only at most serial cubic, parallel log time (see below) for BCNs in this class (significantly extending results in *[VilKau86]*. We present a technique for constructing new algebras from old, such that algorithms for satisfying BCNs in the appropriate class over these new algebras can be derived from an algorithm for the same task in the old algebra.

Organisation of the Paper

We comment next on various ways in which binary relations can be represented. Then we introduce the Interval Calculus, since we shall use it and its subalgebras as a source of examples while we introduce the algebraic approach to BCNs. Section 2 introduces binary constraint networks. In section 3, we formulate binary constraint networks algebraically, introduce the main concepts and algorithms used for consistency checking. Section 4 introduces two classes of examples for which path-consistency is hard. The first class is used to illustrate the technique for example-generation, and shows parallel linear complexity. The second class uses the same approach but is more complicated, and shows parallel quadratic complexity for path-consistency, for reduction-type algorithms. So with the estimate for the Matrix Reduction Algorithm from the previous section, we have parallel $O(n^2 \log n)$ upper bounds for path-consistency and parallel $O(n^2)$ lower bounds for reduction-type algorithms, providing fairly tight complexity bounds for these algorithms. Section 5 focusses on the Containment Algebra, in which not all atomically labelled BCNs are satisfiable, and the **PA**, in which they are. Section 6 shows that CSPs whose labels are sums of the thirteen basic relations over intervals on arbitrary finite or countable

linear orders may be solved using just the **IA**. Section 7 introduces a new algebraic construction, with examples of how **IA** and the *Point-Interval Algebra* **PIA** arise from **PA** by this construction, and includes theorems concerning BCNs over the *pointisable* relations, which include the atomic relations, and shows that consistency checking over these is serial cubic, parallel log time at worst.

Binary Relations

The study of binary relations and their logic is classical *[DeM64, Pei92, Sch95, Tar41]*. The early work of De Morgan, Peirce, Schröder was formulated by Tarski in the context of an algebraic theory he termed *relation algebra [op. cit.]*. In this work, we shall formulate constraint satisfaction problems involving binary relations in terms of relation algebra. We should consider first what ways there are available to represent binary relations.

A binary relation is usually defined as a set of pairs of objects from some domain, so that saying that two objects are in the relation is the same thing as saying that the pair formed by those objects is in the set. We shall be considering sets of binary relations in this paper, i.e. a set of sets of pairs from some domain. In other work on constraint satisfaction problems the binary relations are often presented explicitly, as enumerated sets of pairs. This can only be done, of course, if the relations are finite (and relatively small!). Many of the constructions and algorithms can only work on finite relations, e.g. the n-consistency algorithms in *[Fre78]*, and the representation of relations as adjacency matrices in *[Mac77]*. However, there many are interesting and important cases, such as reasoning with time intervals *[All83]* where the relations are infinite.

Another way of presenting binary relations, which works for infinite relations also, is to present a first order language which includes binary relation symbols, and give some axioms which are supposed to determine the properties of the relations interpreting the symbols. This approach was taken in *[AllHay88]* for the relations on intervals from *[vBen83, All83]*, although as we note later, the theory is not quite the same. In the axiomatic approach, a binary relation will be chosen to interpret each binary relation symbol in each model, and there may be many models of the axioms. Each model will contain a collection of relations interpreting the symbols.

We advocate another approach new to this area *[LadMad88.1]*, which is to present a *relation algebra*. This is similar in approach to presenting a first-order theory for relations, but different in essential respects. The objects in a relation algebra are intended to be binary relations themselves, and the algebra has operators on relations, such as sum (intended to be the set-theoretic union of relations considered as sets of pairs), product (intersection), converse and composition (defined later). So one may present a relation algebra, and enquire what collections of binary relations actually have the structure defined by that particular algebra, and again there may be many,

or, more surprisingly, none. An algebra which has interpretations as actual collections of relations is said to be *representable*. We often consider algebras which arise from actual collections of relations, so *a fortiori* they are representable, since the collections we start from are representations by definition. However, there are important algebras which are not known to be representable, e.g. the algebra generated from the single relation m (for *meets*), satisfying the equations that are equivalent to the Allen-Hayes axiomatisation *[AllHay88]*. This algebra is infinite *[LadMad88.2]*. The algebraic techniques for reducing BCNs to simpler, path-consistent ones with smaller labels still work for this algebra, and the algebraic consequences thus hold for intervals over all unbounded linear orders, even though the algebra itself is not known to have a representation. We refer to *[Jon82, LadMad88.1]* for a further exposition of relation algebraic concepts, and to *[JonTar52, Mad82, TarGiv88]* for deeper treatments of this subject.

Of these three approaches to dealing with binary relations, *[All83]* employed the first approach in essence, and in the next section we shall define the sets of pairs that were considered in that paper. *[AllHay88]* takes the second approach, using axioms in a first-order logical language to define properties of the binary relations studied in *[All83]*, although as the first author has shown *[Lad87.3]*, there are more models of the Allen-Hayes axioms than of the Allen Calculus. (The models of the Allen Calculus are shown in *[LadMad88.1]* to be intervals over a dense unbounded linear order, and the Allen-Hayes axioms do not imply density of the underlying order.) The third approach was used in *[LadMad88.1]* to characterise the structure of the relations in *[All83]*. The intent of the Allen-Hayes work was to axiomatise the theory that results from interpreting the transitivity table (i.e. the table for relation composition) from *[All83]*, not as equalities (i.e. the table entry *is equal to* the composition of the relations), but as implications (the table entry *contains* the composition of the relations), and this may have been the original intent of the table *[All86]*, although the intervals over the reals in fact satisfy the stronger condition of equality for table entries. Allen and Hayes show that the weaker (inequality) interpretation of the table is provable from their axioms *[AllHay88]*.

We consider all three approaches. Actual binary relations for us will be sets of pairs. We define *binary constraint networks* (BCNs) by formulas in a first-order logical language with relation symbols, or from an algebra of relations (which may be generated from the logical theory). All that is known about the relations, then, in a BCN, is inferred from the algebraic information concerning them. However, we shall show that this is sufficient for most of the algorithms used in solving constraint satisfaction problems over the class of relations under consideration.

In contrast, we define a *binary constraint satisfaction problem* (BCSP) to be a binary constraint network over a specific collection of actual binary relations. We shall argue that the relations, and those derived relations generated in the course of solving the

BCSP, form a relation algebra, and thus we can perform most of the operations needed to solve the BCSP purely algebraically, and only pass to the actual relations at the final stage when actual values are needed for the variables. The BCSP thus is equivalent to solving for a BCN over a specific representation of the algebra. The advantage of taking the more general algebraic approach is that by solving a BCN, one has thereby performed most of the steps involved in solving a particular BCSP, but the results are valid for all BCSPs arising from that BCN, i.e. all representations of the BCN. For example, results from BCNs over the Allen-Hayes algebra would be valid for CSPs over arbitrary unbounded linear orders.

We formulate relation algebras from the relations in a given domain. These algebras will often be finite, and when they are, by a theorem in *[LadMad88.1]* they will correspond to a finitely axiomatised first-order theory which has as models precisely the representations of the algebra as collections of actual binary relations. So the algebraic and logical approaches cohere in this way for BCNs that generate a finite algebra of relations. The algebraic approach has a clear advantage when dealing with problems which are not known either to be over finite domains or to be representable, e.g. the algebra arising from the Allen-Hayes axiomatisation. The algebraic algorithms of constraint satisfaction still apply in large part.

In most formulations of constraint satisfaction problems, the relations are given explicitly as collections of pairs, and the satisfaction algorithms (e.g. those in *[Fre78]*) operate on those pairs explicitly. However, there are constraint satisfaction problems that are formulated for infinite relations (i.e. relations which contain infinitely many pairs), such as those for reasoning with intervals in *[All83]*, for which these finitary explicit representations of relations do not exist. Furthermore, one may be in a situation where there is incomplete information concerning the relations, even if they are finite. We regard it as an advantage of our approach that most of the algorithms involved in solving BCSPs do not require explicit extensions of the relations to be given, and thus work equally well for finite or infinite relations, and in situations where there is incomplete knowledge. (It is necessary to distinguish between an infinite relation, and an infinite algebra of relations. For example, all of Allen's relations are infinite in extent, but they generate a finite algebra of relations, namely one with 2^{13} relations in it). A collection of relations that generates a finite relation algebra is highly desirable for solving BCNs. It matters less whether the relations themselves are finite or infinite.

The Interval Calculus

We introduce the interval calculus, as defined in *[All83]*. Intervals are pairs $\langle a, b\rangle$ of real numbers, with $a < b$. Such pairs are natural models of non-trivial convex intervals of real numbers, so we simply refer to such pairs as *convex intervals*. We

consider binary relations between convex intervals which are definable using the natural ordering of the reals. There are 13 basic, or atomic, relations which are obtained by considered the precise ordering of the endpoints of two intervals (including equality of endpoints), which we note below. Each of these basic relations is disjoint from the others, and every pair of intervals is in precisely one of these relations. Binary relations are usually considered as sets of pairs, and so there are 2^{13} relations definable from the 13 basics. We obtain the 2^{13} relations by taking all possible set-theoretic unions of the basics, and these are precisely the relations on intervals first-order definable from the underlying order on the endpoints, which follows from the quantifier-elimination property for the natural ordering on the real numbers *[ChaKei73]*.

In *[LadMad88.1]* it was shown that exactly the same structure results from considering pairs of rational numbers, rather than real numbers, in the sense that the two structures are indistinguishable either algebraically or by means of first-order logic. It was also shown that the rational intervals are the only countable model (up to isomorphism) of the theory of intervals in *[All83]* (although not of the inequivalent theory of Allen and Hayes *[AllHay88]*).

The 13 primitive relations may be presented pictorially in the following way. In the pictures, we consider the linearly ordered set to be implicitly represented along the horizontal, and we draw a finite line segment to indicate an interval on this set. There is no metric assumed on the set, so relative position of multiple intervals is the only observable quality.

A convex interval **d** looks like this (with time going from left to right):

d

The relations are:

- i equals j, strict identity of intervals

  i

  j

- i meets j, which means i is before j, but there is no gap (non-empty interval) separating them

  i

  j

- i precedes j, which means i is before j, and there **is** a non-empty interval separating them

  i ———————

  j ———————

- i overlaps j, meaning i starts before j, and j ends after i

  i ———————

  j ———————

- i starts j, intuitively i has the same starting point as j, but is otherwise strictly contained in j (ends strictly earlier than j)

  i ———————

  j ———————

- i ends j, intuitively i has the same ending point as j, but is otherwise strictly contained in j (starts strictly later than j)

  i ———————

  j ———————

- i during j, intuitively i has a later starting point than j, and an earlier ending point

  i ———————

  j ———————

and there are six others; namely

- the *converse* relations to all these (except for **equals**, which is self-converse). Given a binary relation R, the *converse* relation $R^\smile$ is defined by the condition $aR^\smile b \Leftrightarrow bRa$

All the relations above are disjoint, and they are also disjoint from their converses, i.e. no pair of intervals is in more than one of these relations. Furthermore, these relations form a *partition* of the universal relation, i.e. every pair of intervals is in

exactly one of these relations. The notion of a partition is crucial - a partition of the universal relation on any domain forms the set of atoms of a relation algebra *[JonTar52, Mad82]* under certain conditions *[LadMad88.1]* (see later for definitions).

The intuition behind the pictures may be formalised in a standard way if we consider a linearly ordered set, whose objects we call 'points', and construct intervals out of ordered pairs of distinct, ordered 'endpoints'. We define the intervals and relations between them, over a given linear order $<$ with domain T. The collection of *intervals over* T is $\{\langle a, b\rangle \ : \ a, b \ \in T \ \& \ a < b\}$.

We abbreviate the names of the relations, to just the first letter, for the purposes of the following definitions, except for *equality* which we denote $Id(T)$ to conform with subsquent usage. We also use $<$-chains to represent the conjunction of formulas in $<$ and $=$, i.e. $x < y = x' < y'$ means $x < y \ \wedge \ y = x' \ \wedge \ x' < y'$. For further considerations of readability, we write a formula such as $(x, y, x', y' \in T) \ \wedge \ (x = x' < y = y')$ as $x = x' < y = y' \in T$, i.e. the variables x, y, x', y' are all to take values in T, such that $x = x' < y = y'$.

$$Id(T) = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x = x' < y = y' \in T\}$$

$$P = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x < y < x' < y' \in T\}$$

$$D = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x' < x < y < y' \in T\}$$

$$O = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x < x' < y < y' \in T\}$$

$$M = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x < y = x' < y' \in T\}$$

$$S = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x = x' < y < y' \in T\}$$

$$F = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x' < x < y = y' \in T\}$$

These relations along with their converses are all the binary relations on ordered pairs of ordered distinct points of T that can be specified by giving the ordering of all four coordinates with respect to $<$ and $=$ in T. Any binary relation between intervals definable by quantifier-free formulas in terms of the ordering on T may be defined from these by taking the appropriate disjunction of conditions. In the case

where T is the rational numbers or the real numbers, more is true - we noted earlier that any binary relation on intervals *first-order definable* (i.e. using quantifiers also) in terms of the ordering on T is a sum of these basic relations. This follows from the quantifier-elimination property for the first-order theory of dense unbounded linear ordering *[ChaKei73]*, and helps to explain some of the power of the approach in *[All83]*.

We define the class of *primary relations over* T, $PR(T)$, to be the class of relations that are sums of the thirteen basic relations above. It turns out that for the case where T is a dense unbounded linear ordering (e.g. the rationals or the reals), the algebra of relations generated by the thirteen atomic relations is precisely $PR(T)$ *[LadMad88.1]*.

Acknowledgements

The Containment Algebra was developed by the first author and Pat Hayes, to whom many thanks for many hours of discussion on the topics in this paper. Thanks to Dick King for discussions on the asymptotic parallel time estimates, and to Richard Jüllig for the observation that the $MGR_3(\mathcal{A})$ is a fixpoint.

2 Binary Constraint Networks

We now introduce the constraint networks with which we shall be concerned. We use a general approach, but illustrate it with examples from the temporal reasoning networks introduced in *[All83]*.

A *binary constraint network* is an abstract representation of a logic specification of the form

$$i_1(R_{(1,1)} \vee R_{(1,2)} \vee ...)j_1 \wedge i_2(R_{(2,1)} \vee R_{(2,2)} \vee ...)j_2 \wedge .....i_n(R_{(n,1)} \vee R_{(n,2)} \vee ...)j_n$$

where the i_k, j_k are variables, and the $R_{(p,q)}$ are symbols denoting specific binary relations. The formula is thus a conjunction, for different variables, of constraints between pairs of variables. Each constraint between two variables is a disjunct of possibilities, for example

$$i_1(R_{(1,1)} \vee R_{(1,2)} \vee ... \vee R_{(1,k)})j_1$$

asserts that one of the relations denoted by $R_{(1,1)}, R_{(1,2)}, ..., R_{(1,k)}$ holds between i_1 and j_1. In the case where these relations generate a finite algebra of relations, we may define a BCN as such a formula. This definition suffices for most of the cases with which we shall be concerned.

In the case where the algebra of relations is not finite, then we have to use a more general algebraic definition of a BCN, as arising over the algebra of relations. For the general case, we shall define a BCN as a specification of the form

$$i_1\, r_1\, j_1 \ \wedge\ i_2\, r_2\, j_2 \ \wedge\ .....i_n\, r_n\, j_n$$

where $r_1, ..., r_n$ are elements of a *relation algebra [Jon82, JonTar52].* The elements of a relation algebra are intended to be binary relations, and thus for example $i_1 r_1 j_1$ is to be read as asserting that the variables i_1, j_1 should take values that are in the relation r_1 to each other. The algebraic operations on relations in the algebra are those of intersection, union, and complement of relations, along with the converse operator, which given a relation returns the converse relation, and the composition operator, which given two relations returns the composition of those relations (in the order given - it's not a commutative operator). Additionally, the empty relation and the universal relation are distinguished constants. Relation algebras were used to interpret a particular class of constraint satisfaction problems, namely those arising from temporal intervals, in *[LadMad88.1].*

When a particular algebra of relations is finite, there is a finitely axiomatised first-order logical theory that corresponds to the algebra in the sense that any representation of the algebra as a class of actual binary relations that satisfy the operation tables of the algebra is a model of the theory, and vice versa *[LadMad88.1].* Hence we may define a BCN by the first-order formula above, in this case. In the case that the algebra of relations is not finite, then even path-consistency algorithms (see next section) may fail to terminate, and thus the computational devices available to solve constraint satisfaction problems over this class of relations may become unusable.

A binary constraint satisfaction problem, or *BCSP*, is defined from a BCN when the actual relations (that is, the sets of pairs), are given that interpret the relation symbols in the BCN. These relations may be given extensionally, in the case that the relations are finite, as in classical constraint satisfaction problems such as n-queens, or may be specified in some other way. For example, in the case of interval temporal reasoning, the domain of the relations is the intervals over the real numbers (it was shown in *[LadMad88.1]* that the intervals over the rational numbers would also be appropriate, and indeed that this was the only appropriate countable domain over which a temporal interval BCSP was interpretable). The relations over this domain are given, as above, by logical formulas specifying which pairs are in a particular relation. The logical theory describing these relations is complete, by definition (it's the set of all sentences true in the domain), and a finite axiomatisation of this theory was given in *[LadMad88.1].*

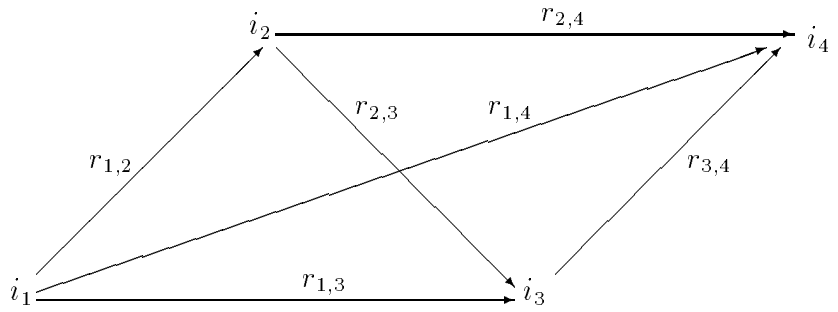
We can interpret these issues using the concept of a logical theory and a model of this theory *[ChaKei73].* The logical theory gives a series of axioms defining the relations, as for example was done for a modification of Allen's interval calculus

in *[AllHay85, AllHay87, AllHay88]*. This logical theory will in general have many (non-isomorphic) models, and if the theory is adequately axiomatised, we will in general get just the class of models we were aiming for, as was shown for the Allen-Hayes theory in *[Lad87.3]*. (Of course, all this may not always be possible, as Gödel showed for formal arithmetic). So we can consider the theory as defining the relations we're interested in, but given this, we further have to pick the model, the domain, over which we want to look for solutions. E.g. for the interval calculus of *[All83]*, we may look for solutions over the rational intervals or the real intervals, and if we find such, we can find other solutions in the domain of any linear order with enough points (see later). So a BCSP corresponds to attempting to solve a BCN over a particular domain. A BCSP corresponds to a model of a BCN.

Although the definition of a BCN has been given in terms of binary constraints only, unary constraints on variables also fit in the scheme, in the following manner. A unary constraint may be represented as a binary relation on the same variable, e.g. $i_k R i_k$ *[Mon74]*. In a domain of interpretation (i.e. for a BCSP corresponding to the BCN), such a binary relation R is a subrelation of the identity, i.e. the relation consists of some, but not all, pairs $\langle x, x \rangle$ from the domain. In many of the BCSPs that we consider, in particular the temporal interval BCSP from *[All83]*, the identity relation has no non-empty subrelations, so this situation does not arise. We shall have more to say later about this.

*BCN*s may be defined for any class of binary relations. The terminology *BCN* arises from representing the *BCN* as a graph in the following way. There is a node for each interval variable, and an edge between every pair of variables for which there is a relation asserted in the formula. The edge is labelled with the relation (which may be the disjunction of other, smaller, relations in the case of an edge between distinct variables) that is asserted by the formula. There may be loops corresponding to unary constraints, labelled with the corresponding binary relation as interpreted above. Figure 1 represents a BCN in four variables, with six relations asserted between the variables. The variables are $i_1, ..., i_4$, and the relation between i_k and i_j is $r_{k,j}$. The nodes of the graph represent the variables, and the oriented edges are labelled with the relations asserted between the tail variable and the head variable. We shall have more to say in the next section about these graphical representations. We shall show later that these kinds of networks may be defined over an arbitrary relation algebra, not necessarily one that has an actual representation as an algebra of binary relations (indeed there are some relation algebras that don't). The techniques we employ for manipulating and reducing networks also apply to networks over these *nonrepresentable* algebras.

In many BCSPs, there are certain distinguished relations, which we call primitive or atomic relations (the concept of an *atomic* relation will be defined explicitly later) which form the strongest constraints (i.e. they are the smallest relations, given by



$$i_1\, r_{1,2}\, i_2 \;\wedge\; i_1\, r_{1,3}\, i_3 \;\wedge\; i_1\, r_{1,4}\, i_4 \;\wedge\; i_2\, r_{2,3}\, i_3 \;\wedge\; i_2\, r_{2,4}\, i_4 \;\wedge\; i_3\, r_{3,4}\, i_4$$

Figure 1: A BCN in 4 variables and its graph

the strongest logical conditions). If there are such, then they will generally be the relations $R_{(i,k)}$ in the BCSP above, and all of the relations between the variables will be disjunctions, or equivalently set-theoretic unions or relation sums, of these primitive relations. In the case of interval temporal reasoning, we have given the primitive relations above. All BCSPs which are associated with BCNs over a finite algebra of relations have such a class of primitive relations associated with them, though they may not be explicit. This follows from general facts about finite relation algebras (in fact, from finite Boolean algebra). We shall indicate later how to obtain such a class of primitives for an arbitrary BCSP, but we also caution that the smallest such class may not have the desirable properties one needs to solve a given BCSP, and one might have to find a finer classification in general. This happens for certain BCSPs in the temporal interval calculus, as we shall show. It may also turn out that the process of searching for such primitives may not terminate, in which case the algebra of relations generated by the BCSP is infinite, and none of the standard algorithms (see below) are likely to terminate either.

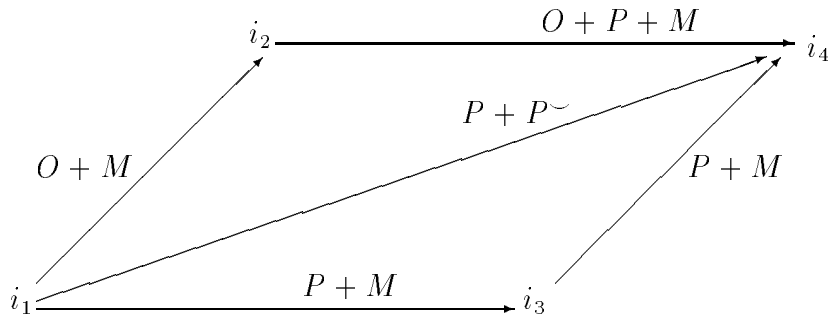
We have noted that the BCN and the corresponding formula may in general be interpreted over many domains. In the case of the Interval Calculus, such a domain is an arbitrary dense, unbounded linear order such as the rational numbers or the real numbers *[LadMad88.1]*. We shall assume a fixed, but indeterminate, domain, for much of what we do, since many comments apply equally to all the different possible domains of interpretation, although for examples and other purposes we shall sometimes specialise to the rational numbers, or the intervals over the rational

numbers.

Let us consider the case where there are primitive relations in the sense given above, of which all other relations in the domain, except for the empty relation, are unions or sums of these primitive relations. We shall also assume (we may in fact do so without loss of generality) that the relations form a partition of the domain (as do the primitive temporal interval relations). Suppose a given *BCN* over the domain is satisfiable, that is, that there are domain values for the variables for which the formula is true. Then it must be the case that, for these values, exactly one of the primitive relations in each disjunction must hold between these values, because the primitive relations are pairwise disjoint (no pair of intervals is in more than one of these relations). So if the *BCN* A is satisfiable, there must be another *BCN* A' with the same nodes and edges, but whose labels are single primitive relations, not disjunctions, such that all intervals which satisfy A' also satisfy A. For if A is satisfiable, pick a collection of intervals which satisfy A. The actual relations between such intervals (one primitive relation asserted between each) form such an A', and clearly any other intervals which satisfy A' also satisfy A, since the relations between intervals asserted by A' are all subrelations of the corresponding relation asserted by A, and thus a fortiori the A-relation must hold between intervals that satisfy the corresponding A'-relation. We shall call a *BCN* A' with labels that are atoms an *atomic BCN* or *ABCN*. We shall call a *BCN* A' a *reduction* of a *BCN* A if it contains the same variables, and the constraint on each pair of variables in A' is logically stronger than the corresponding constraint in A. By the phrase '*logically stronger than*' we mean '*subrelation of* ', or in the algebraic framework *less-than* in the standard Boolean partial ordering.

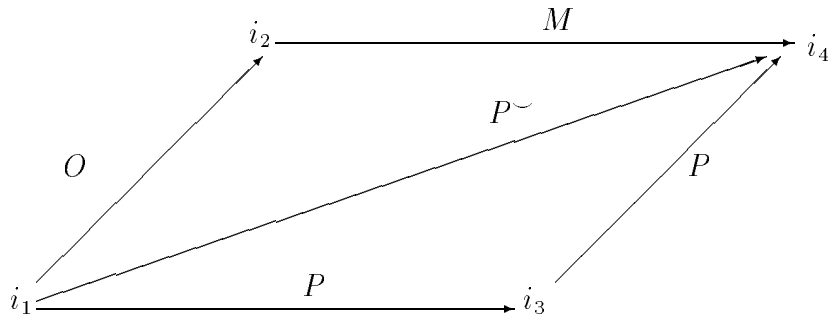
Figure 2 represents a BCN in the Interval Calculus *[All83]*. For certain pairs of variables a constraint is asserted which consists of a relation that is a union of basic relations. Union relations are represented here by sums (which is the appropriate algebraic notation, see below). Figure 4 is a reduction of the BCN in figure 3. Each relational constraint is a subrelation of the constraint in figure 3. Notice that the reduction is unsatisfiable, since the constraints between i_1 and i_3, and between i_3 and i_4 are both precedes, which implies that i_1 should precede i_4, by the composition table *[All83]* (this should be clear intuitively from the pictorial representation of the interval relations given above). However, the BCN asserts that i_4 should precede i_1 instead, which contradicts the inference above. Hence the BCN in figure 3 is unsatisfiable. Figure 4 shows a satisfiable reduction of the BCN in figure 2. Figure 5 shows a satisfying assignment of rational intervals to the variables.

So one technique for constraint satisfaction is to attempt to find an *ABCN* A' that is a reduction of the given *BCN* A. However, we want more than this. We want both *BCN*s to be satisfiable, or be able to detect that they are not. It has been suggested for temporal interval reasoning in *[All83]* that testing a *BCN* for



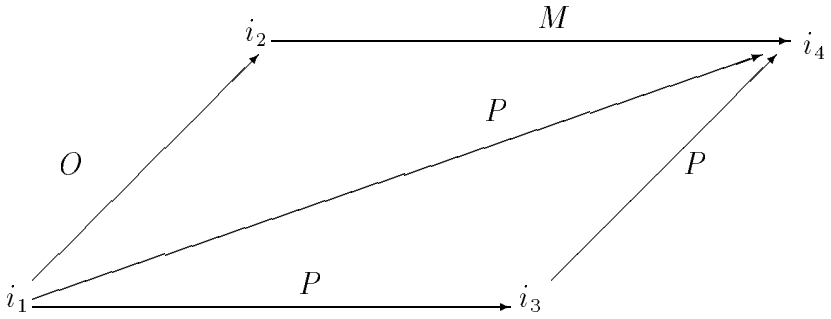
$$i_1\,(O+M)\,i_2 \;\wedge\; i_1\,(P+M)\,i_3 \;\wedge\; i_1\,(P+P^{\smile})\,i_4 \;\wedge\; i_2\,(O+P+M)\,i_4 \;\wedge\; i_3\,(P+M)\,i_4$$

Figure 2: An Interval BCN in 4 variables and its graph



$$i_1\,O\,i_2 \;\wedge\; i_1\,P\,i_3 \;\wedge\; i_1\,P^{\smile}\,i_4 \;\wedge\; i_2\,M\,i_4 \;\wedge\; i_3\,P\,i_4$$

Figure 3: An inconsistent reduction



$$i_1\, O\, i_2 \;\wedge\; i_1\, P\, i_3 \;\wedge\; i_1\, P\, i_4 \;\wedge\; i_2\, M\, i_4 \;\wedge\; i_3\, P\, i_4$$

Figure 4: A consistent reduction

$$\begin{aligned} i_1 &\longleftarrow \langle 0, 2\rangle \\ i_2 &\longleftarrow \langle 1, 5\rangle \\ i_3 &\longleftarrow \langle 3, 4\rangle \\ i_4 &\longleftarrow \langle 5, 6\rangle \end{aligned}$$

Figure 5: A satisfying assignment

path-consistency (see below), and converting a *BCN* into a path-consistent *BCN* is a reasonable heuristic for satisfiability. However, it was recognised (by an example due to Kautz) that there are path-consistent *BCN*s that cannot be reduced to path-consistent *ABCN*s in this domain and therefore are unsatisfiable. However, there are algorithms (e.g. *[Sch86]*, a sketch in *[Val87]*, and others exist in the relation algebra folk literature) that will reduce a path-consistent *BCN* to a path-consistent *ABCN* if possible, or fail. These algorithms in general look infeasible (i.e. NP-hard). The heuristic consistency algorithm, which just checks path consistency, is at most serial cubic time, parallel log time, in the number of nodes. However, one really wants not just to check path-consistency, but to obtain a path-consistent *reduction* of the BCN. This is serial cubic, parallel $n^2 \log n$ time (see below).

Suppose one has a path-consistent *ABCN*. Is this satisfiable? Not necessarily, for example in certain natural subalgebras of the Interval Algebra, e.g. the Containment Algebra, there are unsatisfiable path-consistent *ABCN*s. Likewise, there are certain subalgebras for which all path-consistent ABCNs are satisfiable, indeed in the full Interval Algebra, there is a class of BCNs which we call *pointisable* for which path-consistent networks are satisfiable. The pointisable BCNs have labels from the *pointisable relations*, a class of 187 relations from the **IA** listed in the appendix. This class includes all the atomic relations, and so in particular ABCNs over the **IA** are pointisable.

The proof provides an algorithm by means of which one may obtain a satisfying assignment to the variables of a path-consistent *ABCN*, by using a simpler algorithm for a particular subalgebra, in certain cases. For the Interval Algebra, this subalgebra is the Point Algebra **PA** *[LadMad88.1]*, and using the mappings from *[Lad88.1]* for Boolean formulas to convert a path-consistent *ABCN* in **IA** into an equivalent path-consistent *ABCN* in **PA**, and pulling the satisfying assignment back to **IA**.

3 The Algebra of Binary Relations

In order to state and solve BCSPs, there are certain operations on binary relations that we need to perform. In this section, we enumerate these operations, and observe that if we close a class of binary relations under these operations, then we have a *relation algebra* in the sense of Tarski *[JonTar52]*. In turn, we shall benefit from looking at BCSPs as problems in the algebra of binary relations.

We have observed that the binary constraint satisfaction problem may be stated in general for any class of binary relations. The statement of a BCSP involves the *sum* of relations, (defined by disjunction of relation formulas, or by the set-theoretic union of relations), so it is reasonable to require that the class is closed under sums. Every sum, and thus every label, will be a single, but not necessarily primitive, relation in the constraint set. Further, to allow the path-consistency

algorithm to proceed, one needs to use the *product* (logical conjunction, or set-theoretic intersection) of relations. Constraint graphs may be completed (that is, every pair of nodes is connected by an edge) if the *universal* relation (containing all possible pairs in the domain) is available to be used as the constraint along the edges not otherwise constrained. This complete graph is equivalent to the original in the sense that the same values for the variables will satisfy either. Similarly, one can detect inconsistency in a constraint network by inferring that the *empty* relation (the relation with no pairs) must hold between two nodes. One may further detect that two nodes are constrained to have the same value, and that therefore the two nodes may be collapsed into one node, with corresponding constraints calculated from the edges connecting either node to others, if one has the *equality* relation (aka the identity relation) as a relation in the class. Further, there is a theorem due to Montanari *[Mon74]* that if a *BCN* is satisfiable, then the identity relation is contained in the composition of relations along the edges of any loop in the *BCN*. If a relation R holds between nodes i and j, then it follows that the *converse* relation $R^\smile$ must hold between j and i, by definition. The converse $R^\smile$ of R is defined by the condition $aR^\smile b \Leftrightarrow bRa$. *Converse* is thus a unary operator on binary relations $^\smile : R \mapsto R^\smile$. Converse is also used in algorithms for checking path-consistency, which must iterate over every directed triangle in the network, so sometimes the directed edge from i to j, (which we shall denote $[i,j]$) and the constraint R must be considered, and in other instances the edge $[j,i]$ and thus the constraint $R^\smile$. Additionally, the consistency algorithms use the *composition* of relations, again by a result of Montanari *[Mon74]*. Given binary relations $R(x,y)$, $S(x,y)$, the composition $R \circ S$ of the relations has the first-order definition

$$R \circ S(x,y) \Leftrightarrow (\exists z)(R(x,z)\ \&\ S(z,y))$$

This has the meaning that $(a,b) \in (R \circ S)$ if and only if there is a value c such that $(a,c) \in R$ and $(c,b) \in S$. See the figure.

We conclude that the class of relations referred to in a *BCN*, either explicitly or implicitly in the algorithms, is a class closed under sum, product, converse, and composition, and includes the universal relation over the domain, the empty relation, and the equality relation for the domain as distinguished relations.

Thus the class of relations in a binary constraint satisfaction problem form a *relation algebra* in the sense of Tarski *[JonTar52]*, which is precisely a collection of relations closed under the above operations, with the distinguished relations as above. A *relation algebra* could be thought of as a Boolean algebra of relations, with sum, product, the universal and empty relations (as in set algebras) with the extra operations of converse and composition, and the extra distinguished relation of equality. (A certain finite set of equational axioms for relation algebras was chosen by Tarski, and it turns out that there are algebras which cannot be represented as algebras of binary relations in the manner indicated, that nevertheless satisfy

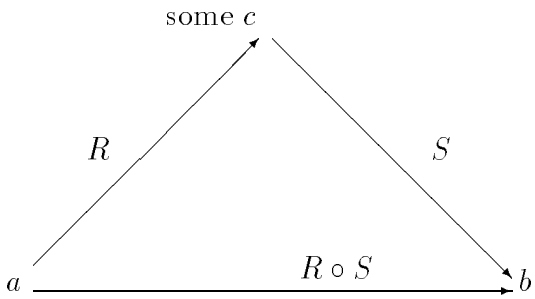


Figure 6: The composition of relations

the axioms *[Lyn50]*. Technically, therefore, a relation algebra is defined to be any member of this larger class, representable or not). This is treated in greater detail in *[LadMad88.1]*. We shall use a different notation for the operations in a relation algebra, from that used for the operations on actual binary relations, so as to make it clear when we refer to algebraic operations, and when to actual binary relations. The notation we use is standard in the relation algebraic literature. Thus we use $+$ for sum, $\cdot$ for product, a superscript suffix $^-$ for complement, 1 and 0 for the universal and empty relations respectively, $^\smile$ for converse, the semicolon ';' for relation composition, and $1'$ for the identity relation. Examples of relation algebraic expressions are $r + s, r \cdot s, r^\smile, (r\ ;\ s)$.

One feature of treating the class of relations as a relation algebra, is that for many BCSPs there is a class of primitive relations, in the sense used earlier, even though they may not be explicit, and there is an algebraic algorithm which finds such a collection (although this algorithm does not terminate if there is no such). This algorithm provides a reasonable heuristic for finding such a collection if used in the following way. A relation algebra is a Boolean algebra, and any Boolean algebra generated from a finite collection of sets (relations in this case) is finite (namely, the variety of Boolean algebras is locally finite, see *[BurSan78]*). Any finite Boolean algebra has a collection of minimal non-zero elements (i.e. smallest, non-empty relations in our case), such that every member of the algebra is a unique sum (union) of these minimal elements. Such a minimal element is called an *atom*. Relation algebras, however, are not locally finite, so in certain cases this algorithm will not yield a collection of atoms of the relation algebra with the desired properties, since the relation algebra will be infinite, and therefore no finite collection of atoms can partition the algebra. So given a BCSP, we may attempt to generate the relation algebra by closing under the relational operators, and by using the Boolean reduct we

may obtain a first approximation to the atom structure that will aid us in solving the BCSP. We can find the atoms by taking arbitrary products of the relations and selecting out the smallest results. All this is standard algebra. In the temporal interval reasoning case in *[All83]*, the atoms are precisely the thirteen relations enumerated explicitly.

Each of the primitive relations is a smallest relation in the algebra, in the sense that there is no other non-empty relation contained in, or intersecting, each primitive. We note from *[Lyn50]* that it is sufficient to define composition and converse only on atoms, to determine the values of these operations on all relations in the algebra, since both distribute over sum, or union. The collection of atoms of a finite relation algebra, whose maximal element is the universal relation, partition the universal relation. This means that every pair belongs to precisely one of the atoms. For example the 13 primitive relations in the temporal constraint case partition the universal relation. Each pair of intervals belongs to precisely one of the 13 primitive relations.

For a finite relation algebra, one can define the *contrary* operator (negation, or set-theoretic complement) on relations, since every relation R is a sum of atoms, and therefore its contrary is the sum of precisely those primitives that are *not* included in the sum for R.

We note that there is a natural partial ordering on the relation algebra, corresponding to the property of one relation being a subrelation of another. This ordering coincides with the standard partial ordering $\leq$ defined on the Boolean part of the algebra. Thus the atoms are the smallest nonzero elements in this partial order. We shall refer to one relation as being *below* another, to mean that it is a proper subrelation of the other.

The Algebraic Interpretation of BCNs

It is standard (since the nineteenth century work of De Morgan, Peirce and Schröder *[op. cit.]*) that a collection of binary relations between objects may be represented by a labelled digraph in which the nodes represent the objects and the edges are labelled with the relation holding between the objects. In fact, such a representation may be used for general relation algebras, including those which have no representation as a collection of binary relations over a domain of objects (it may seem counterintuitive that there are such, but see *[Lyn50, Jon82]*).

Thus we can represent a *BCN* as a graph, in which the nodes represent variables to which values are to be assigned, and the edges are labelled with single elements from a relation algebra. There are some relation algebras that do not arise from collections of binary relations over some underlying domain (i.e. *nonrepresentable* relation algebras) and we note in passing that one may define BCNs over nonrepresentable algebras as well *[Mad83]*. Indeed such BCNs share many of the

same properties as BCNs from representable algebras, e.g. the concepts of arc- and path-consistency, and one may ask about reducibilites to ABCNs.

For interval temporal reasoning, the relation algebra could be the Allen algebra defined from the 13 Allen primitives *[All83]*, as defined in *[LadMad88.1]*. Since each element in the algebra is a sum of atoms, a general label will be a sum of atomic relations, as envisaged in *[All83]*. See *[Mon74, Mac77, DecPea88]* for definitions and concepts concerning *BCN*s that we use here. For convenience, we shall consider the nodes of a *BCN* as labelled with the integers $1...n$, rather than the variables $x_1...x_n$. We shall also refer to the arc, or edge, between i and j, as the *edge [i,j]*. We let $R_{i,j}$ be the label on $[i,j]$. If x is in the relation R to y, we shall express this either as $R(x,y)$ or as xRy, whichever is more convenient.

There are three concepts of consistencies in *BCSP*s that we shall particularly need, namely *domain consistency*, *arc consistency* and *path consistency*, The first two conditions are also called 1-consistency, 2-consistency and the third is equivalent to 3-consistency by a result of Montanari *[Mon74, Mac77, Fre78]*. We formulate the consistency concepts for a BCSP, and in each case obtain an algebraic formulation which then can be interpreted as a constraint on the BCN itself, implying the same condition for any BCSP interpreting that BCN. Thus we can consider the consistency concepts purely algebraically, as properties of classes of BCNs.

Domain Consistency A *BCSP* is *domain consistent* iff, for any node, there is some possible value that that the variable at that node can take. Thus a unary constraint at a variable may affect domain consistency if there are no values that satisfy the constraint. As we have noted, we can interpret a unary constraint on the i'th variable in a BCN as follows. At the node i there is an edge $[i,i]$, and a binary relation $R_{i,i}$ labelling that edge. Corresponding to that relation is the domain $D_i = \{x : \langle x,x\rangle \in R_{i,i}\}$. D_i is the domain of values which the variable at node i is constrained to range over range over. Then $R_{i,i} = \{\langle x,x\rangle : x \in D_i\}$ is a binary relation that is below the identity relation, specifically the restriction of the identity relation to D_i. A relation algebra is said to be *integral*, iff the identity is an atom, thus has no relations below it except for the empty relation. Thus in an integral algebra of relations, there are no non-trivial domain constraints one can impose in this algebra, and thus every *BCN* in this algebra is domain consistent. The constraint imposed by domain consistency can be expressed by the condition $R_{i,i} \neq \emptyset$, and the algebraic interpretation of domain consistency is thus the condition $R_{i,i} \neq 0$, so we shall call a BCN domain consistent if it satisfies this condition. The **PA**, the **IA** and the **CA**, indeed most of the examples we shall consider, are integral.

Arc Consistency: A *BCSP* is arc consistent iff, for any edge $[i,j]$, and any value a_i for the node i, there is a value a_j for the node j such that $R_{i,j}(a_i,a_j)$. Arc consistency may be formulated as follows. A value for the node i is given by any

value for $R_{i,i}$, and similarly for the node j. So the arc consistency condition is that for any pair of nodes i, j, $R_{i,i} \subseteq R_{i,j} \circ R_{j,i}$, and the algebraic formulation of this condition is $R_{i,i} \leq (R_{i,j} \ ; \ R_{j,i})$, We call a BCN arc consistent if it satisfies this condition. Integrality of the algebra also implies arc consistency, because $R_{i,i} = 1'$, so the condition reduces to $1' \leq R_{i,j} \circ R_{j,i}$ and $R_{j,i} = (R_{i,j})^{\smile}$, and $1' \leq x \circ x^{\smile}$ holds in all integral relation algebras.

Path Consistency: A *BCSP* is path-consistent iff, for any pair of values a_i and a_j for the nodes i and j, such that $R_{i,j}(a_i, a_j)$, and any path $(i, k_1, k_2, ..., k_m, j)$ in the associated *BCN* between i and j, there is a sequence of values $b_1, ..., b_m$ for $k_1, ..., k_m$ such that $R_{i,k1}(a_1, b_1)$ and $R_{k1,k2}(b_1, b_2)$ and ..... and $R_{km,j}(b_m, a_j)$. Not all temporal CSPs are path-consistent, by any means! A BCSP is *3-consistent* iff for any pair of values a_i and a_j for the nodes i and j, such that $R_{i,j}(a_i, a_j)$, and any path (i, k, j) in the *BCN* between i and j, there is a value b for k such that $R_{i,k}(a_i, b)$ and $R_{k,j}(b, a_j)$. To formulate these conditions in an algebraic framework, notice that the definition of 3-consistency involves a form similar to the definition of relation composition given earlier. Namely, checking whether there is a value b for k such that $R_{i,k}(a_i, b)$ and $R_{k,j}(b, a_j)$ is simply to check whether $(a_i, a_j) \in R_{i,k} \circ R_{k,j}$. The definition of 3-consistency then requires this check be performed for every pair $(a_i, a_j) \in R_{i,j}$. This is simply to ask whether $R_{i,j}$ is a subrelation of $R_{i,k} \circ R_{k,j}$, i.e. set-theoretically whether $R_{i,j} \subseteq R_{i,k} \circ R_{k,j}$. Thus 3-consistency may be formulated algebraically as the condition $R_{i,j} \leq (R_{i,k} \ ; \ R_{k,j})$ for the BCN. It is well-known that for general constraint networks, a network is path-consistent if and only if it is 3-consistent (see the lemma below). Arc-consistency is a special case of 3-consistency with $i = j$.

Note that the algebraic formulations of domain-, arc- and path-consistency are valid for general BCNs, whether the algebras from which the BCN labels are taken are representable or not.

Lemma 1 (Montanari *[Mon74]*:) *A BCN is path-consistent iff, for any pair of values a_i and a_j for the nodes i and j, such that $R_{i,j}(a_i, a_j)$, and any path (i, k, j) of length 2 in the BCN between i and j, there is a value b for k such that $R_{i,k}(a_i, b)$ and $R_{k,j}(b, a_j)$.*

This leads to the following algorithm for checking path-consistency of a network.

Path Consistency Checking Algorithm: Given a *BCN A*: For every triangle (i, k, j) in A, check $R_{i,j} \leq R_{i,k} \ ; \ R_{k,j}$.

Procedures for checking path consistency are versions of this algorithm, with more or less attention spent on clever ways to perform the iteration *[Mon74, Mac77,*

Fre78, All83, VilKau86]. The composition table is used in the path-consistency checking algorithm. Composition distributes over relation sum (set union of relations), so it is sufficient (although maybe not computationally optimal) to use the composition table for atoms alone, as in *[All83].* For non-atoms which are the sum of n atoms and m atoms, it requires $n \times m$ looks-up in the composition table to calculate the composition.

An algorithm for checking path-consistency may also incorporate an algorithm for putting a *BCN* in path-consistent form.

Path Consistency Algorithm: Given a *BCN* A, Iterate until no more changes: For every triangle (i,k,j) in A: do $R_{i,j} \leftarrow R_{i,j} \cdot (R_{i,k} \ ; \ R_{k,j})$.

Note that the occurrence of the operation $\cdot$ (set-theoretic intersection of relations) in this algorithm is an argument for including product of relations in the operations needed for constraint satisfaction. The algorithm checks every triangle to see whether the label on the direct edge is below the composition of the labels on the 2-path. If not, it replaces the direct label with the intersection of the composition with the direct label. This relation is strictly below the composition (otherwise the triangle would already be o.k.), and thus the triangle is o.k. However, some later assignment to a different triangle which shares one of the edges on the 2-path may change the label on this edge, necessitating an iteration of the operation on the triangle. Clearly the *BCN* is path-consistent when and only when the fixed point is reached and no more changes need be made.

Since all of the operations performed in the path-consistency algorithm are in the algebra, the algorithm always runs to completion whenever the relation algebra is finite. There is one distinguished class of path-consistent networks which cannot ever be satisfiable in any BCN, namely those with 0 on every edge. The algorithm will propagate a 0 on a single edge throughout the network, to obtain a member of this class. This happens when, for some i,j,k, $R_{i,j} \cap (R_{i,k} \circ R_{k,j}) = \emptyset$ (or $R_{i,j} \cdot (R_{i,k} \ ; \ R_{k,j}) = 0$ when expressed algebraically). In this case, the constraint on the edge $[i,j]$ cannot be satisfied, because there is no pair of values that is both in the constraint on $[i,j]$ and also in the induced constraint $(R_{i,k} \ ; \ R_{k,j})$ from the 2-path. Thus the BCSP corresponding to the network is unsatisfiable, and we may terminate the path-consistency algorithm when this happens, since we are usually interested not in the path-consistency per se, but in using it as a filter for satisfiability. If a network has all zero labels, we call it a *zero network*, and if a network has no 0 labels we call it a *nonzero network*.

Revised Path-Consistency Algorithm: Given a *BCN* A, Iterate until no more changes, or until 0 is assigned to an edge (whence return **unsatisfiable**): for every triangle (i,k,j) in A, $R_{i,j} \leftarrow R_{i,j} \cdot (R_{i,k} \ ; \ R_{k,j})$.

Lemma 2 *If A has a path-consistent reduction, then the path-consistency algorithm will terminate with a path-consistent reduction. Further, every path-consistent reduction is itself a reduction of the output of the algorithm.*

Proof: Let the output of the algorithm be the $BCN\ B$. Suppose a path-consistent reduction C of A is not a reduction of B. Let the label on edge $[i,j]$ in C be $C_{i,j}$. Consider the first edge $[i,j]$ which is assigned a label $R_{i,j}$ which is lower than the corresponding label in C, namely $C_{i,j}$ which must happen by assumption. Then it is assigned this label because of some triangle inconsistency in a triangle (i,k,j). By assumption, at the time this assignment is made, the labels $R_{i,k}, R_{k,j}$ are larger than or equal to the corresponding labels on C, and so is the previous label R' on $[i,j]$. So $R_{i,j} = R' \cdot (R_{i,k}\ ;\ R_{k,j})$. But $C_{i,j} \leq (C_{i,k}\ ;\ C_{k,j})$, by path-consistency, and so $C_{i,j} \leq C_{i,j} \cdot (C_{i,k}\ ;\ C_{k,j}) \leq R' \cdot (R_{i,k}\ ;\ R_{k,j})$, since both $\cdot$ and ; are monotonic with respect to $\leq$. Hence $C_{i,j} \leq R_{i,j}$. Contradiction. Hence there is no such network C, and the lemma is proved.
End of Proof.

From the lemma, we may define the *Most General path-consistent Reduction of* $\mathcal{A}$, $MGR_3(\mathcal{A})$[1], to be the BCN output by this algorithm, and of which all other path-consistent reductions of $\mathcal{A}$ are also reductions. $MGR_3(\mathcal{A})$ may also be characterised as the greatest fixpoint solution of the following equation scheme, where the relations $r_{i,j}$ are original, and the equations are to be solved for the $r'_{i,j}$.

$$r'_{m,n} \leq r_{m,n}$$

$$r'_{m,n} = r'_{m,n} \cdot \prod_s (r'_{m,s}\ ;\ r'_{s,n})$$

(Someone, somewhere, ought to deduce something clever from this).

The computation of $MGR_3(\mathcal{A})$ may be improved even further by using the following observation. Consider a triangular network as in the figure. Without any information concerning the relations denoted by R, S and T, we may reduce this network to a path-consistent network, which we call the *CT-reduction* of the triangle. The path-consistency of the CT-reduction follows from a theorem of relation algebra due to Chin and Tarski *[ChiTar51]*, namely that the following equation is satisfied in every relation algebra.

$$x \cdot y\ ;\ z\ \leq\ (y \cdot x\ ;z^{\smile})\ ;(z \cdot y^{\smile}\ ;x)$$

Thus the algorithm could also be modified to use the CT-reduction of a triangle at each step, instead of modifying edges one at a time, e.g.

[1]The subscript 3 is for 3-consistent. we suppose there are analogous notions of $MGR_k(\mathcal{A})$, for $k > 3$.

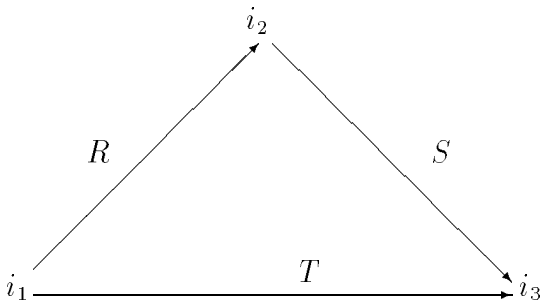


Figure 7: A triangular network

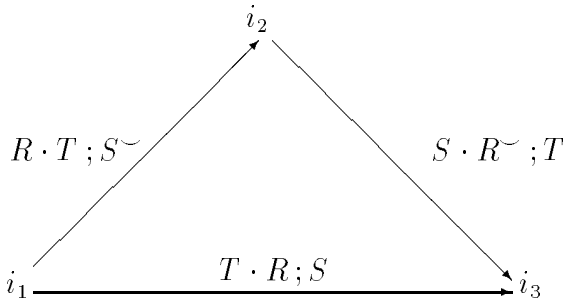


Figure 8: The CT-reduction

Path-Consistency Algorithm II: Given a *BCN* A, Iterate until no more changes, or until 0 is assigned to an edge (whence return **unsatisfiable**): replace every triangle (i,k,j) in A, with its CT-reduction.

The algorithm computes $MGR_3(\mathcal{A})$ as before. Intuitively, this algorithm could converge faster than the previous, (although asymptotically they are the same). However, in parallel, there is always a tradeoff between waiting for more computation from another processor, and using its partial, or old, results without waiting *[Ada88]*, and so we cannot assert categorically that it is an improvement.

For the **IA**, we have the following lemma.

Lemma 3 *Every nonzero BCN over* **PA**, *or* **IA** *is both domain- and arc-consistent.*

Proof: We have noted that domain- and arc- consistency are guaranteed by the fact that both algebras are integral.
End of Proof.

These conditions are not necessarily true for every BCSP based on intervals. For example, the formulation of Allen and Hayes *[AllHay88]* allows models that are intervals over arbitrary unbounded linear orders which are not necessarily dense. Suppose, in such the order S, that density is not satisfied. Then there are points $x < y$ such that there is no z in between, i.e. no z such that $x < z < y$. Thus, for the interval $\langle x, y\rangle$, there will be no interval that starts it, finishes it, or is during it, i.e. if $i = \langle x, y\rangle$, there is no j such that iSj, iFj, or iDj. Thus a BCSP over the intervals over S which is domain consistent may very well not be arc consistent, if one of these labels is the sole label on a well- (or badly-) chosen edge. We shall show later, however, that using BCNs over the **IA** allows one to satisfy constraints over any countable linear order (or finite order with an appropriate number of elements), provided that the labels are sums of interval algebra atoms, i.e. labels from the class $PR(T)$.

Every *BCN* over **IA** is both domain- and arc-consistent, but it may nevertheless be true that one can narrow the search space for a satisfying assignment by pruning out those values from consideration at a node that cannot partake in a satisfying assignment for that node. So one may be tempted to run algorithms such as in *[Fre78]* for reasons of attempted efficiency. The following lemma shows that this doesn't help at all. The lemma is stated for the rational intervals, but holds for any dense, unbounded linear order, as the proof shows.

Lemma 4 *Suppose A is a satisfiable BCSP over the rational intervals, or the real intervals. For each node i, let D_i be the set of values of i that actually occur in some satisfying assignment to A. Then D_i contains every interval.*

Proof: Any collection of intervals over a dense unbounded linear order satisfies the condition of homogeneity, that is that if a sequence $a_1, a_2, ...a_n$ of elements satisfies a formula $\phi(x_1, x_2, ..., x_n)$ in the theory of dense linear order, then any other sequence $b_1, b_2, ...b_n$ which has the same relative ordering as $a_1, a_2, ...a_n$ also satisfies $\phi(x_1, x_2, ..., x_n)$. (By relative ordering, we mean that for all $i, j \leq n$, $a_i < a_j$ iff $b_i < b_j$, and $a_i = a_j$ iff $b_i = b_j$.) Hence, using the mappings in *[Lad88.1]* between the theory of intervals over an unbounded dense linear order, and the theory of unbounded dense linear order, which preserve satisfiability, we can see that the theory of intervals over an unbounded dense linear order is also homogeneous in the following sense. Let S be an unbounded dense linear order. Let f be a strict order-preserving map $S \rightarrow S$, i.e. an S-endomorphism. For an interval $i = \langle a, b \rangle$, let $f`i$ be the interval $\langle f(a), f(b) \rangle$. Then if a sequence $i_1, i_2, ....i_n$ of intervals over $S$ satisfies a formula $\psi(x_1, x_2, ..., x_n)$ in the theory of intervals over $S$, then $f`i_1, f`i_2, ....f`i_n$ also satisfies $\psi(x_1, x_2, ..., x_n)$. Note also that for any pair of intervals i, j over S, there is an S-endomorphism f such that $f`i = j$.

Suppose a *BCN* A, over the intervals over S, S a dense inbounded linear order, is satisfiable. Suppose A has n nodes, labelled $1, .., n$, and let the interval i be the value of node p in a satisfying assignment for A. Then, for any other interval j, there is a satisfying assignment for A in which the value of node p is j, since the constraint to be satisfied is just a (quantifier-free) formula ψ in the theory of intervals over S, as we noted in section 1 of this paper. Note that this statement is vacuously true if ψ is contradictory, i.e. A is unsatisfiable. Hence every temporal *BCN* is domain-consistent.

It is easy to see that for every primitive relation R, and every interval i, there is an interval j such that iRj *[LadMad88.1]*. This follows from the unboundedness and density properties of S (and furthermore is not true unless S is unbounded and dense, so for example is not true for intervals over the integers). Since every arc in A is labelled with a primitive or a sum of primitives, it follows that for every value for a node p (i.e. every interval, as we have seen) and every possible label R' for $[p, q]$, there is a value for q (namely some other interval, since q can take all values) for which $pR'q$.
End of Proof.

Hence, *BCN*s over dense unbounded linear orders such as the reals or the rationals, path-consistency is the lowest kind of consistency check that makes sense.

The next step in attempting to satisfy a *BCN* has been to attempt to reduce a path-consistent *BCN* to a path-consistent *ABCN*, a *BCN* with atomic labels on the edges *[Mac77, All83, Val87]*. One hopes that the path-consistency algorithm will narrow the problem practically, so that not much remains to be done by a search algorithm for a reduced *ABCN*. Existing algorithms have the following general form, although of course most of the effort is spent on reasonable search procedures.

	i_1	i_2	i_3	i_4
i_1	r_{11}	r_{12}	r_{13}	r_{14}
i_2	r_{21}	r_{22}	r_{23}	r_{24}
i_3	r_{31}	r_{32}	r_{33}	r_{34}
i_4	r_{41}	r_{42}	r_{43}	r_{44}

Figure 9: The matrix representation of the 4-node RAN

Reduction Algorithm: Given a path-consistent *BCN* A, Iterate until a path-consistent *ABCN* is found: For every edge $[i, j]$ in A, pick an atom R below $R_{i,j}$, to form a reduced *ABCN*, and check it for path-consistency.

Note that if an *ABCN* is not path-consistent, then it is unsatisfiable, for if all the $R_{i,j}$ are atoms, then if $R_{i,j} \leq (R_{i,k} \ ; \ R_{k,j})$ does not hold, then $R_{i,j} \cdot (R_{i,k} \ ; \ R_{k,j}) = 0$, since any atom is either below or disjoint from any non-empty relation in the algebra. We might wish the converse to hold, namely that any path-consistent *ABCN* is satisfiable. However, this is false in general, as we shall show in the next section, although it is true for a large class of algebras, including the full **IA**, as we shall show.

A Matrix Formulation

Relations in constraint satisfaction problems have been represented by matrices before, as in *[Mac77]*, where adjacency matrices are used to represent finite relations, and the various consistency algorithms, particularly for path-consistency, use matrix multiplication. We present another method of matrix representation of BCNs, where the dimension of the matrix is now the number of nodes of the BCN, and therefore this representation is available for BCNs over infinite relations, as well as BCNs over non-representable relation algebras. The reformulation of an BCN as a matrix is straightforward, but leads directly to observations on the parallel complexity of path-consistency, as well as providing a particularly elegant formulation of consistency properties. The formulation first appeared in *[Mad83]*.

We associate an $n \times n$ matrix M_A with an n-node BCN A in the following way. If $(i_j \ r_{jk} \ i_k)$ is asserted by the BCN, then the entry in row j column k of M_A is r_{jk}. Let $(M_A)_{jk}$ be the entry in M_A in row j column k. Then the definition of M_A can be expressed as

$$(M_A)_{jk} = r_{jk}$$

An example is shown in the figure. (By the convention that $r_{ji} = (r_{ij})^{\smile}$, only the upper- or lower-half of the matrix need be shown).

In order to represent a BCN as a matrix in this way, it is necessary that for every pair of variables i_j, i_k there is a relation r_{jk} asserted by the BCN. We use the convention that if there is no explicit relationship asserted by the BCN between i_j and i_k then $r_{jk} = 1$, which is easily seen to be correct. (We may regard all BCNs as complete graphs by the same device). Note that all BCN matrices are square, thus it makes sense to talk of the size n of an $n \times n$ BCN-matrix.

Let M, M' be two BCN-matrices of the same size. We define pointwise the operations $(M \cdot M')$ and $(M \,;\, M')$. The operation $\cdot$ is a pointwise operation. The operation ; is not.

$$(M \cdot M')_{ij} = M_{ij} \cdot (M')_{ij}$$
$$(M \,;\, M')_{ij} = \prod_{k \leq n} M_{ik} \,;\, (M')_{kj}$$

The definitions of these two operations resemble the definition of matrix sum and matrix product for matrices over a ring or field. We shall call them respectively the intersection and product of BCN-matrices. (Notice that this is overloading the term *product*, since the intersection of matrices is the pointwise *product* of its elements, in the relation algebra). It is immediate that intersection is an n^2 operation, and product an n^3 operation over the relation algebra from which the entries are drawn, from considering the same for matrices over rings. We are unable to use the Schönhage-Strassen asymptotic speedup directly, as this relies on the existence of inverses for the addition operation, in our case therefore the existence of inverses for intersection, and intersection has no inverse in general. However, matrix multiplication may be accomplished in $\log n$ time in parallel, so we may infer that product may be computed in parallel $\log n$ time.

We may now formulate consistency conditions in terms of matrices. Define the relations $\leq, <, >, \geq$ on matrices by their pointwise values, i.e. let rel be any of these four relations on the relation algebra elements. Then

$$M \; rel \; M' \Leftrightarrow (\forall i \forall j)(M_{ij} \; rel \; (M')_{ij})$$

Define also M^n by the usual induction,

$$M^1 = M,$$

$$M^{n+1} = (M^n) \,;\, M$$

So, for example, $M^2 = (M \,;\, M)$. Recall that path-consistency of a BCN is equivalent to the condition $(\forall i \forall j \forall k)(r_{ij} \leq (r_{ik} \,;\, r_{kj}))$. Let M be the matrix of the BCN. The condition becomes $(\forall i \forall j \forall k)(M_{ij} \leq (M_{ik} \,;\, M_{kj}))$. Fix i and j. The condition is $(\forall k)(M_{ij} \leq (M_{ik} \,;\, M_{kj})$, from which it follows directly that $M_{ij} \leq \prod_{k \leq n}(M_{ik} \,;\, M_{kj})$,

that is $M_{ij} \leq (M\ ;\ M)_{ij}$, i.e. $M_{ij} \leq (M^2)_{ij}$. Hence we may formulate path-consistency as the succinct condition $M \leq M^2$. We may also formulate the path-consistency algorithm in matrix terms, noting that the algorithm is equivalent to the first path-consistency algorithm we stated, with a particular ordering of the triangles. By analogy with the BCN case, we define the *Most General 3-consistent Reduction* of a matrix M, $MGR_3(M)$, to be the matrix of $MGR_3(\mathcal{A})$, where M corresponds to the BCN $\mathcal{A}$.

Matrix Path-Consistency Algorithm Check $M \leq M^2$.

Matrix Reduction Algorithm: Repeat $M \longleftarrow M \cdot M^2$ until $M \leq M^2$.

This algorithm computes $MGR_3(M)$. Matrix product is parallel $\log n$ time. Let the depth (i.e. the length of the longest chain in the natural ordering) of the relation algebra R from which the labels are drawn be a. It is easy to see that if R is finite, with a atoms, then the depth of R is a. Therefore the replacement of a particular element in M by a smaller one, in the path-consistency algorithm, can happen at most a times. There are n^2 elements in M, therefore $a.n^2$ is a crude upper bound on the number of iterations required in the algorithm. We obtain the following lemma.

Lemma 5 *(a) Path-consistency checking is parallel $\log n$ time. (b) Path-consistency (computing $MGR_3(M)$) is parallel $a.n^2 \log n$ time.*

The proof is immediate from the foregoing considerations.

The question arises whether the $a.n^2$ term in the matrix reduction algorithm estimate can be improved. We shall show that path-consistency is at least parallel $O(n^2)$.

4 The Complexity of Path-Consistency

We construct two classes of examples using a simple finite representable *symmetric* relation algebra $\mathcal{A}$, with four atoms. Both classes have examples of order n, for infinitely many n. The first class has examples for each n, and the second only for n a multiple of 5. We use the first class to illustrate the technique we use in constructing the second class of examples. The first class requires $O(n)$ (in fact, $n-1$) distinct states to approach path-consistency, and the progression of the computation is easily pictured. The second class requires $O(n^2)$ distinct states (in fact $(2n^2+2n-1) \div 5$) and provides the class of examples establishing the lower bound. It is also harder to picture, but, we hope, straightforward subsequent to understanding the principle by which the computation over the first class progresses. We call an algorithm

for $MGR_3(\mathcal{A})$ *reduction-type* if it operates by making subnetworks path-consistent iteratively. The path consistency algorithm schemes, and the Matrix Reduction Algorithm, are all reduction-type. All published algorithms (serial or parallel) are reduction-type (they all fit the schemes in the previous section). The classes of examples exhibit the property that there is information flow through the network, one triangle at a time being path-inconsistent, and there is only one pattern the information may flow through the network. The number of distinct states provide the bound estimate. Reduction-type algorithms fit the paradigm of producing distinct intermediate states, and therefore both parallel and serial reduction-type algorithms are subject to this lower bound.

Both examples illustrate the same kind of approach. We call a 3-node complete subnetwork of any network a *triangle*, and we say a subnetwork is *closed* if it is path-consistent (terminology due to Montanari *[Mon74]*). In each state of a computation, there is precisely one non-closed triangle, labelled the same way in each instance. During a path-consistency computation, this triangle will, in the course of being closed, propagate its character to a neighbor. Precisely one label is changed at each iteration of a closure technique. The network becomes closed only when all propagations have been accomplished. For the first class of examples, the propagation takes place along a row of $n-1$ edges, and in the second class, over a square of edges.

The Algebra $\mathcal{A}$

The atoms of $\mathcal{A}$ are $1'$, a, b, and c. Every atom is *symmetric*, i.e. $x^\smile = x$ for every atom x, and it follows easily that every element in the algebra must be symmetric (hence the appellation symmetric algebra). We define $0'$, the diversity element, to be the complement of $1'$, i.e. $(1')^-$ (we shall also use the symbol $\neq$ for $0'$, in the context of the **PA**, below). Thus $0' = a + b + c$ in A. We define relation composition on the atoms thus:

$$a \,;\, b = b \,;\, a = b \,;\, c = c \,;\, b = 0' = a + b + c$$

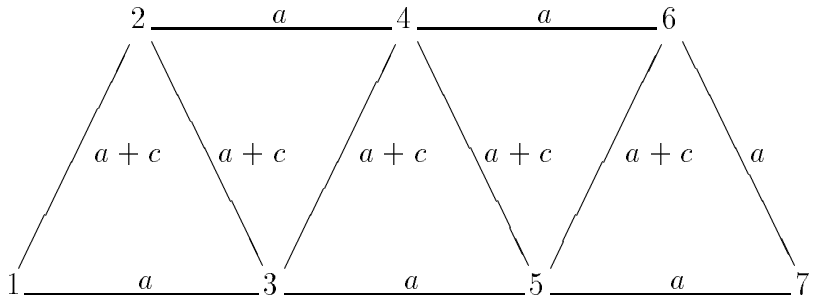
$$b \,;\, b = c \,;\, c = 1$$

$$a \,;\, c = c \,;\, a = b + c$$

$$a \,;\, a = c^- = 1' + a + b$$

Since converse and composition have been defined over the atoms, they are thus defined over the whole algebra, as we have mentioned before.

Notice that $b \leq \prod_{0 \neq x, y \leq 0'} (x \,;\, y)$. It is straightforward to show from this, in relation algebra, that $\mathcal{A}$ is representable over an infinite set. (This observation does not preclude a representation of A over a finite set as well).



All edges not shown are labelled with $(a + b)$.
The only nonclosed triangle is $\{5, 6, 7\}$.

Figure 10: The BCN corresponding to M_7

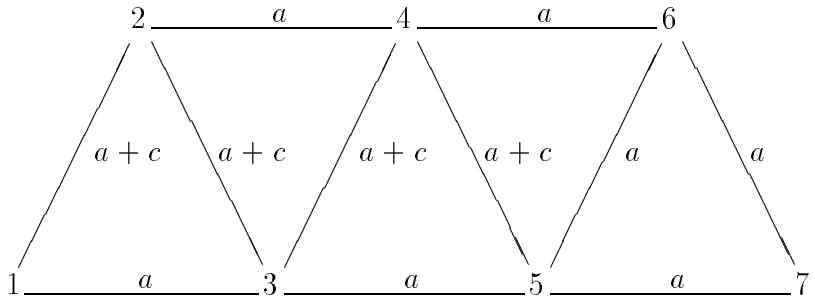
We shall need to know the following compositions for seeing what happens in both classes of examples. Firstly, $(a + c)\,;\,(a + c) \geq (c\,;\,c) = 1$, since composition commutes with $+$. Similarly, from the table $(a+c); a = a; (a+c) = 1$, $(a+b); (a+c) = (a + c)\,;\,(a + b) = 1$, and $(a + b)\,;\,(a + b) = 1$.

A Linear Time Class

We define a class of matrices M_n, for each odd positive integer n, and show that the reduction algorithm for M_n takes $n - 1$ iterations. For convenience, we shall omit the subscript when defining M_n. Notice that matrices over a symmetric algebra are symmetric matrices. Thus we may represent BCNs over a symmetric algebra by undirected graphs, since the labels read correctly in either direction.

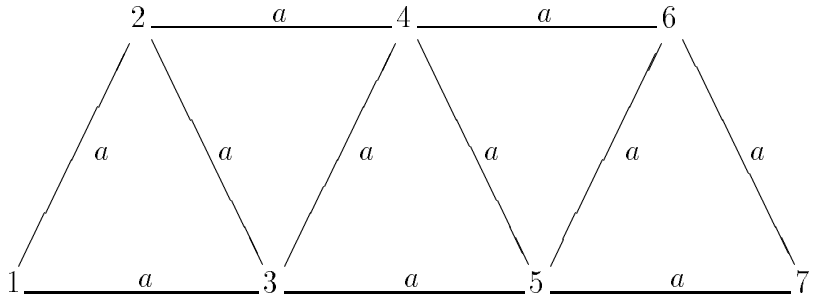
Let $n = 2m + 1$, $m \geq 1$. Set $M(k, k + 1) = a + c$ whenever $1 \leq k \leq 2m$, $M(2m, 2m + 1) = a$, $M(k, k + 2) = a$ whenever $1 \leq k \leq 2m - 1$, $M(i, i) = 1'$ whenever $1 \leq i \leq 2m + 1$, and $M(i, j) = a + b$ in all other cases. The network corresponding to $n = 7$ is shown in the figure.

It is easy to check that every triangle in M is closed with one exception, namely $2m - 1$, $2m$, and $2m + 1$. We have $M(2m - 1, 2m) = a + c$, but $M(2m - 1, 2m + 1)\,;\,M(2m + 1, 2m) = a\,;\,a = c^{-}$, so $M(2m - 1, 2m) \cdot M(2m - 1, 2m + 1)\,;\,M(2m + 1, 2m) = a \neq M(2m - 1, 2m)$. It follows that M^2 agrees with M except along arc $(2m - 1, 2m)$, which is now labelled a. Furthermore, M^2 has exactly one nonclosed triangle, namely, $2m - 2$, $2m - 1$, and $2m$, which is isomorphic to the nonclosed



All edges not shown are labelled with $(a + b)$.
One label has changed, on the edge $[5, 6]$.
The only nonclosed triangle is $\{4, 5, 6\}$.

Figure 11: The BCN after $\{5, 6, 7\}$ is closed.



All edges not shown are labelled with $(a + b)$.

Figure 12: The final reduced, path-consistent BCN

triangle in M. Hence the process repeats itself as the labels propagate along the diagonals from right to left in the figure. On the first iteration of the Matrix Reduction Algorithm, the label on the arc $(2m-1, 2m)$ changes from $a+c$ to a. On the next iteration, the label on the arc $(2m-2, 2m-1)$ changes from $a+c$ to a. On the next, $(2m-3, 2m-2)$, etc. On the kth iteration, $(2m-k, 2m-k+1)$ changes. So $(1,2)$ changes on iteration number $2m-1$, which is iteration number $n-1$.

This computation trace is followed by all reduction-type algorithms, as should be clear. Any clever way of iterating over triangles to obtain path-consistency must change the triangles in the order in which they are changed by the matrix reduction algorithm, since $n-1$ arcs need to change. Furthermore, they must change between distinct states. Each state of a path-consistency reduction algorithm has exactly one non-closed triangle, except for the final state. Furthermore, the states of such an algorithm must include each of the states enumerated above, since each new state is a consequence of closing a single non-closed triangle. There are $n-2$ such states. Our argument does not depend on whether the algorithms are serial or parallel, just whether they are reduction-type, and we would obtain a parallel linear lower bound for path-consistency. But we can do better than this, with a quadratic bound, next.

A Quadratic Time Class

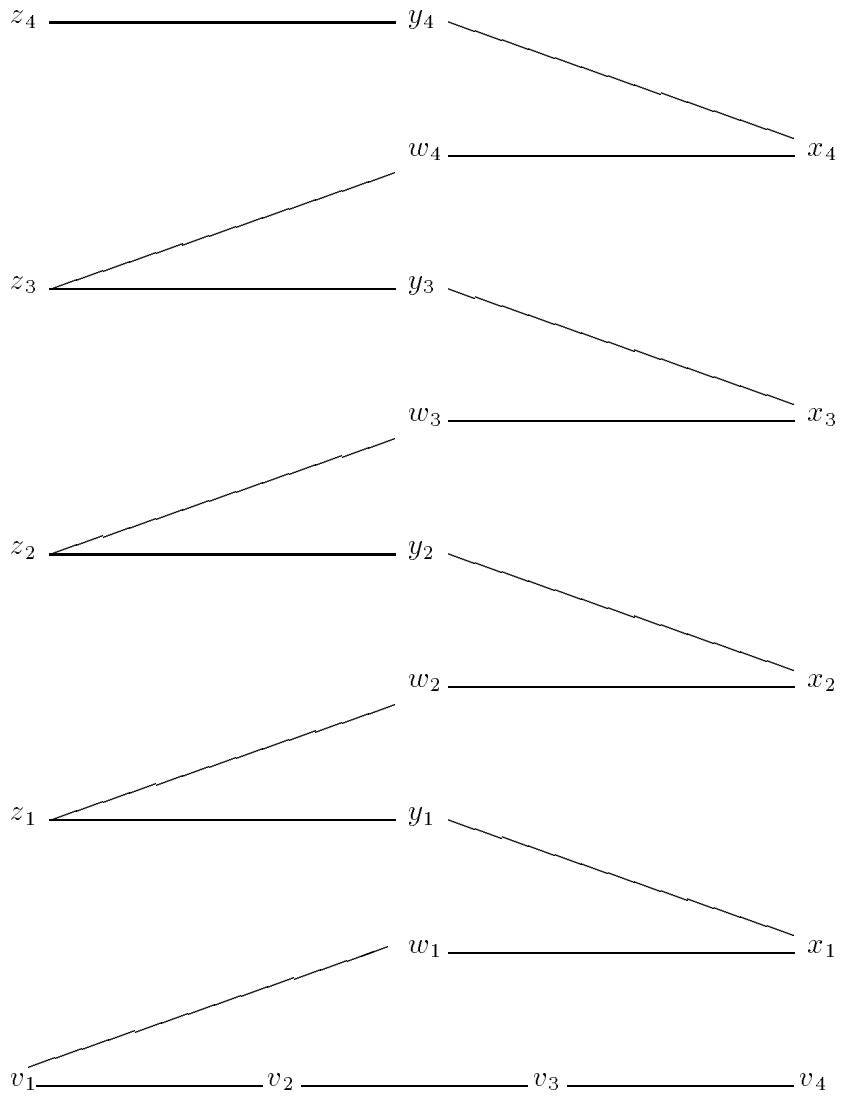
We describe a class of matrices N_k where k is a multiple of 5, say $k = 5m$. N_k is defined as follows. The nodes will fall into 5 types of m nodes each, namely $v_1, ..., v_n$, $w_1, ..., w_n$, $x_1, ..., x_n$, $y_1, ..., y_n$, $z_1, ..., z_n$. The entries in N_k are

$$N(v_1, w_1) = a$$

$$\begin{aligned}
N(v_i, v_{i+1}) &= a, \quad i = 1, ..., m-1 \\
N(w_i, x_i) &= a, \quad i = 1, ..., m \\
N(x_i, y_i) &= a, \quad i = 1, ..., m \\
N(y_i, z_i) &= a, \quad i = 1, ..., m \\
N(z_i, w_{i+1}) &= a, \quad i = 1, ..., m-1 \\
N(w_i, v_j) &= a+c, \quad i, j = 1, ..., m \\
N(y_i, v_j) &= a+c, \quad i, j = 1, ..., m \\
N(x_i, v_m) &= a+c, \quad i = 1, ..., m \\
N(z_i, v_1) &= a+c, \quad i = 1, ..., m
\end{aligned}$$

For all other distinct nodes s, t, $N(s,t) = a+b$, and for any node s, $N(s,s) = 1'$.

The similarity to the class M_n should be clear, however, the topology of the N_k requires elaboration. In the figure we illustrate a way to think of an arrangement



All the lines shown are labelled with a.
All the z_i are connected to v_1 (vertically), labelled $(a + c)$, not shown.
All the x_i are connected to v_4 (vertically), labelled $(a + c)$, not shown.
All the w_i and y_i are connected to all the v_j, labelled $(a + c)$, not shown.
All edges not shown are labelled with $(a + b)$.

Figure 13: The arrangement of N_k for $m = 4$

of the nodes that we hope aids comprehension of the example. Not all the edges are drawn. The horizontal edges shown, and the diagonal edge $[v_1, w_1]$, are labelled with a. The z_i are each connected to v_1 with an edge labelled $(a + c)$, as are the x_i to v_4, and the w_i and y_i to each of $v_1, ..., v_4$. Other edges, as before, are labelled with $(a+b)$. As before, edges change from labels $(a+c)$ to label a, one at a time per iteration, so that at each intermediate state there is just one nonclosed triangle, with two edges labelled a, and one edge labelled $(a+c)$. The seed triangle is $\{v_1, v_2, w_1\}$, and the edges change in the following order.

$$[w_1, v_2], ..., [w_1, v_4], [x_1, v_4],$$

$$[y_1, v_4], ..., [y_1, v_1], [z_1, v_1],$$

$$[w_2, v_1], ..., [w_2, v_4], [x_2, v_4],$$

$$[y_2, v_4], ..., [y_2, v_1], [z_2, v_1],$$

.....

In other words, the first w to the v's, left to right, then the first x to v_4. Then the first y to the v's, right to left, followed by the first z to v_1. And so on up the figure. In the general case, replace 4 by m, and the number of edge-label changes is $(2m(m + 1) - 1)$. Again, all edges must change, and each intermediate state has precisely one nonclosed triangle, and each nonclosed triangle propagates its characteristic labelling to a neighbor as it is closed. Thus there are $(2m(m+1)-1)$ intermediate states, so path consistency for this class of examples takes at least $(2m(m + 1) - 1)$ time, for reduction-type algorithms, parallel or serial.

The Complexity of Path-Consistency

Putting together the results above, we have

Theorem 1 *Path-consistency has an asymptotic lower bound of $O(n^2)$ with constant factor 0.4 for reduction-type algorithms, Thus path-consistency is between n^2 and parallel $n^2 \log n$ time for reduction-type algorithms.*

Sketch of Proof: The class N_k above provides the lower bound, and we have given an informal argument for that already. The constant factor estimate comes from the term $2m^2$, in the edge-change estimate, since $n = 5m$. The Matrix Reduction Algorithm provides the upper bound.
End of Proof.

Conjecture: We conjecture that the restriction to reduction-type algorithms is actually unnecessary, and that in fact this class of examples provides a lower bound for all path-consistency computations. This conjecture is fueled by the lack of algorithms for these problems that are not reduction-type. As we have mentioned, computing $MGR_3(\mathcal{A})$ is a greatest fixpoint computation, so the question really depends on algorithms for computing fixpoints.

5 Some Special Algebras

In this section we shall present an example of a temporal constraint satisfaction problem in an important subalgebra of the **IA**. The example consists of a path-consistent *ABCN* which is unsatisfiable. Because of the existence of such examples, the algorithms presented above for determining satisfiability do not succeed in general. For a particular class of constraint satisfaction problems over a given relation algebra, it may or may not be the case that path-consistent *ABCN*s are satisfiable.

The Containment Algebra[2]

We consider the following algebra of relations, a proper subalgebra of **IA**, which we call the *Containment Algebra*, **CA**. The five primitive relations of the **CA** are those of inclusion (In), containing ($In^\smile$), overlapping-but-not-contained (Lap), disjointness (Out) , and equality (Id). Let T be any dense unbounded linear order (e.g., the rationals, or the real numbers). As before, we write a formula $x, y, x', y' \in T \ \wedge \ x' \leq x < y \leq y'$ as $x' \leq x < y \leq y' \in T$ for readability. Note that **CA** is a subalgebra of **IA**, since **IA** is the algebra generated by the thirteen relations over an arbitrary dense unbounded linear order T *[LadMad88.1]*.

$$In = (S + F + D)$$

$$In^\smile = (S^\smile + F^\smile + D^\smile)$$

$$Out = (P + P^\smile + M + M^\smile)$$

$$Lap = (O + O^\smile)$$

$$Id(T)$$

These relations have the definitions given in the figure as sets of pairs over an arbitrary unbounded dense linear order T. The sum of the five is the universal relation, and the product of any pair of them is zero (this follows from the corresponding

[2]The CA is joint work with Pat Hayes, and presents a more natural occurrence of pathological behaviour than the original example subalgebra. Suffice it to say that these examples seem to be more common than not.

$$In = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x' \leq x < y \leq y' \in T\}$$

$$In^{\smile} = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x \leq x' < y' \leq y \in T\}$$

$$Lap = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x < x' < y < y' \vee x' < x < y' < y \in T\}$$

$$Out = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : y \leq x' \vee y' \leq x \in T\}$$

$$Id(T) = \{\langle\langle x, y\rangle, \langle x', y'\rangle\rangle : x = x' \wedge y = y' \in T\}$$

Figure 14: Definitions of the atoms of the Containment Algebra

facts for the **IA** primitives of which these are sums), and hence these five relations partition the universal relation. The five relations satisfy the further conditions that the converse of each is one of the five, and the composition of any two is a sum of some of the five, and thus these five relations are the atoms of a relation algebra (see *[Lyn50, LadMad88.1]*. This algebra is therefore a proper subalgebra of **IA**, since, of the five, only $Id(S)$ is an atom of **IA**. Note that Out and Lap are self-converse. The composition table for the atoms of **CA** may be calculated from the composition table for **IA** *[All83, LadMad88.1]* and is as follows, omitting the entry for $Id(S)$, which is algebraically both a left and a right identity (pun unavoidable) for the composition operation. The second author has recently shown that satisfiability in **CA**-networks is NP-hard. It follows from this that not all path-consistent **CA**-networks are satisfiable, since path-consistency checking is cubic time. One may further enquire whether there are path-consistent atomically labelled unsatisfiable networks (thus showing that the reduction approach to satisfiability fails for this algebra). We give an example with four nodes, which is the minimum (since any three-node network which is path-consistent is clearly satisfiable, from the equivalence of path-consistency with 3-consistency).

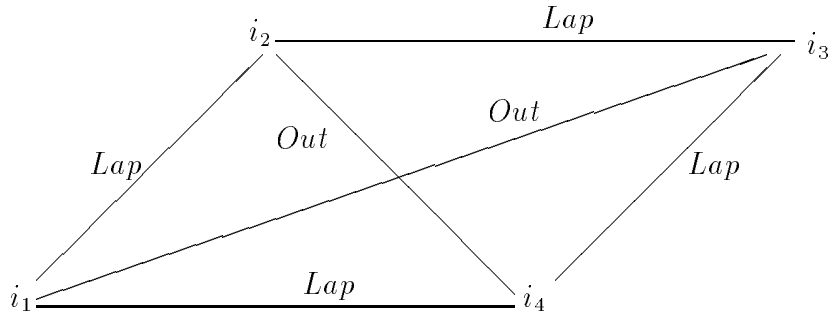
Open Problems: Find an algebra with a minimum number of atoms for which there are path-consistent unsatisfiable ABCNs. Ditto for BCNs. Find an algebra with a minimum number of atoms for which satisfiability is NP-hard.

Consider the *ABCN* on four nodes, in the figure. Let the nodes be i_1, i_2, i_3, i_4, and the labels on $[i_1, i_2], [i_1, i_4], [i_2, i_3], [i_3, i_4]$ are all Lap. The labels on the two other edges $[i_1, i_3]$ and $[i_2, i_4]$ are Out. Since Lap and Out are their own converses, it is unnecessary to orient the edges in the figure. Let us call the *ABCN* CE (for counterexample).

Lemma 6 *CE is a path-consistent, unsatisfiable ABCN in* **CA**.

$\circ$	In	$In^\smile$	Lap	Out
In	In	1	$Out + Lap + In$	Out
$In^\smile$	$In + In^\smile$ $+Lap + Id(S)$	$In^\smile$	$In^\smile + Lap$	$In^\smile + Lap$ $+Out$
Lap	$Lap + In$	$Out + Lap$ $+In^\smile$	1	$Out + Lap$ $+In^\smile$
Out	$Out + Lap$ $+In$	Out	$Out + Lap$ $+In$	1

Figure 15: The composition table for the atoms of the CA



$$i_1\ Lap\ i_2\ \wedge\ i_1\ Out\ i_3\ \wedge\ i_1\ Lap\ i_4\ \wedge\ i_2\ Lap\ i_3\ \wedge\ i_2\ Out\ i_4\ \wedge\ i_3\ Lap\ i_4$$

Figure 16: An unsatisfiable, path-consistent ABCN in CA

Proof: Observe first that both *Lap* and *Out* are symmetric relations, i.e. self-converse, so the *ABCN* may be considered to be a graph from the point of view of labelling. To check path-consistency in CE, it is sufficient to observe that $Out \leq Lap \circ Lap$, $Lap \leq Out \circ Lap$ and $Lap \leq Lap \circ Out$.

To show unsatisfiability, we argue by contradiction. Let $\langle l_n, r_n \rangle$ be the value for node i_n, for $n \leq 4$, in some satisfying assignment of intervals to the nodes. Then $\langle l_1, r_1 \rangle$ *Lap* $\langle l_2, r_2 \rangle$, so without loss of generality assume $\langle l_1, r_1 \rangle$ O $\langle l_2, r_2 \rangle$, i.e. $l_1 < l_2 < r_1 < r_2$. Assume now $\langle l_2, r_2 \rangle O \langle l_3, r_3 \rangle$ so $l_1 < l_2 < r_1 < r_2$ and $l_2 < l_3 < r_2 < r_3$. So $l_1 < l_3$ and hence $\langle l_1, r_1 \rangle (P + M) \langle l_3, r_3 \rangle$ from the constraint on $[1,3]$, so $r_1 \leq l_3$, and we have $l_1 < l_2 < r_1 \leq l_3 < r_2 < r_3$. We now enquire about the relation of $\langle l_4, r_4 \rangle$ to all this. Since $\langle l_1, r_1 \rangle (P + M) \langle l_3, r_3 \rangle$, the only relation consistent with this between $\langle l_1, r_1 \rangle$ and $\langle l_4, r_4 \rangle$ is $\langle l_1, r_1 \rangle O \langle l_4, r_4 \rangle$, and likewise $\langle l_4, r_4 \rangle O \langle l_3, r_3 \rangle$. So $\langle l_2, r_2 \rangle$ and $\langle l_4, r_4 \rangle$ have at least the subinterval $\langle max(l_2, l_4), min(r_2, r_4) \rangle$ in common (and this is indeed a proper interval), and yet the constraint between them asserts they are disjoint. Contradiction. Hence there is no satisfying assignment for CE.
End of Proof.

The Point Algebra

Let L be the natural relation $<$ on the rational numbers $\mathbf{Q}$. L generates a relation algebra, which we call the *Point Algebra* **PA**. It turns out that there is a subalgebra of **IA** isomorphic to **PA** *[LadMad88.1]*. We use the notation $<^Q, >^Q, \approx^Q$, etc, in the description of the relations in the **PA**. For ease of reading the tables below, we omit the superscripts, e.g. $<$ is $<^Q$. However, the reader should bear in mind that these are just abbreviations for these specific binary relations over the rationals, and not interpret the symbols as referring to a general order relation. An entry in row $<$, column $<$ of the composition table then represents the binary relation that is the composition $< \circ <$ over Q. In Q, this composition is $<$, and in fact this will only be the case in a relation algebra generated by a dense ordering. The transitivity law for orderings will always guarantee that $<^O \circ <^O \subseteq <^O$, in any algebra generated from a partial ordering over a domain O, but equality holds only in dense orderings, and for the relation algebra generated by some non-dense ordering $<^O$, $<^O \circ <^O$ will not be $<^O$, but some strictly smaller relation. Thus our use of $<$ for $<^Q$ might lead to confusion if it is forgotten that we are using it here just for readability. **PA** has three atoms, hence eight elements. Remember that $1'$ is the identity relation (equality) in the algebra. The elements are 0 , $<$, $>$, $1'$, $\leq$, $\geq$, $\neq$, 1. The atoms are $<$, $>$ $(= <^\smile)$ and $1'$. The tables for $+$, $^\smile$ and $\circ$ are given in the figure. The additions not shown in the figure follow from relation-algebraic laws, namely $0 + r = r + 0 = r$ and $1 + r = r + 1 = 1$, and the associative laws for addition. For ease of readability in the figure, we use the symbol $\approx$ instead of $1'$ for the equality relation in **PA**.

$+$	$\approx$	$<$	$>$
$\approx$	$\approx$	$\leq$	$\geq$
$<$	$\leq$	$<$	$\neq$
$>$	$\geq$	$\neq$	$>$

$\smile$	
$\approx$	$\approx$
$<$	$>$
$>$	$<$

$\circ$	$\approx$	$<$	$>$
$\approx$	$\approx$	$<$	$>$
$<$	$<$	$<$	1
$>$	$>$	1	$>$

Figure 17: The operation tables for atoms of **PA**

$\circ$	$<$	$>$	$\leq$	$\geq$	$\neq$
$<$	$<$	1	$<$	1	1
$>$	1	$>$	1	$>$	1
$\leq$	$<$	1	$\leq$	1	1
$\geq$	1	$>$	1	$\geq$	1
$\neq$	1	1	1	1	1

Figure 18: The composition table for **PA**, except for 0, 1, $1'$

The relation 0 is a null element for composition, i.e. $0 \circ r = r \circ 0 = 0$ for any relation r. Similarly, $1'$ is an identity for composition, i.e. $(1' \circ r) = (r \circ 1') = r$, and since the **PA** is integral *[LadMad88.1]*, we also have that $1 \circ r = 1$ in the **PA**. The full composition table for the **PA** is in the figure, omitting the rows and columns for $0, \approx$, and 1.

We noted in *[LadMad88.1]* that any countable representation of this algebra is isomorphic to the rationals **Q** with the natural $<$, $>$ and $=$ as relations, and that the **PA** is a subalgebra of **IA**. (In fact, any representation of **PA** looks like a dense linear order without endpoints, of whatever cardinality).

The following result and algorithm are essential for our later results.

Theorem 2 *Every path-consistent BCN for* **PA** *is satisfiable*

The proof of this theorem will follow immediately from the proof of correctness of the algorithm below, which produces a satisfying assignment of rational numbers to variables in a path-consistent *BCN* for the **PA**.

PA BCN Satisfaction Algorithm Suppose A is a path-consistent BCN with labels from the **PA**. Suppose A has vertices $1, ..., n$ representing variables $x_1, ..., x_n$, and labels N_{ij} on edge $[i, j]$. The algorithm assigns a rational number v_j to x_j. The assignment proceeds by iteration on the nodes. The algorithm is displayed in the figure.

We assume that, at the i'th stage, we have a sorted list $v_{(k,1)} < ... < v_{(k,i-1)}$ of the values already assigned, i.e. that we have a list of assigned values indexed both by order, and by node-assigned-to. (The k here is just the letter k, added to distinguish one way of listing from the other). There are various easy ways to do this, so there is no need to be more specific. It is also assumed that when a value is assigned to a program variable, such as *upper-bound*, that the node index and the order index are passed with it. Thus one is really passing pointers to a data structure when doing these assignments. Note that we use the symbol 0 both for the rational number 0 and for the relation symbol for $\emptyset$ in the relation algebra, which appears on edges. This should not lead to confusion. There is only one path-consistent BCN with a 0 label on an edge, and that is the network with 0 on all edges, which is trivially testable, and if this algorithm follows a path-consistency algorithm, we assume the former tests for 0 labels. So we assume without loss of generality that there are no 0 labels on any edge.

Since the only output of the algorithm is the values assigned to variables i_j, the ordered list can just be a linked list, whose entries are pairs (v_j, j), and then all the above steps are single algorithm steps, i.e. O(1) in complexity. The algorithm runs in time $2n + e$, where e is the number of edges. The path-consistency algorithm produces a complete graph, so the BCN is complete, and $e = n^2$. Intuitively, to assign a value v_j to i_j, the algorithm looks at the values assigned to all those nodes which, according to the edge labels, are to have a value greater than (or greater-than-or-equal-to) v_j. The minimum value assigned to any of those is noted, and eventually i_j is assigned a value v_j less than any of these, and greater than any of the other nodes which have values assigned so far. The labels $N_{i,j} = \neq, 1$ are resolved in favor of $v_j < v_i$. We now prove correctness of the algorithm.

Proof of Correctness: This proof also proves the theorem. We proceed by induction, and just argue the induction step. Suppose the algorithm produces a satisfying assignment for every network with fewer than i nodes. We shall show the

Step 1: $v_1 \leftarrow 0$; max $= 1$.

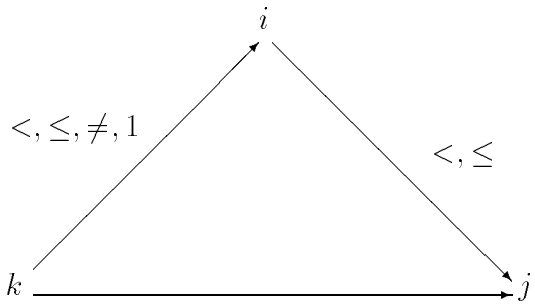
Step i: upper-bound $\leftarrow max$;
for $j < i$ do:
    case $N_{ij} = 1'$: $v_i \leftarrow v_j$ and enddo;
    case $N_{ij} = <, \leq,$: upper-bound $\leftarrow min(v_j$,upper-bound);
    case $N_{ij} = >, \geq, \neq$, 1: continue
enddo;
case upper-bound $= v_j = v_{(k,m)}$ and $1 < m < i$:
    $v_i \leftarrow (v_j + v_{(k,m-1)}) \div 2$;
case upper-bound $= v_j = v_{(k,1)}$:
    $v_i \leftarrow v_j - 1$;
case upper-bound $= max$:
    $\{v_i \leftarrow max$;
    $max \leftarrow max + 1\}$;
insert v_i into the list;
next i

Figure 19: The **PA** Satisfaction Algorithm

algorithm produces one for BCNs with i nodes. Let A' be the network consisting of A with node i and all edges to i deleted. Then $v_1, ..., v_{i-1}$ is a satisfying assignment for A', by inductive assumption, and thus at the i'th step in the algorithm for A we are ok. Suppose for some $j < i, N_{ij} = 1'$. Then $v_i = v_j$. By path-consistency, $N_{ik} = N_{jk}$ for any $k < i$, and so order relations with other values are the same as for v_k, and so A is satisfied by this assignment. Else suppose $M_i = \{j : N_{ij} = <, \leq\}$, and $N_i = \{k : N_{ik} = >, \geq, \neq, 1\}$. See the figure. We claim that for any $j \in M_i, k \in N_i$, we have $v_k < v_i < v_j$. Let $maxN_i = max\{v_k : k \in N_i\}$ and $minM_i = min\{v_j : j \in M_i\}$. We show that $maxN_i < minM_i$ and it follows easily from this that $maxN_i < v_i < minM_i$.

We proceed by induction on the number of nodes in the network, since the algorithm proceeds serially on the nodes. Let i be the number of nodes. The case $i = 1$ is trivial. For the induction step, we assume that the claim is true for the network obtained by deleting the i'th node and all connecting edges. The algorithm runs on this deleted network identically to its first $i - 1$ steps on the full network. It follows that if $j, k < i$, then $v_j < v_k$ if $N_{jk} = <, \leq$, and $v_j > v_k$ if $N_{jk} = >, \geq, \neq, 1$. We show $maxN_i < minM_i$. Suppose $j \in M_i, k \in N_i$. Then the triangle $[i, j, k]$ is labelled as in the figure for N_{ik} and N_{ij}. We consider the labellings on N_{kj}.

Inspection of the figure, and use of the composition table shows that, by path-consistency, in all cases except where $N_{ki} = N_{ij} = \leq$, or $N_{ki} = \neq, 1$, it follows that $N_{kj} = <$ and thus $v_k < v_j$ by inductive assumption. Similarly, if $N_{ki} = N_{ij} = \leq$



The labels shown are N_{ki} (the converse of N_{ik}) and N_{ij}.

Figure 20: The triangle $[i, j, k]$

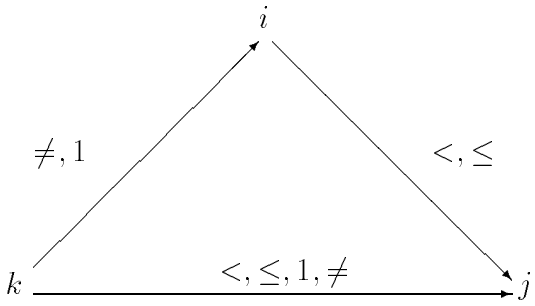


Figure 21: The case N_{ki} is $\neq, 1$

then by path-consistency $N_{kj} = \leq$ and $v_k < v_j$ by assumption. So we consider the cases $N_{ki} = \neq, 1$. The possible labels for N_{kj} are shown in the figure.

Since the possible labels are $N_{kj} = <, \leq, 1, \neq$, by inductive assumption $v_k < v_j$. Hence for any $k \in N_i, j \in M_i$ we have $v_k < v_j$, and thus $maxN_i < minM_i$. The inductive step is proved.
End of Proof.

Corollary 1 *All non-zero path-consistent* **PA** *BCNs have a non-zero path consistent ABCN reduct with* $1'$ *labels on the same edges.*

Proof: Label the edges with the ordering on the v_i produced by the algorithm.
End of Proof.

We include a reduction algorithm for **PA** for completeness, but given that we have a satisfaction algorithm, use of the reduction algorithm would appear to be limited.

PA Reduction Algorithm The algorithm for finding this ABCN is similar to the algorithm for ordering above. The edge labels are replaced, rather than values assigned to the *upper-bound* variable, and the nodes themselves are kept in a sorted list. And, of course, there is no assignment to v_i, just insertion of node i into the list. The label replacements are that $\leq, \neq$ go to $<$, and $\geq$ goes to $>$, excpet for the cases where the equality label is forced by the presence of other equality labels. The algorithm runs in time $n + e$.

6 The Completeness of the IA

We show in this section that reasoning in the Interval Algebra is sufficient for constraint satisfaction over arbitrary finite and countable linear orders S, provided that one is only concerned with the relations that are sums of the thirteen basic relations on S, i.e. that BCSPs labelled only with relations in $PR(T)$ only are satisfiable in any large enough linear order if and only if they are in **IA**.

This result, while not hard, is not immediately apparent, since the Interval Algebra has only interval structures over countable dense linear orders as its models, and the axiomatic theory of interval relations of Allen and Hayes *[AllHay88]* was formulated in order to allow interval structures over arbitrary unbounded linear orders. In *[LadMad88.2]*, we formulate the Allen-Hayes axioms as a relation algebra, about which we know much less than about the Interval Algebra. For example, the composition table is no longer valid for the Allen-Hayes algebra. However, the table may be interpreted as giving upper bounds for the compositions, rather than the actual result of composition, as shown in *[AllHay88]*, i.e. if the table entry opposite row R_i column R_j is t_{ij}, the entry should be interpreted as $R_i \circ R_j \subseteq t_{ij}$. These interpretations are valid for all interval structures which model the Allen-Hayes axioms, i.e. interval structures over unbounded linear orders. This introduces a certain weakness into reasoning with intervals. For example, it is no longer easy to check a network for path-consistency, since the path-consistency condition involves checking a label against a composition. In order to establish path-consistency of a network in the Allen-Hayes formulation, it would be necessary to know the composition table. The composition table for the Allen-Hayes algebra may not be easy to calculate, in particular because the algebra is not finite (*[LadMad88.2]*), and therefore there is no finite collection of atoms in terms of which the table may be expressed.

We can show here that for reasoning with BCNs in $PR(T)$ there is no need to use the Allen-Hayes formulation since the original Allen formulation, the **IA**, suffices.

Define an *interval BCN* to be a BCN whose labels are in $PR(T)$.

Theorem 3 *An interval BCN in n nodes is satisfiable in a linear order S with k distinct points if and only if it is satisfiable in the* **IA** *using intervals with a total of k distinct rational endpoints.*

Corollary 2 *An interval BCN in n nodes is satisfiable in a linear order S with at least $2n$ points if and only if it is satisfiable in the* **IA**.

We should note here that $PR(T)$ is the closure of the thirteen basic relations under sum and converse. By the example below, the result cannot be generalised to labels including intersection or composition of the thirteen basic relations. Hence the result above is best possible. To construct the example, we note that there are nonzero relations in the Allen-Hayes algebra that are zero in the **IA**, for example the relation $1' \cdot (1 \ ; \ d)^-$ is an element properly below $1'$ in both algebras, and is therefore 0 in the **IA**. However, there are many (other) linear orderings which have intervals that are in this relation. Notice that the relation is a subrelation of $1'$, the identity relation, and the condition intuitively says that an interval i is in this relation to itself if and only if i has no subintervals that are during it. Thus let the underlying order be that of the integers. The interval $\langle 3, 4\rangle$ is in the relation $1' \cdot (1 \ ; \ d)^-$ to itself, and therefore the relation is not zero in all interval structures, hence nonzero in the Allen-Hayes relation algebra (it could be zero in the algebra if and only if it were identically zero in every interval structure *[LadMad88.2]*). The one-point network with this label on the edge joining the point to itself is satisfiable in the interval structure over the integers (by, e.g. $\langle 3, 4\rangle$) but not in the **IA**, since it is the one-point zero network over the **IA**.

To prove the theorem, we need to introduce some terminology from *[Lad88.1]*. Let S be a linear order, i.e. a structure $\langle dom(S), <\rangle$ with domain $dom(S)$ and the single binary relation $<$ which is a linear order on $dom(S)$. We shall systematically confuse notation, and often write S for $dom(S)$ when the meaning is clear, e.g. $a \in S$ means strictly $a \in dom(S)$. Let $L(S)$ be the language of S, i.e. a first-order language with a single binary relation symbol. Let $INT(S)$ be the interval structure over S, i.e. the structure whose domain is $\{\langle a, b\rangle \ : \ a \in S \ \wedge \ b \in S \ \wedge \ a < b\}$, and which has the thirteen binary relations over S specified in section 1. (The use of the term *structure* is standard from model theory *[ChaKei73]*). The *language of* $INT(S)$, or $L(INT(S))$ is therefore a first-order language with equality and thirteen binary relation symbols. (We note that the equality relation is also one of the thirteen, thus is redundant. We won't worry about that here). Suppose S, S' are two linear orders. Then $L(S) = L(S')$ and $L(INT(S))) = L(INT(S'))$, since, *inter alia*, neither language contains individual constant symbols or function symbols. Thus we shall use L for the language $L(S)$ where S is an arbitrary linear order, and similarly $L(INT)$ for $L(INT(S))$.

We use the notation

$$M \models \phi\{x_1 \leftarrow a_1, ..., x_n \leftarrow a_n\}$$

where the free variables of ϕ are included in the list $x_1, .., x_n$, and $a_1, ..., a_n$ are elements of the domain of M, to mean that the formula ϕ is true in structure M under all assignments that assign a_1 to x_1, ... , a_n to x_n. We shall implicitly assume that the variables $x_1, ..., x_n$ are all distinct. Let $\phi_R(x, y, x', y')$ be the defining formula in L of the relation R in $INT(S)$, i.e. the formula given in section 1 that defines R in terms of the endpoints of pairs of intervals in R. For example, $\phi_P(x, y, x', y')$ is $x < y < x' < y'$.

As in *[Lad88.1]*, we use three infinite sequences of distinct variables $e_1, .., e_n, ..,$ $f_1, .., f_n, .., g_1, .., g_n, ...$ We shall consider formulas in $L(INT)$ to contain variables only from the list of e_n, and formulas in L only to contain variables from the two other lists. The intuitive reason for the three different lists of variables is that we shall be translating assertions about intervals into assertions about their left and right endpoints, and vice versa, and so we shall associate each interval variable e_n with corresponding variables f_n for its left endpoint, and g_n for its right endpoint. We shall use metavariables such as z, w, x, x', y, y' to range over the list of e_n's, f_n's, and g_n's. We shall use the notation $\phi(x_1, \cdots, x_n)$ to indicate that the free variables of ϕ are amongst $x_1 \cdots x_n$, and the notation $\phi[x_1, \cdots, x_n]$ to indicate that the free variables of ϕ are exactly $x_1 \cdots x_n$, i.e. they're all used in ϕ. We shall use the functions $(-)_L, (-)_R : INT(S) \to S$ to denote the left and right endpoints of intervals in $INT(S)$, i.e. for $i \in INT(S)$, $(i)_L, (i)_R \in S$ and $i = \langle (i)_L, (i)_R \rangle$. Define an *order relation* to be a formula of the form

$$u_{i_1} < u_{i_2} < ..... < u_{i_k}$$

where the iterated $<$ is shorthand for the conjunction of atomic formulas involving adjacent variables, and each u_{i_j} is either some f_m or some g_n.

Lemma 7 *Suppose $\theta[e_1, \cdots, e_n]$ is a conjunction of atomic formulae in $L(INT)$. Then there is an order relation $\psi(f_1, g_1, \cdots, f_n, g_n)$ in L such that, for any linear order S and intervals $a_1, \cdots, a_n \in INT(S)$,*

$$INT(S) \models \theta\{e_1 \leftarrow a_1, \cdots, e_n \leftarrow a_n\}$$

$$\Leftrightarrow$$

$$S \models \psi\{f_1 \leftarrow (a_1)_L, g_1 \leftarrow (a_1)_R, \cdots, f_n \leftarrow (a_n)_L, g_n \leftarrow (a_n)_R\}$$

and conversely given any order relation $\psi(f_1, g_1, \cdots, f_n, g_n)$ in L, there is a formula $\theta[e_1, \cdots, e_n]$ in $L(INT)$, which is a conjunction of atomic formulae, satisfying the same condition.

Proof: We define a translation $(-)^*$ from $L(INT)$ to L such that the required ψ is equivalent to θ^*.

- If θ is an atomic formula $R(z,w)$, then θ^* is $\phi_R(x,y,x',y')$, (where if $z = e_n$, then $x = f_n$ and $y = g_n$, and if $w = e_m$, then $x' = f_m$ and $y' = g_m$)
- If $\theta = (\psi \wedge \rho)$ then $\theta^* = (\psi^* \wedge \rho^*)$.

Clearly this definition may be used in induction on the length of θ to define θ^* for all θ of the appropriate form. We show the condition first for θ and θ^*. For atomic θ, the condition is true by definition of ϕ_R. For θ a conjunction of atomic formulas, it is clear that a conjunction is satisfied if and only if each conjunct is satisfied, and hence the condition is true since it holds for every atomic formula in $L(INT)$, and hence for any conjunction of atomic formulas. Finally, we use the standard fact that every conjunction of atomic formula in L is equivalent to an order relation *[ChaKei73]*, to define the formula ψ as such an order relation equivalent to θ^* in the case that θ is satisfiable in some $INT(S)$, and define ψ as $f_1 < f_1$ in the case that θ is unsatisfiable in any $INT(S)$.
End of Proof.

We note that it is straightforward to extend the lemma for θ any Boolean formula in $L(INT)$, by means of the additional clauses

- If $\theta = (\neg\psi)$ then $\theta^* = (\neg\psi^*)$.
- If $\theta = (\psi \vee \rho)$ then $\theta^* = (\psi^* \vee \rho^*)$.
- If $\theta = (\psi \rightarrow \rho)$ then $\theta^* = (\psi^* \rightarrow \rho^*)$.

and the proof may be easily extended in the manner of the similar Lemma 1 of *[Lad88.1]* to give the following lemma.

Lemma 8 *Suppose $\theta[e_1, \cdots, e_n]$ is any quantifier-free formula in $L(INT)$. Then there is a quantifier-free formula $\psi(f_1, g_1, \cdots, f_n, g_n)$ in L, which is a disjunction of order relations, such that, for any linear order S and intervals $a_1, \cdots, a_n \in INT(S)$,*

$$INT(S) \models \theta\{e_1 \leftarrow a_1, \cdots, e_n \leftarrow a_n\}$$

$$\Leftrightarrow$$

$$S \models \psi\{f_1 \leftarrow (a_1)_L, g_1 \leftarrow (a_1)_R, \cdots, f_n \leftarrow (a_n)_L, g_n \leftarrow (a_n)_R\}$$

and conversely given any Boolean formula $\psi(f_1, g_1, \cdots, f_n, g_n)$ in L, there is a formula $\theta[e_1, \cdots, e_n]$ in $L(INT)$, which contains no instances of the negation symbol, (i.e. it is a positive formula) satisfying the same condition.

Proof: For the first part, we use the translation given for the Boolean formulae to give θ^*. Let ϕ be a disjunctive normal form of θ^* We may then eliminate negations in ϕ^*, which appear only in front of atomic formulae, by the following clauses. (The metavariables x, y here range over *any* of the variables of L).

- Replace every formula of the form $\neg\ x < y$ by the formula $(x > y\ \vee\ x = y)$
- Replace every formula of the form $\neg\ x = y$ by the formula $(x > y\ \vee\ x < y)$

Now we put the result back in disjunctive normal form to give a formula ρ which is a disjunction of conjunctions of atomic formulae of L. Finally, each disjunct is equivalent to an order relation (as we noted before in the proof of the previous lemma), so replacing each disjunct in ρ by an equivalent order relation gives ϕ.

To prove the second part, we begin by noting that atomic formulae in L are equivalent to certain disjunctions of atomic formulae in $L(INT)$. To state the translations, we first observe that from the definitions of the interval relations in section 1, we have the following equivalences.

- $i_L < j_L \ \Leftrightarrow\ i\ (P\ +\ M\ +\ O\ +\ F^{\smile}\ +\ D^{\smile})\ j$
- $i_R < j_R \ \Leftrightarrow\ i\ (P\ +\ M\ +\ O\ +\ S\ +\ D)\ j$
- $i_L < j_R \ \Leftrightarrow\ i\ (P\ +\ M\ +\ O\ +\ Id(L)\ +\ S\ +\ F\ +\ D)\ j$
- $i_R < j_L \ \Leftrightarrow\ i\ P\ j$
- $i_L = j_L \ \Leftrightarrow\ i\ (S\ +\ Id(L)\ +\ S^{\smile})\ j$
- $i_R = j_R \ \Leftrightarrow\ i\ (F\ +\ Id(L)\ +\ F^{\smile})\ j$
- $i_L = j_R \ \Leftrightarrow\ i\ M^{\smile}\ j$
- $i_R = j_L \ \Leftrightarrow\ i\ M\ j$

To motivate the syntactic definition that follows, consider the following example.

$$S \models (f_i < f_j)\{f_i \leftarrow i_L, f_j \leftarrow j_L\} \ \Leftrightarrow\ i_L < j_L \ in\ S$$

and *mutatis mutandis* for the other seven clauses of the equivalences (with the i_R interpreting variables g_i, of course). Hence we may define the translations from a quantifier-free formula ϕ of L to a quantifier-free formula $\phi^\dagger$ of $L(INT)$ in the following way. We use the lower case roman letters $p, d, o, m, s, f, p\ \smile, d\ \smile, o\ \smile, m\ \smile, s\ \smile, f\ \smile$ to stand for the twelve relation symbols of $L(INT)$ that we need to express the translation.

- If $\phi = (f_m < f_n)$ then
  $\phi^\dagger = (p(e_m, e_n) \ \vee \ m(e_m, e_n) \ \vee \ o(e_m, e_n) \ \vee \ f^\smile(e_m, e_n)$
  $\vee \ d^\smile(e_m, e_n))$

- If $\phi = (g_m < g_n)$ then
  $\phi^\dagger = (p(e_m, e_n) \ \vee \ m(e_m, e_n) \ \vee \ o(e_m, e_n) \ \vee \ s(e_m, e_n) \ \vee \ d(e_m, e_n))$

- If $\phi = (f_m < g_n)$ then
  $\phi^\dagger = (p(e_m, e_n) \ \vee \ m(e_m, e_n) \ \vee \ o(e_m, e_n) \ \vee \ s(e_m, e_n)$
  $\vee \ e_m = e_n \ \vee \ f(e_m, e_n) \ \vee \ d(e_m, e_n))$

- If $\phi = (g_m < f_n)$ then $\phi^\dagger = p(e_m, e_n)$

- If $\phi = (f_m = f_n)$ then
  $\phi^\dagger = (s(e_m, e_n) \ \vee \ e_m = e_n \ \vee \ s^\smile(e_m, e_n))$

- If $\phi = (g_m = g_n)$ then
  $\phi^\dagger = (f(e_m, e_n) \ \vee \ e_m = e_n \ \vee \ f^\smile(e_m, e_n))$

- If $\phi = (f_m = g_n)$ then
  $\phi^\dagger = m^\smile(e_m, e_n)$

- If $\phi = (g_m = f_n)$ then
  $\phi^\dagger = m(e_m, e_n)$

We give an example of how this translation works.

$$S \models (g_m < g_n)\{g_m \leftarrow i_R, g_n \leftarrow j_R\}$$

$$\Leftrightarrow$$

$$i_R < j_R \ in \ S$$

$$\Leftrightarrow$$

$$i \ (P \ + \ M \ + \ O \ + \ S \ + \ D) \ j \ in \ S$$

$$\Leftrightarrow$$

$$INT(S) \models (p(e_m, e_n) \ \vee \ m(e_m, e_n) \ \vee \ o(e_m, e_n) \ \vee \ s(e_m, e_n) \ \vee \ d(e_m, e_n))$$
$$\{e_m \leftarrow i, e_n \leftarrow j\}$$

$$\Leftrightarrow$$

$$INT(S) \models \phi^\dagger\{e_m \leftarrow i, e_n \leftarrow j\}$$

We extend the translation to all of the Boolean formulae of L by the clauses

- If $\phi = (\neg\psi)$ then $\phi^\dagger = (\neg\psi^\dagger)$.

- If $\phi = (\psi \wedge \rho)$ then $\phi^\dagger = (\psi^\dagger \wedge \rho^\dagger)$.
- If $\phi = (\psi \vee \rho)$ then $\phi^\dagger = (\psi^\dagger \vee \rho^\dagger)$.
- If $\phi = (\psi \rightarrow \rho)$ then $\phi^\dagger = (\psi^\dagger \rightarrow \rho^\dagger)$.

Finally, $\psi^\dagger$ is almost the formula θ we wish for. To obtain θ, we must ensure that all the variables in the list $e_1 \cdots e_n$ are used. Thus for each variable e_i not mentioned in $\psi^\dagger$, we conjoin the formula $e_i = e_i$ to get θ.

We can improve this result slightly by noting that any Boolean formula $\psi(f_1, g_1, \cdots, f_n, g_n)$ in L is equivalent to a positive formula, by the negation-elimination above, and thence to a disjunction of order relations, by disjunctive normal form and using the equivalence of a conjunction of atomic formulas to an order relation. Hence we may assume without loss of generality that $\psi(f_1, g_1, \cdots, f_n, g_n)$ is a disjunction of order relations. So the θ we obtain is a disjunction of conjunctions of eight kinds of disjunction of atomic formulas in $L(INT)$, the eight kinds corresponding to the translation of the atomic formulas of L above.
End of Proof.

We now state a simple lemma concerning preservation of order relations under strict order-preserving mappings between orderings *[ChaKei73]*. A *strict order-preserving mapping* $F : S \rightarrow S'$ is a function from S to S' such that, for $a, b \in S$, $a < b \Rightarrow F(a) < F(b)$. Notice that strict order-preserving mappings are necessarily one-to-one.

Lemma 9 (Standard) *Suppose $\psi(f_1, g_1, \cdots, f_n, g_n)$ is an order relation, and $H : S \rightarrow S'$ is a strict order-preserving mapping between linear orders S and S'. Then*

$$S \models \psi\{f_1 \leftarrow l_1, g_1 \leftarrow r_1, \cdots f_n \leftarrow l_n, g_n \leftarrow r_n\}$$

$$\Leftrightarrow$$

$$S' \models \psi\{f_1 \leftarrow H(l_1), g_1 \leftarrow H(r_1), \cdots f_n \leftarrow H(l_n), g_n \leftarrow H(r_n)\}$$

i.e. a collection of points satisfies ψ in S if and only if the images under H also satisfy ψ in S'.

We also need another standard lemma about orderings *[ChaKei73]*.

Lemma 10 (Standard) *Suppose S is a finite or countable linear order. Then there exists a strict order-preserving mapping $H : S \rightarrow \mathbf{Q}$, the rational numbers.*

If H is a strict order-preserving mapping $S \rightarrow S'$, let F_H be the mapping $INT(S) \rightarrow INT(S')$ defined by $F_H(\langle m,n\rangle) = \langle H(m), H(n)\rangle$. We call F_H the *mapping induced by* H. F_H is well-defined by the strictness of H, since $m < n \Rightarrow H(m) < H(n)$, so the image of $\langle m,n\rangle$ is a proper interval.

Lemma 11 *Let* $H : S \rightarrow \mathbf{Q}$ *be strict order-preserving. Let* $\theta[e_1, \cdots, e_n]$ *be a conjunction of atomic formulae in* $L(INT)$*. Then*

$$INT(S) \models \theta\{e_1 \leftarrow a_1, \cdots, e_n \leftarrow a_n\}$$

$$\Leftrightarrow$$

$$INT(\mathbf{Q}) \models \theta\{e_1 \leftarrow F_H(a_1), \cdots, e_n \leftarrow F_H(a_n)\}$$

Before we prove the main theorem of this section, we need a further lemma connecting satisfaction of a BCN with satisfaction (in the first-order sense) of a formula in $L(INT)$.

Lemma 12 *Suppose* A *is satisfiable over* S*, and* $a_1, ..., a_n \in INT(S)$ *satisfy* A*. Then there is a formula* $\theta[e_1, \cdots, e_n]$ *in* $L(INT)$ *such that* θ *is a conjunction of atomic formulas in* $L(INT)$ *and* $INT(S) \models \theta\{e_1 \leftarrow a_1, \cdots, e_n \leftarrow a_n\}$*.*

Proof: Since A is an interval BCN, each relation R_{jk} between nodes is a disjunction of atomic relations. We may assume without loss of generality that every pair of variables i_j, i_k for $j,k \leq n$ has a constraint expressed in A. If $a_1 \cdots a_n$ satisfy A, then $a_j \ R_{jk} \ a_k$ for each $j,k \leq n$. Since the atomic relations form a partition of $dom(INT(S))$, each a_j, a_k pair satisfy precisely one of the atomic relations in the disjunction R_{jk}. Let r_{jk} be the atomic relation satisfied by the pair of intervals a_j, a_k for each $j,k \leq n$, and let θ be the conjunction of the formulas $(a_j \ r_{jk} \ a_k)$ for $j,k \leq n$. Then θ satisfies the conditions of the lemma.
End of Proof.

We are now ready to prove the main theorem.

Proof of the Theorem: First, we show that if an interval BCN A is satisfied over S, then it is satisfied in the **IA**. Suppose $a_1 \cdots a_n$ satisfy A over S, and let $\theta[e_1, \cdots, e_n]$ be as in the previous lemma. Then there is an order relation $\psi(f_1, g_1, \cdots, f_n, g_n)$ such that $S \models \psi\{f_1 \leftarrow (a_1)_L, g_1 \leftarrow (a_1)_R, \cdots, f_n \leftarrow (a_n)_L, g_n \leftarrow (a_n)_R\}$ by a previous lemma. Consider the substructure of S whose domain is $S' = \{(a_1)_L, (a_1)_R, \cdots, (a_n)_L, (a_n)_R\}$. Then $S' \models \psi\{f_1 \leftarrow (a_1)_L, g_1 \leftarrow (a_1)_R, \cdots, f_n \leftarrow (a_n)_L, g_n \leftarrow (a_n)_R\}$ by a previous lemma, and hence $INT(S') \models \theta\{e_1 \leftarrow a_1, \cdots, e_n \leftarrow a_n\}$. Let $H : S' \rightarrow \mathbf{Q}$ be a strict order-preserving mapping.

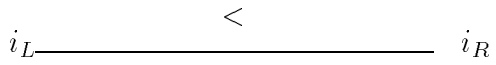


Figure 22: The **PA** network A

Then $INT(\mathbf{Q}) \models \theta\{e_1 \leftarrow F_H(a_1), \cdots, e_n \leftarrow F_H(a_n)\}$. The statement about cardinalities follows directly from counting the number of points involved in the proof.

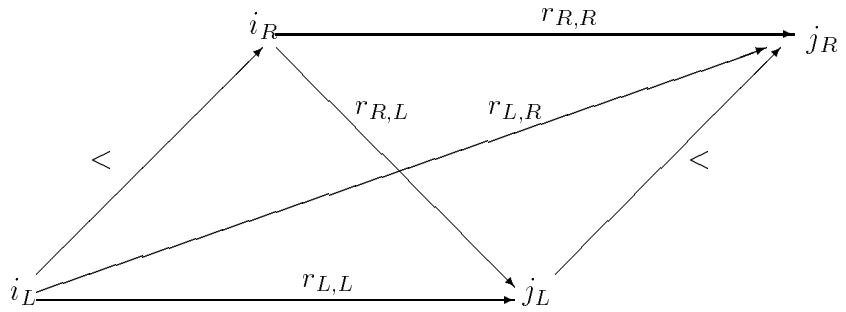
To show the reverse direction, suppose A is satisfied in the **IA**, and let S in the above reasoning be $INT(\mathbf{Q})$. The hypothesis of the theorem assures that the cardinality of the set $\{(a_1)_L, (a_1)_R, \cdots, (a_n)_L, (a_n)_R\}$ is at most k, and therefore that the substructure S' of $INT(\mathbf{Q})$ selected, of which this is the domain, is of size less than k. From the reasoning above, we know that $INT(S') \models \theta\{e_1 \leftarrow a_1, \cdots, e_n \leftarrow a_n\}$. and hence $a_1 \cdots a_n$ satisfy A over $INT(S')$. For any linear order S with at least k distinct points, there is a strict order-preserving mapping H ; $S' \rightarrow S$, and hence $INT(S) \models \theta\{e_1 \leftarrow H(a_1), \cdots, e_n \leftarrow H(a_n)\}$. Thus $H(a_1) \cdots H(a_n)$ satisfy A over S.
End of Proof.

7 A Construction of Complex Networks.

The Construction

We consider the **PA**-network, A, on two nodes, with one edge labelled $<$. Suppose we take two copies of A, say over the nodes i_L, i_R and j_L, j_R respectively, and complete them in a path-consistent manner, with *atomic* labels on the edges, i.e. construct the complete graph between these two networks, and label each edge with an atom of **PA**, in such a way that the resulting network is path-consistent. There are thirteen such ways of doing this, illustrated in the figure. If the network A is considered to correspond to an interval, then two isomorphic copies of A correspond to two intervals, and if we look at the 13 path-consistent completions, reading the labels on the edges in the natural way, then we can see that the networks correspond to the 13 atomic relations of the **IA**. In the table, we identify each path-consistent labelling by prepending its equivalent **IA** label, on the left. We shall use this correspondence to define a relation algebra which will be isomorphic to the **IA**.

We glorify, by assigning lemmahood to, the fact that these 13 are exactly the path-consistent atomic labellings.



$$i_L < i_R \ \wedge \ j_L < j_R \ \wedge \ i_L \, r_{L,L} \, j_L \ \wedge \ i_L \, r_{L,R} \, j_R \ \wedge \ i_R \, r_{R,L} \, j_L \ \wedge \ i_R \, r_{R,R} \, j_R$$

Figure 23: Completing the BCN

Lemma 13 *The thirteen networks in the table are precisely the path consistent completions of the disjoint sum of two networks isomorphic to A.*

The proof is just straightforward verification, using the composition table for **PA**. One should note, here, that although this step might appear trivial, it depends on the **PA** having the appropriate composition table. For algebras which arise from other orderings besides the rationals, this approach does not necessarily give the same result.

A more formal way of describing the construction is to take the BCN which consists of two disjoint copies of A, and completing the BCN path-consistently with atomic labels i.e. forming the complete graph, labelling the edges with **PA** atoms and taking only the path-consistent resulting complete graphs on four nodes.

We shall show how to turn these thirteen networks into the atoms of a new relation algebra, which is isomorphic to the interval algebra. We have to define the objects and the operations on this algebra. The domain of the algebra is the set algebra on these 13 objects, i.e. the objects in the algebra are *sets* of the 4-node networks above. Any such set is an object in the algebra. The sum operation is defined as set union, and product as set intersection. The collection of all the diagrams is the 1 of the algebra, and the empty set is the 0. Thus we can see that the collection of diagrams above forms a Boolean algebra. To turn it into a relation algebra, we need to define the identity object $1'$, and the converse and composition operations on the atoms. As we have noted *[LadMad88.1, Lyn50]*, defining converse and composition on the atoms is sufficient to define it on the whole algebra, because both operations commute

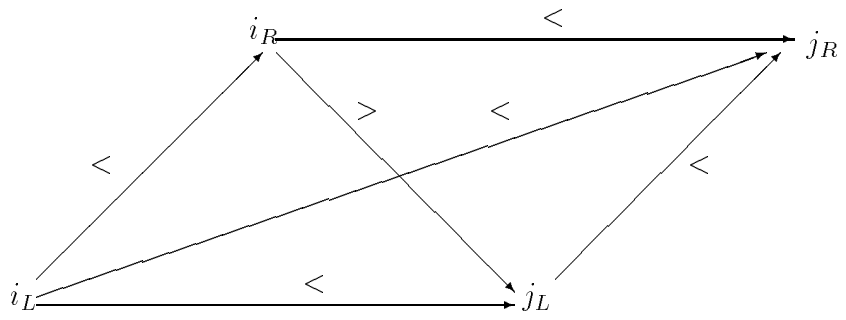
	$r_{L,L}$	$r_{L,R}$	$r_{R,L}$	$r_{R,R}$
$Id(S)$	$=$	$<$	$>$	$=$
P	$<$	$<$	$<$	$<$
D	$>$	$<$	$>$	$<$
O	$<$	$<$	$>$	$<$
M	$<$	$<$	$=$	$<$
S	$=$	$<$	$>$	$<$
F	$>$	$<$	$>$	$=$
$P^\smile$	$>$	$>$	$>$	$>$
$D^\smile$	$<$	$<$	$>$	$>$
$O^\smile$	$>$	$<$	$>$	$>$
$M^\smile$	$>$	$=$	$>$	$>$
$S^\smile$	$=$	$<$	$>$	$>$
$F^\smile$	$<$	$<$	$>$	$=$

Figure 24: The 13 path-consistent atomically-labelled completions

with the sum operator. The identity object $1'$ is the singleton set whose member is the diagram with labels corresponding to the row in the table labelled $Id(S)$. Similarly, the converses to each of the atoms are identified by the relation names on the left of each in the table. The compositions are defined by the construction we shall give. The definition of composition will complete the definition of the relation algebra.

The names we have given the diagrams above may be considered as the definition of an isomorphism from the algebra of diagrams to the **IA**, which maps the diagram onto the element in the **IA** with that name. An isomorphism of the Boolean part is generated by a one-to-one mapping of the atoms, so it is sufficient to check converse and $1'$ (which are definitional, and therefore trivial to check) and composition on the atoms. So once we have defined composition, we shall be able to verify that we have defined (an isomorphic copy of) the **IA** as a purely relation-algebraic construction from the **PA**.

To define composition, we put two diagrams together, identifying the second 'interval' in the first with the first 'interval' in the second, to give a BCN with six nodes. We label the edges between the first four nodes with the labels corresponding to the atom that is the first argument to the composition, and the edges between the last four with the labels corresponding to the atom that is the second argument. Then we complete the network by adding the edges between the first and last pair, and for the new edges we consider all path-consistent ways to label with **PA** atoms. Each way gives a different path-consistent network with atomic labels. For each of



$$i_L < i_R \ \wedge \ j_L < j_R \ \wedge \ i_L < j_L \ \wedge \ i_L < j_R \ \wedge \ i_R > j_L \ \wedge \ i_R < j_R$$

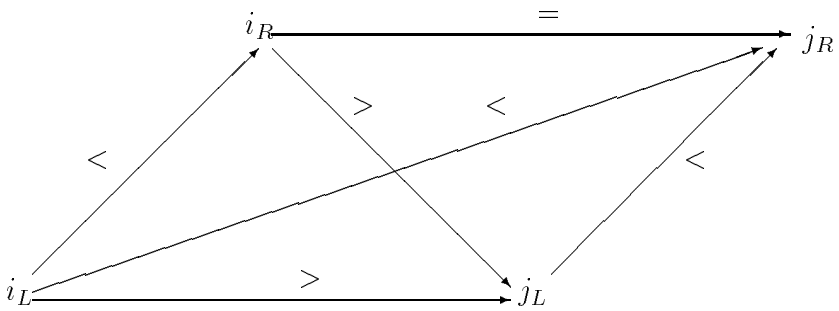
Figure 25: The network corresponding to O

these new networks, we obtain 4-node subnetworks by deleting the middle two nodes, and all edges involving those nodes. Each of these 4-node subnetworks is atomically labelled, and path-consistent since the bigger network is, and thus is one of the list of 13 atoms. Finally, the composition is defined as the set of atoms obtained in this way.

For example, consider the diagram with six nodes obtained from O and F. This network has six nodes, and the labels on the edges connecting the leftmost two nodes to the rightmost have not yet been defined. In the case of O and F. this gives 3 ways to complete the network, as in the figure. We now consider the subnetworks of the 3 4-node networks obtained from these 3 6-node networks by removing the middle pair of nodes. These networks correspond to O, S and D. The composition $O \circ F$ is defined as this 3-element set of atoms (remember that each element of the algebra is a set of atoms), which is the union, and thus the relation-algebraic sum, of the singletons containing the atoms. Thus we conclude that $(O \circ F) = (O + S + D)$. We illustrate this process in the figures.

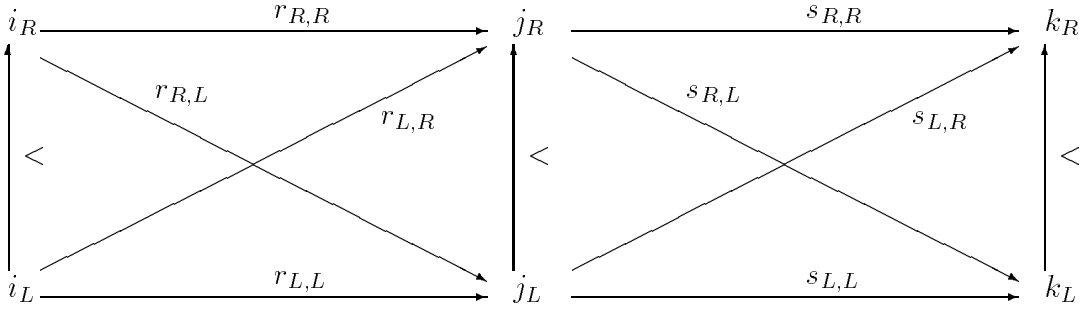
In order to define the full composition table, we perform this construction for each ordered pair of atoms, which gives 169 constructions. We have the following theorem, which is straightforward but very tedious to prove.

Theorem 4 *The algebra defined above, with the definition of composition given, is isomorphic to* **IA**.



$$i_L < i_R \ \wedge \ j_L < j_R \ \wedge \ i_L > j_L \ \wedge \ i_L < j_R \ \wedge \ i_R > j_L \ \wedge \ i_R = j_R$$

Figure 26: The network corresponding to F



To be completed: $(i_L \ ? \ k_L)$, $(i_L \ ? \ k_R)$, $(i_R \ ? \ k_L)$, $(i_R \ ? \ k_R)$.

Figure 27: The uncompleted composition BCN

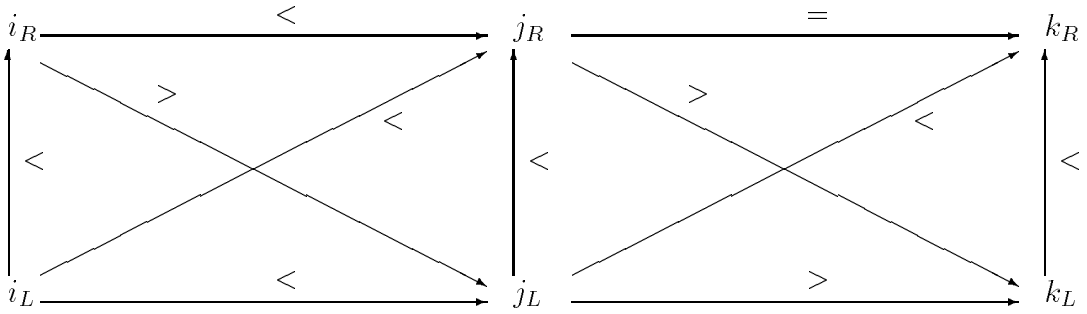
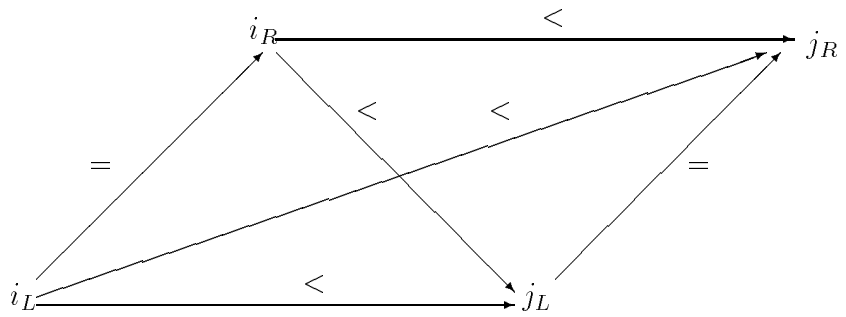


Figure 28: The composition BCN for $(O \circ F)$

	$(i_L?k_L)$	$(i_L?k_R)$	$(i_R?k_L)$	$(i_R?k_R)$
O	$<$	$<$	$>$	$>$
S	$=$	$<$	$>$	$>$
D	$>$	$<$	$>$	$>$

We conclude that $(O \circ F) \ = \ (O + S + D)$

Figure 29: The path-consistent completions of the BCN for $O \circ F$



$$i_L = i_R \ \wedge \ j_L = j_R \ \wedge \ i_L < j_L \ \wedge \ i_L < j_R \ \wedge \ i_R < j_L \ \wedge \ i_R < j_R$$

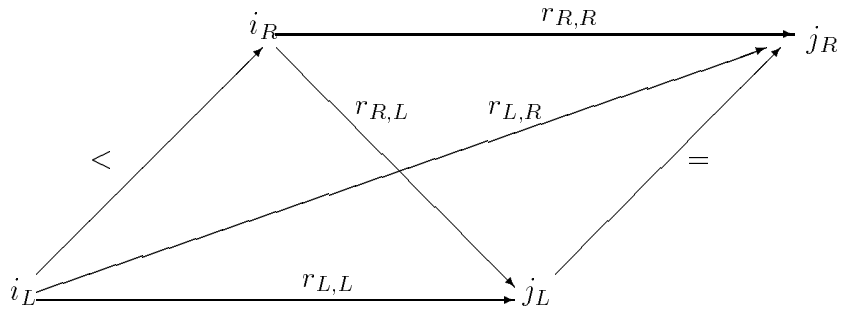
Figure 30: One of the = - = networks

Proof Sketch: The relevant observations to justify the theorem have been made above, and verifying that the composition of atoms as defined leads to the same composition table as in *[All83]* is lengthy but routine verification.
End of Sketch.

Other Constructed Algebras

Using this technique, we may start with a different class of **PA**-networks to build atoms from. We may start not just from a single network, but from a class of networks.

For example, suppose we include the 2-node network with the label = as well as the <-network. Thus we start with a class of two networks, each with two nodes, the <-network (as we have called it), and the =-network. Let this class be N. The construction proceeds by taking two copies of members of N, i.e. the graph which has two components, each isomorphic to a member of N, and completing the graph in every path-consistent way with labels which are atoms of **PA**. We obtain a new algebra with 26 atoms, which we call the *Point-Interval Algebra* **PIA**. The 26 atoms are classified in the following way. There are 13 atoms which come from the construction above, with two <-networks. Next, there are 3 networks which come from two =-networks. The only path-consistent way of completing two =-networks is by labelling all four additional edges with the same label, since = is both a left and a right identity for composition. Hence all four additional edges may be labelled with $<$, $>$ or $=$, giving three path-consistent completions.



$$i_L < i_R \ \wedge \ j_L = j_R \ \wedge \ i_L \, r_{L,L} \, j_L \ \wedge \ i_L \, r_{L,R} \, j_R \ \wedge \ i_R \, r_{R,L} \, j_L \ \wedge \ i_R \, r_{R,R} \, j_R$$

Figure 31: Completing a $<$ - $=$ network

$r_{L,L}$	$r_{L,R}$	$r_{R,L}$	$r_{R,R}$
$>$	$>$	$>$	$>$
$=$	$=$	$>$	$>$
$<$	$<$	$>$	$>$
$<$	$<$	$=$	$=$
$<$	$<$	$<$	$<$

Figure 32: The 5 path-consistent atomically-labelled $<$ - $=$ networks

Now consider all networks obtained by taking a left $<$-network with a right $=$-network. Again, since $=$ is a left and a right identity for composition, the path-consistent ways of completing these networks are those in which both right nodes have identical labels on the edges to the corresponding left nodes, i.e. the subnetworks formed by omitting either right node are isomorphic. It is easy to check that there are 5 such networks, as in the table. Another 5 networks are obtained by taking (what will be defined to be) the converses of these, namely those networks obtained by putting the $=$-network on the left, and the $<$-network on the right of the diagram. This of course gives a symmetric collection to the 5 we have illustrated for $<$ - $=$ networks. Totalling, we have 26 path-consistent, atomically labelled four-node networks in all, starting from the two-network class N, so we have defined the domain of the **PIA**. Sum, product, 0, 1, $1'$ are defined as before, converses are defined as before, and as indicated for the new atoms. Composition is defined as

before, but we need to comment on the definition in the case where the right edge of the first argument to the composition is, say, an =-network, and the left edge of the second argument is a <-network, and vice versa. We cannot just identify these networks when constructing the composition network, as we did before, since they have different labels. What is the intuition here? If the intermediate interval in a composition is to be a degenerate interval according to the first relation (it has an = label), and a non-degenerate interval according to the second (it has a < label), then there can be no such object. This is supposed to be the intermediate object in the definition, and since there can be no such objects, the composition of such relations should intuitively be empty. This gives us the correct network-composition criterion. The intermediate pair of nodes should be labelled with the product of the relations on the corresponding node pairs in the two arguments. So in the case of say an interval-point relation composed with an interval-interval relation, the middle pair of nodes is labelled with two contradictory labels, and there can be no path-consistent completions. Thus the composition is $\emptyset$, i.e. the 0 of the algebra.

Applicability and Formalisation of the Construction

Starting from a class N of networks over a relation algebra A, one may define this construction provided A is an algebra in which every element is determined by the collection of atoms below it. Such an algebra is known as *atomic [JonTar52].* We call a complete network *atomic* if all labels are atoms. The procedure is that of constructing all path-consistent atomic networks extending two disjoint copies of members of N, and defining a new algebra from this by taking the set algebra over this collection, and defining $1'$, and thence converses and compositions in the manner shown.

This process may be phrased in terms of matrices, in the manner of our earlier section. One considers all ways of completing network matrices M with a given upper-left and lower-right corner, with atomic entries, in a path-consistent manner, and then defines converses and compositions.

Pointisable Relations

We consider again the example construction of the **IA** from the **PA**, which involved the class N whose only member is the <-network. We now can consider the relations that can be constructed by completing the BCN, not with atomic relations, but with any **PA** relations, starting with two disjoint copies of the <-network, in a path-consistent manner. There are 187 binary relations on intervals that we obtain in this manner, which we call the *pointisable* relations. The pointisable relations were counted by phrasing the enumeration procedures in terms of network matrices, and checking for path-consistency by squaring the matrices. We enumerate the

pointisable relations below. The significance of this class of relations is shown by the theorem

Theorem 5 *Every nonzero path-consistent BCN over pointisable relations is satisfiable.*

Sketch of Proof: Suppose A is a path-consistent BCN over pointisable relations. Construct a BCN A' over the **PA** by replacing every node i_m of A by a copy of the $<$-network with nodes $i_{m,L}$ and $i_{m,R}$. Replace every edge $[i_m, i_n]$of A by the equivalent **PA** network on $i_{m,L}, i_{m,R}, i_{n,L}$ and $i_{n,R}$. Since the edge has a pointisable label, this is always possible. An inductive argument on the number of nodes in a network shows that the A' is path-consistent if and only if A is. Hence A' is satisfiable, and the **PA** satisfaction algorithm may be used to satisfy A'. Suppose the values $v_{m,L}, v_{m,R}$ satisfy the nodes $i_{m,L}, i_{m,R}$ respectively. We claim that the assignment $i_m \leftarrow \langle v_{m,L}, v_{m,R} \rangle$ satisfies A.
End of Proof Sketch.

Corollary 3 *Checking the consistency of BCNs over pointisable relations is serial cubic, parallel log n time.*

Proof: The replacement of nodes and edges in A to get A' is linear.
End of Proof.

This result extends those noted in *[VilKau86].*

Appendix: Pointisable Relations

The pointisable relations in the **IA** are listed below, followed by first-order formulae defining the relations in terms of the endpoints. The list was created by enumerating 4×4 matrices whose upper left and lower right corners define the $<$-network in the **PA**, completing the matrices and squaring to check for path-consistency. Each relation occupies one line, and + signs are omitted between the atoms, for readability, e.g. a list entry of m $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$ should be read as (m + $\mathrm{f}^\smile$ + o + $\mathrm{d}^\smile$). This list is followed by a listing of the pointisable relations in terms of the relations on the interval endpoints. The intervals are $\langle i_L, i_R \rangle$ and $\langle j_L, j_R \rangle$.

$1'$
s
$1'$ s
$\mathrm{s}^\smile$
$1'$ $\mathrm{s}^\smile$
s $\mathrm{s}^\smile$
$1'$ s $\mathrm{s}^\smile$
m
p
m p
$\mathrm{f}^\smile$
o
$\mathrm{f}^\smile$ o
$\mathrm{d}^\smile$
$\mathrm{f}^\smile$ $\mathrm{d}^\smile$
o $\mathrm{d}^\smile$
$\mathrm{f}^\smile$ o $\mathrm{d}^\smile$
m o
m $\mathrm{f}^\smile$ o
m o $\mathrm{d}^\smile$
m $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$
p o
p $\mathrm{f}^\smile$ o
p o $\mathrm{d}^\smile$
p $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$
m p o
m p $\mathrm{f}^\smile$ o
m p o $\mathrm{d}^\smile$
m p $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$
$1'$ $\mathrm{f}^\smile$
s o

$1'\ s\ f^{\smile}\ o$
$s^{\smile}\ d^{\smile}$
$1'\ s^{\smile}\ f^{\smile}\ d^{\smile}$
$s\ s^{\smile}\ o\ d^{\smile}$
$1'\ s\ s^{\smile}\ f^{\smile}\ o\ d^{\smile}$
$s\ m\ o$
$1'\ s\ m\ f^{\smile}\ o$
$s\ s^{\smile}\ m\ o\ d^{\smile}$
$1'\ s\ s^{\smile}\ m\ f^{\smile}\ o\ d^{\smile}$
$s\ p\ o$
$1'\ s\ p\ f^{\smile}\ o$
$s\ s^{\smile}\ p\ o\ d^{\smile}$
$1'\ s\ s^{\smile}\ p\ f^{\smile}\ o\ d^{\smile}$
$s\ m\ p\ o$
$1'\ s\ m\ p\ f^{\smile}\ o$
$s\ s^{\smile}\ m\ p\ o\ d^{\smile}$
$1'\ s\ s^{\smile}\ m\ p\ f^{\smile}\ o\ d^{\smile}$
$m^{\smile}$
f
d
$f\ d$
$o^{\smile}$
$f\ o^{\smile}$
$d\ o^{\smile}$
$f\ d\ o^{\smile}$
$m^{\smile}\ o^{\smile}$
$m^{\smile}\ f\ o^{\smile}$
$m^{\smile}\ d\ o^{\smile}$
$m^{\smile}\ f\ d\ o^{\smile}$
$p^{\smile}$
$m^{\smile}\ p^{\smile}$
$o^{\smile}\ p^{\smile}$
$f\ o^{\smile}\ p^{\smile}$
$d\ o^{\smile}\ p^{\smile}$
$f\ d\ o^{\smile}\ p^{\smile}$
$m^{\smile}\ o^{\smile}\ p^{\smile}$
$m^{\smile}\ f\ o^{\smile}\ p^{\smile}$
$m^{\smile}\ d\ o^{\smile}\ p^{\smile}$
$m^{\smile}\ f\ d\ o^{\smile}\ p^{\smile}$
$1'\ f$
$s\ d$

1′ s f d
s⌣ o⌣
1′ s⌣ f o⌣
s s⌣ d o⌣
1′ s s⌣ f d o⌣
s⌣ m⌣ o⌣
1′ s⌣ m⌣ f o⌣
s s⌣ m⌣ d o⌣
1′ s s⌣ m⌣ f d o⌣
s⌣ o⌣ p⌣
1′ s⌣ f o⌣ p⌣
s s⌣ d o⌣ p⌣
1′ s s⌣ f d o⌣ p⌣
s⌣ m⌣ o⌣ p⌣
1′ s⌣ m⌣ f o⌣ p⌣
s s⌣ m⌣ d o⌣ p⌣
1′ s s⌣ m⌣ f d o⌣ p⌣
f⌣ f
o d
f⌣ o f d
d⌣ o⌣
f⌣ d⌣ f o⌣
o d⌣ d o⌣
f⌣ o d⌣ f d o⌣
m o d
m f⌣ o f d
m o d⌣ d o⌣
m f⌣ o d⌣ f d o⌣
p o d
p f⌣ o f d
p o d⌣ d o⌣
p f⌣ o d⌣ f d o⌣
m p o d
m p f⌣ o f d
m p o d⌣ d o⌣
m p f⌣ o d⌣ f d o⌣
d⌣ m⌣ o⌣
f⌣ d⌣ m⌣ f o⌣
o d⌣ m⌣ d o⌣
f⌣ o d⌣ m⌣ f d o⌣
m o d⌣ m⌣ d o⌣

m f$^\smile$ o d$^\smile$ m$^\smile$ f d o$^\smile$
p o d$^\smile$ m$^\smile$ d o$^\smile$
p f$^\smile$ o d$^\smile$ m$^\smile$ f d o$^\smile$
m p o d$^\smile$ m$^\smile$ d o$^\smile$
m p f$^\smile$ o d$^\smile$ m$^\smile$ f d o$^\smile$
d$^\smile$ o$^\smile$ p$^\smile$
f$^\smile$ d$^\smile$ f o$^\smile$ p$^\smile$
o d$^\smile$ d o$^\smile$ p$^\smile$
f$^\smile$ o d$^\smile$ f d o$^\smile$ p$^\smile$
m o d$^\smile$ d o$^\smile$ p$^\smile$
m f$^\smile$ o d$^\smile$ f d o$^\smile$ p$^\smile$
p o d$^\smile$ d o$^\smile$ p$^\smile$
p f$^\smile$ o d$^\smile$ f d o$^\smile$ p$^\smile$
m p o d$^\smile$ d o$^\smile$ p$^\smile$
m p f$^\smile$ o d$^\smile$ f d o$^\smile$ p$^\smile$
d$^\smile$ m$^\smile$ o$^\smile$ p$^\smile$
f$^\smile$ d$^\smile$ m$^\smile$ f o$^\smile$ p$^\smile$
o d$^\smile$ m$^\smile$ d o$^\smile$ p$^\smile$
f$^\smile$ o d$^\smile$ m$^\smile$ f d o$^\smile$ p$^\smile$
m o d$^\smile$ m$^\smile$ d o$^\smile$ p$^\smile$
m f$^\smile$ o d$^\smile$ m$^\smile$ f d o$^\smile$ p$^\smile$
p o d$^\smile$ m$^\smile$ d o$^\smile$ p$^\smile$
p f$^\smile$ o d$^\smile$ m$^\smile$ f d o$^\smile$ p$^\smile$
m p o d$^\smile$ m$^\smile$ d o$^\smile$ p$^\smile$
m p f$^\smile$ o d$^\smile$ m$^\smile$ f d o$^\smile$ p$^\smile$
1′ f$^\smile$ f
s o d
1′ s f$^\smile$ o f d
s$^\smile$ d$^\smile$ o$^\smile$
1′ s$^\smile$ f$^\smile$ d$^\smile$ f o$^\smile$
s s$^\smile$ o d$^\smile$ d o$^\smile$
1′ s s$^\smile$ f$^\smile$ o d$^\smile$ f d o$^\smile$
s m o d
1′ s m f$^\smile$ o f d
s s$^\smile$ m o d$^\smile$ d o$^\smile$
1′ s s$^\smile$ m f$^\smile$ o d$^\smile$ f d o$^\smile$
s p o d
1′ s p f$^\smile$ o f d
s s$^\smile$ p o d$^\smile$ d o$^\smile$
1′ s s$^\smile$ p f$^\smile$ o d$^\smile$ f d o$^\smile$
s m p o d

$1'$ s m p $\mathrm{f}^\smile$ o f d
s $\mathrm{s}^\smile$ m p o $\mathrm{d}^\smile$ d $\mathrm{o}^\smile$
$1'$ s $\mathrm{s}^\smile$ m p $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$ f d $\mathrm{o}^\smile$
$\mathrm{s}^\smile$ $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ $\mathrm{o}^\smile$
$1'$ $\mathrm{s}^\smile$ $\mathrm{f}^\smile$ $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ f $\mathrm{o}^\smile$
s $\mathrm{s}^\smile$ o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ d $\mathrm{o}^\smile$
$1'$ s $\mathrm{s}^\smile$ $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ f d $\mathrm{o}^\smile$
s $\mathrm{s}^\smile$ m o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ d $\mathrm{o}^\smile$
$1'$ s $\mathrm{s}^\smile$ m $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ f d $\mathrm{o}^\smile$
s $\mathrm{s}^\smile$ p o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ d $\mathrm{o}^\smile$
$1'$ s $\mathrm{s}^\smile$ p $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ f d $\mathrm{o}^\smile$
s $\mathrm{s}^\smile$ m p o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ d $\mathrm{o}^\smile$
$1'$ s $\mathrm{s}^\smile$ m p $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ f d $\mathrm{o}^\smile$
$\mathrm{s}^\smile$ $\mathrm{d}^\smile$ $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
$1'$ $\mathrm{s}^\smile$ $\mathrm{f}^\smile$ $\mathrm{d}^\smile$ f $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
s $\mathrm{s}^\smile$ o $\mathrm{d}^\smile$ d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
$1'$ s $\mathrm{s}^\smile$ $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$ f d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
s $\mathrm{s}^\smile$ m o $\mathrm{d}^\smile$ d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
$1'$ s $\mathrm{s}^\smile$ m $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$ f d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
s $\mathrm{s}^\smile$ p o $\mathrm{d}^\smile$ d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
$1'$ s $\mathrm{s}^\smile$ p $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$ f d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
s $\mathrm{s}^\smile$ m p o $\mathrm{d}^\smile$ d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
$1'$ s $\mathrm{s}^\smile$ m p $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$ f d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
$\mathrm{s}^\smile$ $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
$1'$ $\mathrm{s}^\smile$ $\mathrm{f}^\smile$ $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ f $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
s $\mathrm{s}^\smile$ o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
$1'$ s $\mathrm{s}^\smile$ $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ f d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
s $\mathrm{s}^\smile$ m o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
$1'$ s $\mathrm{s}^\smile$ m $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ f d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
s $\mathrm{s}^\smile$ p o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
$1'$ s $\mathrm{s}^\smile$ p $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ f d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
s $\mathrm{s}^\smile$ m p o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$
$1'$ s $\mathrm{s}^\smile$ m p $\mathrm{f}^\smile$ o $\mathrm{d}^\smile$ $\mathrm{m}^\smile$ f d $\mathrm{o}^\smile$ $\mathrm{p}^\smile$

$i_L = j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R = j_R,$
$i_L = j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R < j_R,$
$i_L = j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \leq j_R,$
$i_L = j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L = j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L = j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L = j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$

$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R = j_L \ \& \ i_R < j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R < j_L \ \& \ i_R < j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R \leq j_L \ \& \ i_R < j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R = j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R < j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \leq j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R < j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R \leq j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R \neq j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R \, 1 \, j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R < j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R \leq j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R \neq j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R \, 1 \, j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R < j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \leq j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \neq j_R,$
$i_L < j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \, 1 \, j_R,$

$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R = j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R < j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \leq j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R < j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R \leq j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R \neq j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R < j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R \leq j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R \neq j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R < j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \leq j_R,$

$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \neq j_R,$
$i_L \leq j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \, 1 \, j_R,$

$i_L > j_L \ \& \ i_L = j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L > j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R = j_R,$
$i_L > j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R < j_R,$
$i_L > j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \leq j_R,$
$i_L > j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L > j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L > j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L > j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$
$i_L > j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L > j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L > j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L > j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$
$i_L > j_L \ \& \ i_L > j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L > j_L \ \& \ i_L \geq j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L > j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L > j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L > j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L > j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$
$i_L > j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L > j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L > j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L > j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$

$i_L \geq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R = j_R,$
$i_L \geq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R < j_R,$
$i_L \geq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \leq j_R,$
$i_L \geq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L \geq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L \geq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L \geq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \geq j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L \geq j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L \geq j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L \geq j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \geq j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L \geq j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L \geq j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L \geq j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$

$i_L \geq j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L \geq j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L \geq j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L \geq j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$

$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R = j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R < j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \leq j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R < j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R \leq j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R \neq j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R < j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R \leq j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R \neq j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R < j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \leq j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \neq j_R,$
$i_L \neq j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \neq j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L \neq j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L \neq j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L \neq j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \neq j_L \ \& \ i_L \leq j_R \ \& \ i_R \geq j_L \ \& \ i_R \neq j_R,$
$i_L \neq j_L \ \& \ i_L \leq j_R \ \& \ i_R \geq j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \neq j_L \ \& \ i_L \leq j_R \ \& \ i_R \neq j_L \ \& \ i_R \neq j_R,$
$i_L \neq j_L \ \& \ i_L \leq j_R \ \& \ i_R \neq j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \neq j_L \ \& \ i_L \leq j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \neq j_R,$
$i_L \neq j_L \ \& \ i_L \leq j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \neq j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L \neq j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L \neq j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L \neq j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \neq j_L \ \& \ i_L \neq j_R \ \& \ i_R \geq j_L \ \& \ i_R \neq j_R,$
$i_L \neq j_L \ \& \ i_L \neq j_R \ \& \ i_R \geq j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \neq j_L \ \& \ i_L \neq j_R \ \& \ i_R \neq j_L \ \& \ i_R \neq j_R,$

$i_L \neq j_L \ \& \ i_L \neq j_R \ \& \ i_R \neq j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \neq j_L \ \& \ i_L \neq j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \neq j_R,$
$i_L \neq j_L \ \& \ i_L \neq j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \neq j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L \neq j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L \neq j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L \neq j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \neq j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R \geq j_L \ \& \ i_R \neq j_R,$
$i_L \neq j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R \geq j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \neq j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R \neq j_L \ \& \ i_R \neq j_R,$
$i_L \neq j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R \neq j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \neq j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \neq j_R,$
$i_L \neq j_L \ \& \ i_L \, 1 \, j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \, 1 \, j_R,$

$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R = j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R < j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \leq j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R < j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R \leq j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R \neq j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R \geq j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R < j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R \leq j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R \neq j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R \neq j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R < j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \leq j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \neq j_R,$
$i_L \, 1 \, j_L \ \& \ i_L < j_R \ \& \ i_R \, 1 \, j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \, 1 \, j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L \, 1 \, j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L \, 1 \, j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L \, 1 \, j_L \ \& \ i_L \leq j_R \ \& \ i_R > j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \, 1 \, j_L \ \& \ i_L \leq j_R \ \& \ i_R \geq j_L \ \& \ i_R \neq j_R,$
$i_L \, 1 \, j_L \ \& \ i_L \leq j_R \ \& \ i_R \geq j_L \ \& \ i_R \, 1 \, j_R,$
$i_L \, 1 \, j_L \ \& \ i_L \leq j_R \ \& \ i_R \neq j_L \ \& \ i_R \neq j_R,$
$i_L \, 1 \, j_L \ \& \ i_L \leq j_R \ \& \ i_R \neq j_L \ \& \ i_R \, 1 \, j_R,$

$i_L 1 j_L \ \& \ i_L \leq j_R \ \& \ i_R 1 j_L \ \& \ i_R \neq j_R,$
$i_L 1 j_L \ \& \ i_L \leq j_R \ \& \ i_R 1 j_L \ \& \ i_R 1 j_R,$
$i_L 1 j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L 1 j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L 1 j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L 1 j_L \ \& \ i_L \neq j_R \ \& \ i_R > j_L \ \& \ i_R 1 j_R,$
$i_L 1 j_L \ \& \ i_L \neq j_R \ \& \ i_R \geq j_L \ \& \ i_R \neq j_R,$
$i_L 1 j_L \ \& \ i_L \neq j_R \ \& \ i_R \geq j_L \ \& \ i_R 1 j_R,$
$i_L 1 j_L \ \& \ i_L \neq j_R \ \& \ i_R \neq j_L \ \& \ i_R \neq j_R,$
$i_L 1 j_L \ \& \ i_L \neq j_R \ \& \ i_R \neq j_L \ \& \ i_R 1 j_R,$
$i_L 1 j_L \ \& \ i_L \neq j_R \ \& \ i_R 1 j_L \ \& \ i_R \neq j_R,$
$i_L 1 j_L \ \& \ i_L \neq j_R \ \& \ i_R 1 j_L \ \& \ i_R 1 j_R,$
$i_L 1 j_L \ \& \ i_L 1 j_R \ \& \ i_R > j_L \ \& \ i_R > j_R,$
$i_L 1 j_L \ \& \ i_L 1 j_R \ \& \ i_R > j_L \ \& \ i_R \geq j_R,$
$i_L 1 j_L \ \& \ i_L 1 j_R \ \& \ i_R > j_L \ \& \ i_R \neq j_R,$
$i_L 1 j_L \ \& \ i_L 1 j_R \ \& \ i_R > j_L \ \& \ i_R 1 j_R,$
$i_L 1 j_L \ \& \ i_L 1 j_R \ \& \ i_R \geq j_L \ \& \ i_R \neq j_R,$
$i_L 1 j_L \ \& \ i_L 1 j_R \ \& \ i_R \geq j_L \ \& \ i_R 1 j_R,$
$i_L 1 j_L \ \& \ i_L 1 j_R \ \& \ i_R \neq j_L \ \& \ i_R \neq j_R,$
$i_L 1 j_L \ \& \ i_L 1 j_R \ \& \ i_R \neq j_L \ \& \ i_R 1 j_R,$
$i_L 1 j_L \ \& \ i_L 1 j_R \ \& \ i_R 1 j_L \ \& \ i_R \neq j_R,$
$i_L 1 j_L \ \& \ i_L 1 j_R \ \& \ i_R 1 j_L \ \& \ i_R 1 j_R.$

Bibliography

Ada88 : Adams, G., *personal communication*, 1988.

All83 : Allen, J.F., *Maintaining Knowledge about Temporal Intervals*, Comm. A.C.M. 26 (11), November 1983, 832-843.

All86 : Allen, J.F., *personal communication*, 1986.

AllHay88 : Allen J.F. and Hayes, P. J., *Moments and Points in an Interval-Based Temporal Logic*, Technical Report TR180, Dept. of Computer Science, University of Rochester, December 1987.

Bel87 : Bell, C.E., *Representing And Reasoning With Disjunctive Temporal Constraints In A Point-Based Model*, preprint, University of Iowa, Department of Management Sciences.

BurSan81 : Burris, S., and Sankappanavar, H.P., *A Course in Universal Algebra*, Springer Verlag, 1981.

ChaKei73 : Chang, C.C., and Keisler, H.J., *Model Theory*, North-Holland, 1973.

ChiTar51 : Chin, L.H., and Tarski, A., *Distributive and Modular Laws in the Arithmetic of Relation Algebras*, Univ. Calif. Publications in Mathematics, New Series, 1 (9), 1951, 341-384.

DecPea88 : Dechter, R., and Pearl, J., *Network-Based Heuristics for Constraint-Satisfaction Problems*, Artificial Intelligence 34, 1988, 1-38.

DeM64 : De Morgan, A., *On the Syllogism, no. IV, and on the Logic of Relations*, Trans. Cambridge Philos. Soc. 10, 1864, 331-358. (Written 1859, presented to C.P.S. 4/23/1860, published as above).

Fre78 : Freuder, E.C., *Synthesizing Constraint Expressions*, Communications of the ACM 21 (11), Nov 1978, 958-966.

Jòn82 : Jònsson, B., *Varieties of Relation Algebras*, Algebra Universalis 15, 273-298, 1982.

Jòn88 : Jònsson, B., *The Theory of Binary Relations*, preprint.

JònTar52 : Jònsson, B. and Tarski, A., *Boolean Algebras with Operators II*, American J. Mathematics 74, 1952, 127-162.

Lad87.3 : Ladkin, P.B., *Models of Axioms for Time Intervals*, Proceedings of AAAI-87, the Sixth National Conference on Artificial Intelligence, Morgan Kaufmann 1987, 234-239, also available in a longer version as Kestrel Institute Technical Report KES.U.87.4.

Lad87.5 : Ladkin, P.B., *Constraint Satisfaction in Time Interval Structures I: Convex Intervals*, Kestrel Institute Technical Report KES.U.87.11, 1987.

Lad88.1 : Ladkin, P.B., *Satisfying First-Order Constraints about Time Intervals*, to appear, Proceedings of the 7th National Conference on AI, (AAAI-88), 1988.

LadMad87 : Ladkin, P.B. and Maddux, R.D., *The Algebra of Convex Time Intervals*, Kestrel Institute Technical Report KES.U.87.2.

LadMad88.1 : Ladkin, P.B., and Maddux, R.D., *Representation and Reasoning with Convex Time Intervals*, Kestrel Institute Technical Report KES.U.88.2, also submitted for publication.

LadMad88.2 : Ladkin, P.B., and Maddux, R.D., *On the Allen-Hayes Theory of Time Intervals*, in preparation.

Lyn50 : Lyndon, R.C., *The Representation of Relational Algebras*, Annals of Math. Series 2, 51, 1950, 707-729.

Mac77 : Mackworth, A.K., *Consistency in Networks of Relations*, Artificial Intelligence 8, 1977, 99-118.

Mac87 : Mackworth, A.K., *Constraint Satisfaction*, in the *Encyclopedia of Artificial Intelligence*, ed. S. Shapiro, Wiley Interscience 1987.

MacFre85 : Mackworth, A.K., and Freuder, E.C., *The Complexity of Some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems*, Artificial Intelligence 25, 65-74, 1985.

Mad82 : Maddux, R.D., *Some Varieties Containing Relation Algebras*, Transactions of the American Mathematical Society 272, 1982, 501-526.

Mad83 : Maddux, R.D., *A Sequent Calculus For Relation Algebras*, Annals of Pure and Applied Logic 25, 1983, 73-101.

MohHen86 : Mohr, R., and Henderson, T.C., *Arc and Path Consistency Revisited*, Artificial Intelligence 28, 1986, 225-233.

Mon74 : Montanari, U., *Networks of Constraints: Fundamental Properties and Applications to Picture Processing*, Inform. Sci. 7, 1974, 727-732.

Pei92 : Peirce, C.S., *The critic of arguments*, The Open Court 6, 1897, pp. 3391–4, 3416–8. (Reprinted in Peirce 1933).

Pei33 : Peirce, C.S., *Collected Papers, Volume III*, ed. Charles Hartshorne and Paul Weiss, Harvard University Press, Cambridge, 1933.

Sch95 : Schröder, F.W.K.E., *Vorlesungen über die Algebra der Logik (exacte Logik). Vol. III, Algebra und Logik der Relative part I*, Leipzig 1895 (viii + 649pp).
Reprinted Chelsea Publishing Co., N.Y. 1966.

Sch86 : Schulz, K., *An Exact Algorithm for Interval-Based Temporal Information*, FNS - Bericht 86-9, Forschungsstelle für natürliche-sprachliche Systeme, Universität Tübingen, 1986.

Tar41 : Tarki, A., *On the Calculus of Relations*, Journal of Symbolic Logic 6, 1941, 73-89.

TarGiv88 : Tarski, A., and Givant, S.R., *A Formalisation of Set Theory without Variables*, Colloquium Publications in Math 41, Amer. Math. Soc. 1988.

Val87 : Valdés-Pérez, R.E., *The Satisfiability of Temporal Constraint Networks*, Proceedings of AAAI-87, the Sixth National Conference on Artificial Intelligence, pp256-260, Morgan Kaufmann 1987.

VilKau86 : Vilain, M., and Kautz, H., *Constraint Propagation Algorithms for Temporal Reasoning*, Proceedings of AAAI-86, 377-382, Morgan Kaufmann, 1986.