Local and Global Relational Consistency

Rina Dechter[1] and Peter van Beek[2]

[1] Department of Information and Computer Science
University of California, Irvine
Irvine, California, USA 92717
dechter@ics.uci.edu
[2] Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2H1
vanbeek@cs.ualberta.ca

Abstract. Local consistency has proven to be an important concept in the theory and practice of constraint networks. In this paper, we present a new definition of local consistency, called *relational consistency*. The new definition is *relation-based*, in contrast with the previous definition of local consistency, which we characterize as variable-based. It allows the unification of known elimination operators such as resolution in theorem proving, joins in relational databases and variable elimination for solving linear inequalities. We show the usefulness and conceptual power of the new definition in characterizing relationships between four properties of constraints— domain tightness, row-convexity, constraint tightness, and constraint looseness—and the level of local consistency needed to ensure global consistency. As well, algorithms for enforcing relational consistency are introduced and analyzed.

1 Introduction

Constraint networks are a simple representation and reasoning framework. A problem is represented as a set of variables, a domain of values for each variable, and a set of constraints between the variables. A central reasoning task is then to find an instantiation of the variables that satisfies the constraints.

In general, what makes constraint networks hard to solve is that they can contain many local inconsistencies. A local inconsistency is a consistent instantiation of $k-1$ of the variables that cannot be extended to a kth variable and so cannot be part of any global solution. If we are using a backtracking search to find a solution, such an inconsistency can lead to a dead end in the search. This insight has led to the definition of conditions that characterize the level of local consistency of a network [17, 20] and to the development of algorithms for enforcing local consistency conditions by removing local inconsistencies (e.g., [3, 7, 11, 17, 20]).

In this paper, we present a new definition of local consistency called *relational consistency*, introduced recently [24]. We show the usefulness and conceptual power of the new definition in generalizing results for binary networks to non-binary networks on the relationships between four properties of constraints— row-convexity, domain sizes, constraint tightness, and constraint looseness— and the level of local consistency needed to ensure global consistency. As well, algorithms for enforcing relational consistency are introduced and analyzed.

The virtue of the new definition of local consistency is that, firstly, it allows expressing the relationships between the properties of the constraints and local consistency in a way that avoids an explicit reference to the arity of the constraints. Secondly, it is operational, thus generalizing the concept of the composition operation defined for binary constraints, and can be incorporated naturally in algorithms for enforcing desired levels of relational consistency. Thirdly, it unifies known operators such as resolution in theorem proving, joins in relational databases, and variable elimination for solving equations and inequalities. In particular it allows the formulation of an elimination algorithm that generalizes algorithms appearing in each of these areas. Finally, it allows identifying those formalisms for which consistency can be decided by enforcing a bounded level of consistency, like propositional databases and linear equalities and inequalities, from general databases requiring higher levels of local consistency. For space considerations almost all proofs are omitted.

2 Definitions and Preliminaries

Definition 1 constraint network. A *constraint network* $\mathcal{R}$ is a set X of n variables $\{x_1, \ldots, x_n\}$, a domain D_i of possible values for each variable, and a set of relations $R_{S_1}, \ldots, R_{S_t}$, each defined on a subset of the variables $S_1, \ldots, S_t$, respectively. A *constraint* or relation R_S over a set of variables $S = \{x_1, \ldots, x_r\}$ is a subset of the product of their domains (i.e., $R_S \subseteq D_1 \times \cdots \times D_r$). The set of subsets $\{S_1, \ldots, S_t\}$ on which constraints are specified is called the *scheme* of $\mathcal{R}$. A *binary constraint network* is the special case where all constraints are over pairs of variables. An *instantiation* of the variables in X, denoted X_I, is an n-tuple $(a_1, \ldots, a_n)$, representing an assignment of $a_i \in D_i$ to x_i. A *consistent instantiation* of a network is an instantiation of the variables such that the constraints between variables are satisfied. A consistent instantiation is also called a *solution*.

The notion of a consistent instantiation of a subset of the variables can be defined in several ways. We use the following definition: an instantiation is consistent if it satisfies all of the constraints that have no uninstantiated variables.

Definition 2 consistent instantiation of subsets of variables. Let Y and S be sets of variables, and let Y_I be an instantiation of the variables in Y. We denote by $Y_I[S]$ the tuple consisting of only the components of Y_I that correspond to the variables in S. An instantiation Y_I is *consistent* relative to a network $\mathcal{R}$ iff for all S_i in the scheme of $\mathcal{R}$ such that $S_i \subseteq Y$, $Y_I[S_i] \in R_{S_i}$. The set of all consistent instantiations of the variables in Y is denoted $\rho(Y)$. One can view $\rho(Y)$ as the set of all solutions of the subnetwork defined by Y.

Definition 3 operations on constraints. Let R be a relation on a set S of variables, let $Y \subseteq S$ be a subset of the variables, and let Y_I be an instantiation of the variables in Y. We denote by $\sigma_{Y_I}(R)$ the selection of those tuples in R that agree with Y_I. We denote by $\Pi_Y(R)$ the projection of the relation R on the subset Y; that is, a tuple over Y appears in $\Pi_Y(R)$ if and only if it can be extended to a full tuple in R. Let R_{S_1} be a relation on a set S_1 of variables and let R_{S_2} be a relation on a set S_2 of variables. We denote by $R_{S_1} \bowtie R_{S_2}$ the join of the two relations. A tuple t is in the join of R_{S_1} and R_{S_2} if it can be constructed by the following steps: (i) take a tuple r from R_{S_1}, (ii) select a tuple s from R_{S_2} such that the components of r and s agree on the variables that R_{S_1} and R_{S_2} have in common (that is, on the variables $S_1 \cap S_2$), and (iii) form the tuple t by combining the components of r and s, keeping only one copy of components that correspond to variables that the original relations R_{S_1} and R_{S_2} have in common. The resulting relation is on the set of variables given by $S_1 \cup S_2$.

A binary relation R_{ij} between variables x_i and x_j can be represented as a (0,1)-matrix with $|D_i|$ rows and $|D_j|$ columns by imposing an ordering on the domains of the variables. A zero entry at row a, column b means that the pair consisting of the ath element of D_i and the bth element of D_j is not permitted; a one entry means that the pair is permitted.

Four properties of constraints central to this paper are domain-tightness, row-convexity, constraint tightness, and constraint looseness.

Definition 4 k-valued domains. A network of constraints is k-valued if the domain sizes of all variables are bounded by k.

Definition 5 row convex constraints. A binary constraint is *row convex* if in every row of the (0,1)-matrix representation of the constraint, all of the ones are consecutive; that is, no two ones within a single row are separated by a zero in that same row. A binary constraint network is row convex if all its binary constraints are row convex.

Definition 6 m-tight binary constraints. A binary constraint is *m-tight* if every row and every column of the (0,1)-matrix representation of the constraint has at most m ones, where $0 \leq m \leq |D| - 1$. Rows and columns with exactly $|D|$ ones are ignored in determining m. A binary constraint network is m-tight if all its binary constraints are m-tight.

Definition 7 m-loose binary constraints. A binary constraint is *m-loose* if every row and every column of the (0,1)-matrix representation of the constraint has at least m ones, where $0 \leq m \leq |D| - 1$. A binary constraint network is m-loose if all its binary constraints are m-loose.

Definition 8 row convex, m-tight, and m-loose general constraints. An r-ary relation R on a set S of variables $\{x_1, \ldots, x_r\}$ is *row convex* (*m-tight*, *m-loose*, respectively) if for every subset of $r - 2$ variables $Y \subseteq S$ and for every instantiation Y_I of the variables in Y, the binary relation $\Pi_{(S-Y)}(\sigma_{Y_I}(R))$ is row convex (m-tight, m-loose, respectively).

Example 1. We illustrate the definitions using the following network $\mathcal{R}$ over the set X of variables $\{x_1, x_2, x_3, x_4\}$. The network is 3-valued. The domains of the variables are D_i = {a,b,c} and the relations are given by,

$$\begin{aligned} R_{S_1} = \{&(a,a,a),\ (a,a,c),\ (a,b,c),\ (a,c,b),\ (b,a,c),\\ &(b,b,b),\ (b,c,a),\ (c,a,b),\ (c,b,a),\ (c,c,c)\},\\ R_{S_2} = \{&(a,b),\ (b,a),\ (b,c),\ (c,a),\ (c,c)\},\\ R_{S_3} = \{&(a,b),\ (a,c),\ (b,b),\ (c,a),\ (c,b)\}, \end{aligned}$$

where $S_1 = \{x_1, x_2, x_3\}$, $S_2 = \{x_2, x_4\}$, and $S_3 = \{x_3, x_4\}$. The set of all solutions of the network is given by,

$$\begin{aligned} \rho(X) = \{&(a,a,a,b),\ (a,a,c,b),\ (a,b,c,a),\ (b,a,c,b),\\ &(b,c,a,c),\ (c,a,b,b),\ (c,b,a,c),\ (c,c,c,a)\}. \end{aligned}$$

Let $Y = \{x_2, x_3, x_4\}$ be a subset of the variables and let Y_I be an instantiation of the variables in Y. The tuple Y_I = (a,c,b) is consistent relative to $\mathcal{R}$ since $Y_I[S_2]$ = (a,b) and (a,b) $\in R_{S_2}$, and $Y_I[S_3]$ = (c,b) and (c,b) $\in R_{S_3}$. The tuple Y_I = (c,a,b) is not consistent relative to $\mathcal{R}$ since $Y_I[S_2]$ = (c,b), and (c,b) $\notin R_{S_2}$. The set of all consistent instantiations of the variables in Y is given by,

$$\rho(Y) = \{(a,a,b), (a,b,b), (a,c,b), (b,a,c), (b,c,a), (c,a,c), (c,c,a)\}.$$

If we order the domains of the variables according to the natural lexicographic ordering, the (0,1)-matrix representation of the binary constraint R_{S_2} between x_2 and x_4 is given by,

$$R_{S_2} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}.$$

For example, the entry at row 3 column 1 of R_{S_2} is 1, which states that the tuple (c,a) corresponding to the third element of D_2 and the first element of D_4 is allowed by the constraint. It can be seen that the constraint is 2-tight, 1-loose, and not row convex. It can also be verified that the other constraints are 2-tight and 1-loose, and therefore the network is 2-tight and 1-loose. As a partial verification of the ternary constraint R_{S_1}, let $Y = \{x_1\}$ and let Y_I = (a) in the definition. Then, $\sigma_{Y_I}(R_{S_1})$ = {(a,a,a), (a,a,c), (a,b,c), (a,c,b)}, and $\Pi_{(S_1-Y)}(\sigma_{Y_I}(R_{S_1}))$ = {(a,a), (a,c), (b,c), (c,b)}, which is a 2-tight and 1-loose binary relation.

3 Local Consistency

Local consistency has proven to be an important concept in the theory and practice of constraint networks. In this section we first review previous definitions of local consistency, which we characterize as variable-based. We then present new definitions of local consistency that are *relation-based* and present algorithms for enforcing these local consistencies.

3.1 Variable-based consistency

Mackworth [17] defines three properties of networks that characterize local consistency of networks: *node*, *arc*, and *path consistency*. Freuder [11] generalizes this to k-consistency.

Definition 9 k-consistency; Freuder [11, 12]. A network is *k-consistent* if and only if given any instantiation of any $k-1$ distinct variables satisfying all of the direct relations among those variables, there exists an instantiation of any kth variable such that the k values taken together satisfy all of the relations among the k variables. A network is *strongly k-consistent* if and only if it is j-consistent for all $j \leq k$.

Node, arc, and path consistency correspond to one-, two-, and three-consistency, respectively. A strongly n-consistent network is called *globally consistent*. Globally consistent networks have the property that any consistent instantiation of a subset of the variables can be extended to a consistent instantiation of all of the variables without backtracking [5]. It is frequently enough to have a globally consistent network along a single ordering of the variables as long as this ordering is known in advance.

Definition 10 globally solved. We say that a problem is globally solved if it is consistent, and if there is a known ordering of the variables along which solutions can be assembled without encountering deadends. An algorithm *globally solves* a problem if it generates a globally solved network.

3.2 Relation-based consistency

In [25], we extended the notions of arc and path consistency to non-binary relations, and used it to specify an alternative condition under which row-convex non-binary networks are globally consistent. The new local consistency conditions were called relational arc- and path-consistency. In [24] we generalized relational arc- and path-consistency to *relational m-consistency*. In the definition of *relational-consistency*, the relations rather than the variables are the primitive entities. This allows expressing the relationships between properties of the constraints and local consistency in a way that avoids an explicit reference to the arity of the constraints. In this section we revisit the definition of relational consistency and provide algorithms for enforcing such conditions.

Definition 11 relational arc, and path-consistency. Let $\mathcal{R}$ be a constraint network over a set of variables X, and let R_S and R_T be two distinct relations in $\mathcal{R}$, where $S, T \subseteq X$. We say that R_S is *relationally arc-consistent relative to variable x* iff any consistent instantiation of the variables in $S - \{x\}$, has an extension to x that satisfies R_S; that is, iff

$$\rho(S - \{x\}) \subseteq \Pi_{S-\{x\}}(R_S).$$

(Recall that $\rho(A)$ is the set of all consistent instantiations of the variables in A.) A relation R_S is *relationally arc-consistent* iff it is relationally arc-consistent relative to each variable in S. A network is relationally arc-consistent iff every relation is relationally arc-consistent. We say that R_S and R_T are *relationally path-consistent relative to variable* x iff any consistent instantiation of the variables in $(S \cup T) - \{x\}$, has an extension to x that satisfies R_S and R_T simultaneously; that is, iff

$$\rho(A) \subseteq \Pi_A(R_S \bowtie R_T),$$

where $A = (S \cup T) - \{x\}$. A pair of relations R_S and R_T is *relationally path-consistent* iff it is relationally path-consistent relative to each variable in $S \cap T$. A network is relationally path-consistent iff every pair of relations is relationally path-consistent.

Note that the definition of relational path-consistency subsumes relational arc-consistency if in the above definition we do not assume distinct pairs of relations. For simplicity, we will assume that relational path-consistency includes relational arc-consistency.

Definition 12 relational m-consistency. Let $\mathcal{R}$ be a constraint network over a set of variables X, and let $R_{S_1}, \ldots, R_{S_{m-1}}$ be $m-1$ distinct relations in $\mathcal{R}$, where $S_i \subseteq X$. We say that $R_{S_1}, \ldots, R_{S_{m-1}}$ are *relational m-consistent relative to variable* x iff any consistent instantiation of the variables in A, where $A = \bigcup_{i=1}^{m-1} S_i - \{x\}$, has an extension to x that satisfies $R_{S_1}, \ldots, R_{S_{m-1}}$ simultaneously; that is, if and only if

$$\rho(A) \subseteq \Pi_A(\bowtie_{i=1}^{m-1} R_{S_i}).$$

A set of relations $\{R_{S_1}, \ldots, R_{S_{m-1}}\}$ is *relationally m-consistent* iff it is relationally m-consistent relative to each variable in $\bigcap_{i=1}^{m-1} S_i$. A network is relationally m-consistent iff every set of $m-1$ relations is relationally m-consistent. A network is strongly relational m-consistent if it is relational i-consistent for every $i \leq m$. Relational arc- and path-consistency correspond to relational two- and three-consistency, respectively.

Definition 13 directional relational consistency. Given an ordering of the variables, a network is *m-directional relational consistent* iff for every set of relations $\{R_{S_1}, \ldots, R_{S_{m-1}}\}$ it is relationally m-consistent relative to the largest indexed variable in $\bigcap_{i=1}^{m-1} S_i$.

Example 2. Consider the constraint network over the set of variables $\{x_1, x_2, x_3, x_4, x_5\}$, where the domains of the variables are all $D = \{$a, b, c$\}$ and the relations are given by,

$R_{2,3,4,5} = \{$ (a,a,a,a), (b,a,a,a), (a,b,a,a), (a,a,b,a), (a,a,a,b) $\}$,
$R_{1,2,5} \;\; = \{$ (b,a,b), (c,b,c), (b,a,c) $\}$.

The constraints are not relationally arc-consistent. For example, the instantiation $x_2 = a$, $x_3 = b$, $x_4 = b$ is a consistent instantiation as it satisfies all the applicable constraints (trivially so, as there are no constraints defined strictly over $\{x_2, x_3, x_4\}$ or over any subset), but it does not have an extension to x_5 that satisfies $R_{2,3,4,5}$. Similarly, the constraints are not relationally path-consistent. For example, the instantiation $x_1 = c$, $x_2 = b$, $x_3 = a$, $x_4 = a$ is a consistent instantiation (again, trivially so), but it does not have an extension to x_5 that satisfies $R_{2,3,4,5}$ and $R_{1,2,5}$ simultaneously. If we add the constraints $R_2 = R_3 = R_4 = \{\text{a}\}$ and $R_1 = R_5 = \{\text{b}\}$, the set of solutions of the network does not change, and it can be verified that the network is both relationally arc- and path-consistent.

When all of the constraints are binary, relational m-consistency is identical (up to minor preprocessing) to variable-based m-consistency; otherwise the conditions are different. The virtue in our definition (relative to the one based on the dual graph [14]), is that it can be incorporated naturally into algorithms for enforcing desired levels of relational m-consistency, it allows a simple generalization of local consistency relationships, and it unifies several operators of variable elimination.

One disadvantage is that verifying relational m-consistency can be exponential even for relational arc-consistency, if the arity of the constraints is not bounded. In general, however, we will never be interested in verifying relational consistency, but rather in *enforcing* that condition. Below we present algorithm Relational-Consistency or RC_m, a brute-force algorithm for enforcing strong relational m-consistency on a network $\mathcal{R}$. Note that R_A stands for the current unique constraint specified over a subset of variables A. If no constraint exists, then R_A is the universal relation over A.

Relational-Consistency($\mathcal{R}, m$)

1. **repeat**
2. $\quad Q \leftarrow \mathcal{R}$
3. $\quad$ **for** every $m-1$ relations $R_{S_1}, \ldots, R_{S_{m-1}} \in Q$
$\quad$ and every $x \in \bigcap_{i=1}^{m-1} S_i$
4. $\quad\quad$ **do** $A \leftarrow \bigcup_{i=1}^{m-1} S_i - \{x\}$
5. $\quad\quad\quad R_A \leftarrow R_A \cap \Pi_A(\bowtie_{i=1}^{m-1} R_{S_i})$
6. $\quad\quad\quad$ **if** R_A is the empty relation
7. $\quad\quad\quad\quad$ **then** exit and return the empty network
8. **until** $Q = \mathcal{R}$

We call the operation in Step 5 *extended m-composition*, since it generalizes the composition operation defined on binary relations. Algorithm RC_m computes the closure of $\mathcal{R}$ with respect to extended m-composition.

Algorithm RC_m is clearly computationally expensive though it can be improved in a manner parallel to the improvements of path-consistency algorithms [18]. Such improvements are not of much interest since enforcing relational consistency is likely to remain exponential for $m \geq 3$, unless the constraints are binary. We will see that even for $m = 3$, RC_3 solves the NP-complete problem of propositional satisfiability.

As with variable-based local-consistency, we can improve the efficiency of enforcing relational consistency by enforcing it only along a certain direction. Below we present algorithm DIRECTIONAL-RELATIONAL-CONSISTENCY or DRC_m, which enforces strong directional relational m-consistency on a network $\mathcal{R}$, relative to a given ordering $o = x_1, x_2, \ldots, x_n$. We call the network generated by the algorithm the *directional extension of* $\mathcal{R}$*, denoted* $E_m(R)$.

DIRECTIONAL-RELATIONAL-CONSISTENCY($\mathcal{R}, m, o$)

1. Initialize: generate an ordered partition of the constraints, $bucket_1, \ldots,$ $bucket_n$, where $bucket_i$ contains all constraints whose highest variable is x_i.
2. **for** $p \leftarrow n$ **downto** 1
3.   **do** (simplify $bucket_p$:) **for** every $S_i, S_j \in bucket_p$, s.t. $S_i \supseteq S_j$
            do $R_{S_i} \leftarrow R_{S_i} \cap \Pi_{S_i}(R_{S_i} \bowtie R_{S_j})$
4. $j \leftarrow \min\{\text{cardinality of bucket}_p,\ m-1\}$
5.   **for** every j relations $R_{S_1}, \ldots, R_{S_j}$ in $bucket_p$,
6.   **do** $A \leftarrow \bigcup_{i=1}^{j} S_i - \{x_p\}$
7. $R_A \leftarrow R_A \cap \Pi_A(\bowtie_{i=1}^{j} R_{S_i})$
8.   **if** R_A is not the empty relation
9.   **then** add R_A to its appropriate bucket
10.   **else** exit and return the empty network
11. **return** $E_m(R) = \bigcup_{j=1}^{n} bucket_j$

Step 3 of simplifying each bucket is optional. It ensures that there is no relation in a bucket whose variables are contained in another relation's subset.

Although algorithm DRC_m enforces extended m-composition only, it generates a *strong* directional relational m-consistent network in the following sense.

Theorem 14. *The closure under* DRC_m *along ordering* $o = x_1, \ldots, x_n$*, is a network such that all the relations in the same bucket are strong directional relational* m*-consistent.*

Like similar algorithms for enforcing directional consistency, the worst-case complexity of DIRECTIONAL-RELATIONAL-CONSISTENCY can be bounded as a function of the topological structure of the problem via parameters like the *induced width* of the graph [7], also known as *tree-width* [1].

Definition 15 width, tree-width. A constraint network $\mathcal{R}$ can be associated with a constraint graph, where each node is a variable and two variables that appear in one constraint are connected. A general graph can be embedded in a *clique-tree* namely, in a graph whose cliques form a tree-structure. The induced width w^* of such an embedding is its maximal clique size and the induced width w^* of an arbitrary graph is the minimum induced width over all its tree-embeddings.

It is known that finding the minimal width embedding is NP-complete [1], nevertheless every ordering of the variables o, yields a simple to compute upper bound denoted $w^*(o)$ (see [8]). The complexity of DRC_m along o can be bounded as a function of $w^*(o)$ of its constraint graph. Specifically [8],

Theorem 16. *The time complexity and size of the network generated by DRC_m along ordering o is $O(\exp(mw^*(o)))$. In particular, the time complexity of DRC_3 is $O(\exp(w^*(o)+1))$.*

The complexity of DRC_2 is polynomial.

Lemma 17. *The complexity of DRC_2 is $O(n \cdot e^2 \cdot t^2)$ when e is the number of input relations, and t bounds the number of tuples in each relation.*

Example 3. Crossword puzzles have been used experimentally in evaluating backtracking algorithms for solving constraint networks [13]. We use an example puzzle to illustrate algorithm DRC_3 (see Figure 1). One possible constraint network formulation of the problem is as follows: there is a variable for each square that can hold a character, $x_1, \ldots, x_{13}$; the domains of the variables are the alphabet letters; and the constraints are the possible words. For this example, the constraints are given by,

$$
\begin{aligned}
R_{1,2,3,4,5} &= \{(\mathrm{H,O,S,E,S}), (\mathrm{L,A,S,E,R}), (\mathrm{S,H,E,E,T}), (\mathrm{S,N,A,I,L}), (\mathrm{S,T,E,E,R})\} \\
R_{3,6,9,12} &= \{(\mathrm{H,I,K,E}), (\mathrm{A,R,O,N}), (\mathrm{K,E,E,T}), (\mathrm{E,A,R,N}), (\mathrm{S,A,M,E})\} \\
R_{5,7,11} &= \{(\mathrm{R,U,N}), (\mathrm{S,U,N}), (\mathrm{L,E,T}), (\mathrm{Y,E,S}), (\mathrm{E,A,T}), (\mathrm{T,E,N})\} \\
R_{8,9,10,11} &= R_{3,6,9,12} \\
R_{10,13} &= \{(\mathrm{N,O}), (\mathrm{B,E}), (\mathrm{U,S}), (\mathrm{I,T})\} \\
R_{12,13} &= R_{10,13}
\end{aligned}
$$

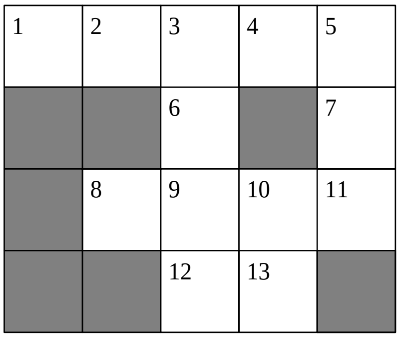

Fig. 1. A crossword puzzle

Let us perform three iterations of DRC_3 with the ordering of variables $o = x_{13}, x_{12}, \ldots, x_1$. Processing bucket$_1$ adds the relation,

$$\begin{aligned} R_{2,3,4,5} &= \Pi_{2,3,4,5}(R_{1,2,3,4,5}) \\ &= \{(\mathrm{O,S,E,S}),\ (\mathrm{A,S,E,R}),\ (\mathrm{H,E,E,T}),\ (\mathrm{N,A,I,L}),\ (\mathrm{T,E,E,R})\}, \end{aligned}$$

to the bucket of variable x_2 which is processed next. The bucket for x_2 contains the single relation $R_{2,3,4,5}$. Processing bucket$_2$ adds the relation,

$$\begin{aligned} R_{3,4,5} &= \Pi_{3,4,5}(R_{2,3,4,5}) \\ &= \{(\mathrm{S,E,S}),\ (\mathrm{S,E,R}),\ (\mathrm{E,E,T}),\ (\mathrm{A,I,L}),\ (\mathrm{E,E,R})\}, \end{aligned}$$

to the bucket of variable x_3 which is processed next. The bucket for x_3 contains the relations $R_{3,4,5}$ and $R_{3,6,9,12}$. Processing bucket$_3$ adds the relations,

$$\begin{aligned} R_{4,5} &= \Pi_{4,5}(R_{3,4,5}) \\ &= \{(\mathrm{E,S}),\ (\mathrm{E,R}),\ (\mathrm{E,T}),\ (\mathrm{I,L}),\ (\mathrm{E,R})\}, \\ R_{6,9,12} &= \Pi_{6,9,12}(R_{3,6,9,12}) \\ &= \{(\mathrm{I,K,E}),\ (\mathrm{R,O,N}),\ (\mathrm{E,E,T}),\ (\mathrm{A,R,N}),\ (\mathrm{A,M,E})\}, \\ R_{4,5,6,9,12} &= \Pi_{4,5,6,9,12}(R_{3,4,5} \bowtie R_{3,6,9,12}) \\ &= \{(\mathrm{E,S,A,M,E}),\ (\mathrm{E,R,A,M,E}),\ (\mathrm{E,T,A,R,N}),\ (\mathrm{I,L,R,O,N}),\ (\mathrm{E,R,A,R,N})\}, \end{aligned}$$

to the buckets of variables x_4 (relations $R_{4,5}$ and $R_{4,5,6,9,12}$) and x_6 (relation $R_{6,9,12}$). Continuing in this manner, at iteration 10 the empty relation is derived and thus the algorithm can stop and report that the network is inconsistent. It can be shown that crossword puzzles can be globally solved by DRC_3.

Finally, we propose algorithm ADAPTIVE-RELATIONAL-CONSISTENCY (ARC) which is the relational counter-part of algorithm adaptive-consistency [7]. Like algorithm DRC_m, it process the buckets in order from last to first. When processing the bucket of x_j, it applies extended composition relative to x_j to *all* the relations in that bucket, and then places the resulting relation in its appropriate bucket. It can be shown that ARC can *globally solve* any constraint network.

ADAPTIVE-RELATIONAL-CONSISTENCY($\mathcal{R}, o$)

Initialize: generate an ordered partition of the constraints, $bucket_1, \ldots, bucket_n$, where $bucket_i$ contains all constraints whose highest variable is x_i.
for $p \leftarrow n$ **downto** 1
    do **for** all the relations $R_{S_1}, \ldots, R_{S_j}$ in $bucket_p$,
        do $A \leftarrow \bigcup_{i=1}^{j} S_i - \{x_p\}$
            $R_A \leftarrow R_A \cap \Pi_A(\bowtie_{i=1}^{j} R_{S_i})$
            if R_A is not the empty relation
                then add R_A to its appropriate bucket
                else exit and return the empty network
return $E_o(R) = bucket_1 \cup bucket_2 \cup \cdots \cup bucket_n$

Theorem 18. *Algorithm* ADAPTIVE-RELATIONAL-CONSISTENCY *(ARC) globally solves any constraint network. The complexity of the algorithm when processed along ordering o is bounded by $O(n \cdot exp((n+e) \cdot w^*(o)))$, where e is the number of relations in the input network.*

4 Variable Elimination Operators

The extended m-composition operator unifies known operators such as resolution in theorem proving, joins in relational databases, and variable elimination for solving equations and inequalities. It is easy to see that pair-wise resolution is equivalent to extended 3-composition.

4.1 Variable elimination in propositional CNF theories

We denote propositional symbols, also called *variables*, by uppercase letters $P, Q, R, ...$, propositional literals (i.e., $P, \neg P$) by lowercase letters $p, q, r, ...$, and disjunctions of literals, or *clauses*, by $\alpha, \beta, ...$ A *unit clause* is a clause of size 1. The notation $(\alpha \vee T)$ is a shorthand for the disjunction $(P \vee Q \vee R \vee T)$, and $\alpha \vee \beta$ denotes the clause whose literal appears in either α or β. The *resolution* operation over two clauses $(\alpha \vee Q)$ and $(\beta \vee \neg Q)$ results in a clause $(\alpha \vee \beta)$, thus eliminating Q. A formula φ in conjunctive normal form (*CNF*) is a set of clauses $\varphi = \{\alpha_1, ..., \alpha_t\}$ that denotes their conjunction. The set of *models* of a formula φ, denoted $models(\varphi)$, is the set of all satisfying truth assignments to all its symbols. A Horn formula is a *CNF* formula whose clauses all have at most one positive literal. Let $EC_Q(R_A, R_B)$ denote the relation generated by extended 3-composition of R_A and R_B relative to Q, $Q \in A \cap B$.

Lemma 19. *The* resolution *operation over two clauses* $(\alpha \vee Q)$ *and* $(\beta \vee \neg Q)$, *results in a clause* $(\alpha \vee \beta)$ *satisfying:* $models(\alpha \vee \beta) = EC_Q(models(\alpha), models(\beta))$.

Incorporating resolution into DRC_3 results in algorithm *Directional Resolution* which is the core of the well known Davis Putnam algorithm for satisfiability [4, 10]. As is well known, and as will follow from our theory, the algorithm globally solves any *CNF* theory.

Directional-Resolution (φ, o)

1. Initialize: generate an ordered partition of clauses to buckets.
2. **for** $i \leftarrow n$ **downto** 1
3. **do** resolve each pair $\{(\alpha \vee Q_i), (\beta \vee \neg Q_i)\} \subseteq bucket_i$. If $\gamma = \alpha \vee \beta$ is empty, return $E_o(\varphi) = \{\}$, the theory is not satisfiable; else, determine the index of γ and add it to the appropriate bucket.
4. **return** $E_o(\varphi) \leftarrow \bigcup_i bucket_i$

It is easy to see that DRC_2 *with the simplification step*, when applied to *CNF* theories is a slight extension of unit-resolution. It allows resolution involving non-unit clauses as long as the variables appearing in one clause are contained in the other clause. Consequently,

Lemma 20. *Algorithm* DRC_2 *decides the satisfiability of Horn theories.*

4.2 Variable elimination in linear inequalities

Let us now consider the class of linear inequalities over finite subsets[3] of the integers and let a constraint between r variables or less be a conjunction of linear equalities and inequalities of the form $\sum_{i=1}^{r} a_i x_i \leq c$, where a_i, and c are integer constants. For example, the conjunction $(3x_i + 2x_j \leq 3) \wedge (-4x_i + 5x_j < 1)$ is an allowed constraint between variables x_i and x_j. A network with constraints of this form can be formulated as an integer linear program where each constraint is on r variables and the domains of the variables are finite subsets of integers that may be restricted by unary linear inequalities. It can be shown that the standard operation of eliminating variables between pairs of inequality is almost equivalent to extended 3-composition. Let us denote by $sol(\alpha)$ the finite set of solutions over the integers of one inequality α. $sol(\alpha)$ is the relational representation of the inequality. We define the elimination operation as follows:

Definition 21 linear elimination. Let $\alpha = \sum_{i=1}^{(r-1)} a_i x_i + a_r x_r \leq c$, and $\beta = \sum_{i=1}^{(r-1)} b_i x_i + b_r x_r \leq d$. Then $elim_r(\alpha, \beta)$ is applicable only if a_r and b_r have opposite signs, in which case $elim_r(\alpha, \beta) = \sum_{i=1}^{r-1}(-a_i \frac{b_r}{a_r} + b_i)x_i \leq -\frac{b_r}{a_r}c + d$. If a_r and b_r have the same sign the elimination implicitly generates the universal constraint.

Lemma 22. *$sol(elim_r(\alpha, \beta)) \supseteq EC_r(sol(\alpha), sol(\beta))$*

Proof. Assume now that a_r and b_r contain opposite signs. Multiplying α by $-\frac{b_r}{a_r}$ and summing the resulting inequality with β yields the inequality

$$\sum_{i=1}^{r-1}(-a_i \frac{b_r}{a_r} + b_i)x_i \leq -\frac{b_r}{a_r}c + d.$$

In other words, any tuple satisfying this inequality can be extended to a *real value* of x_r in a way that satisfies both α and β. It is unclear, though, that there is an integer extension to x_r which is the reason for partial containment. □.

Incorporating linear elimination into DRC_3 results in algorithm *Directional Linear Elimination* (abbreviated DLE) which is the well known Fourier elimination algorithm (see [16]). It was shown that the algorithm decides the solvability of any set of linear inequalities over the Reals.

Directional-Linear-Elimination (φ, o)

1. Initialize: generate an ordered partition of the inequalities into buckets.
2. **for** $i \leftarrow n$ **downto** 1
3. **do** for each pair $\{\alpha, \beta\} \subseteq bucket_i$, compute $\gamma = elim_i(\alpha, \beta)$. If γ has no solutions, return $E_o(\varphi) = \{\}$, the theory is not satisfiable; else, add γ to the appropriate bucket.

[3] Our treatment can be extended to non-finite domains. However to simplify we will assume that the domains are integers bounded between [-M,M] for very large M, whenever other bounds are not explicitly given.

4. **return** $E_o(\varphi) \leftarrow \bigcup_i bucket_i$

If DLE is processed over the relational representation of the linear inequalities, (in which case it becomes DRC_3), we will be able to bound its complexity using the notion of induced width. However, when DLE uses inequality representation only (the only option for infinite domains) its complexity may be worst-case exponential even when the induced width $w*$, is bounded. The reason is that an exponential number of inequalities may be recorded, even on one or two variables. We cannot "intersect" two inequalities and replace them by one. Even for binary inequalities the algorithm may be exponential, unless we use relational representation and DRC_3. In summary,

Theorem 23. *Algorithm DLE is exponential even for binary inequalities and even for bounded induced width. For finite domains DRC_3 is applicable and its complexity is polynomial for binary constraints and bounded induced width.*

Propositional *CNF*s as well as linear inequalities share an interesting syntactic property: It is easy to recognize whether applying extended-3-composition relative to variable x_i results in a universal constraint. Both resolution and linear elimination relative to x_i are effective only when the variable to be eliminated appears with opposite signs. This leads to a simple-to-identify tractable class for both these languages. If there exists an ordering of the variables, such that in each of its $bucket_i$, x_i appears with the same sign, then the theory is already globally solved relative to that ordering. We called in [10] such theories as "zero diversity" and we showed that they can be recognized in linear time.

5 From Local to Global Consistency

Much work has been done on identifying relationships between properties of constraint networks and the level of local consistency sufficient to ensure global consistency. This work falls into two classes: identifying topological properties of the underlying graph of the network and identifying properties of the constraints.

For work on identifying topological properties, Freuder [12] identifies a relationship between the *width* of a constraint graph and the level of local consistency needed to ensure a solution can be found without backtracking. Dechter and Pearl [7] provide an adaptive scheme where the level of local consistency is adjusted on a node-by-node basis. Dechter and Pearl [8] generalize the results on trees to hyper-trees which are called acyclic databases in the database community [2].

For work on identifying properties of the constraints, Montanari [20] shows that path consistency is sufficient to guarantee that a binary network is globally consistent if the relations are monotone. Dechter [5] identifies a relationship between the size of the domains of the variables, the arity of the constraints, and the level of local consistency sufficient to ensure the network is globally consistent. These results were extended recently by van Beek and Dechter to the property of tightness and looseness of the constraint networks [24, 23]. Van

Hentenryck, Deville, and Teng [26] show that arc consistency is sufficient to test whether a network is satisfiable if the relations are from a restricted class of functional and monotone constraints. These properties were generalized recently to the property of row-convexity [25].

Finally, for work that falls into both classes, Dechter and Pearl [9] present effective procedures for determining whether a constraint network can be formulated as a *causal theory* and thus a solution can be found without backtracking. Whether a constraint network can be so formulated depends on the topology of the underlying constraint graph and the type of the constraints.

In this section we show the power of relational consistency in generalizing some recently identified relationships between properties of the constraints and the level of local consistency sufficient to ensure that a network is globally consistent. This formulation leads to a characterization of classes of problems that can be solved by a restricted level m of DRC_m. The general pattern we will see is as follows. We present a sufficient condition showing that a network satisfying a property p, and having a corresponding level of local consistency $l(p)$, is globally consistent. This implies that whenever the property p is maintained under extended $l(p)-composition$, those networks (satisfying p) can be globally solved by $DRC_{l(p)}$. Furthermore, it is sufficient for condition $l(p)$ to hold only relative to the particular ordering on which the algorithm is applied.

5.1 Domain tightness and global consistency

In [5], we have shown that:

Theorem 24. *[5] If $\mathcal{R}$ is a k-valued binary constraint network that is $k+1$ consistent then it is globally consistent. If $\mathcal{R}$ is a k-valued r-ary constraint network that is $k(r-1)+1$ consistent then it is globally consistent.*

We now show that by using the notion of relational consistency the above relationship for r-ary networks (as well as its proof), are simplified. Moreover, the implied algorithm can be stated more coherently.

Theorem 25. *A k-valued constraint network $\mathcal{R}$, that is $k+1$-relational-consistent is globally consistent.*

Since the domains do not increase by extended $(k+1)$-composition we get:

Theorem 26. *Any k-valued network $\mathcal{R}$ can be globally solved by DRC_{k+1}.*

Example 4. From Theorem 26, bi-valued networks can be globally solved by DRC_3. In particular, propositional *CNFs* can be globally solved by DRC_3. As we have seen, in this case, the operator of extended 3-composition takes the form of pair-wise resolution yielding algorithm directional resolution [10].

5.2 Row-convexity and global consistency

In [22], we have shown that:

Theorem 27. *[22] Let $\mathcal{R}$ be a path-consistent binary constraint network. If there exists an ordering of the domains $D_1, \ldots, D_n$ of $\mathcal{R}$ such that the relations are row convex, the network is globally consistent.*

The result for binary networks was generalized to constraints of arbitrary arity, using relational path consistency.

Theorem 28. *[25]. Let $\mathcal{R}$ be a relational path consistent constraint network. If there exists an ordering of the domains $D_1, \ldots, D_n$ of $\mathcal{R}$ such that the relations are row convex, the network is globally consistent.*

We can conclude that:

Theorem 29. *If $\mathcal{R}$ is a network whose closure under extended 3-composition is row convex then $\mathcal{R}$ can be globally solved by DRC_3.*

Example 5. Consider a set of linear equalities and inequalities over finite subsets of integers of the form: $\sum_{i=1}^{r} a_i x_i \leq c$ where a_i, c are integers. It can be shown that the $(0, 1)$ matrices of such relations are row convex and, when the constraints are binary, their row-convexity is invariant to extended 3-composition. It is also easy to see that any bi-valued relation is row-convex. Cosequently,

Theorem 30. *A set of linear inequalities over finite set of integers can be globally solved by DRC_3. A* CNF *formula can be globally solved by DRC_3.*

Two special cases are a restricted and discrete version of Dechter, Meiri, and Pearl's [6] continuous, bounded difference framework for temporal reasoning and a restricted and discrete version of Vilain and Kautz's [27] qualitative framework for temporal reasoning. Another known class that can be shown to be row-convex is implicational constraints [15]. For more details see [25].

5.3 Constraint tightness and global consistency

For some networks, Theorem 24 is tight in that the level of local consistency specified by the theorem is really required (graph coloring problems formulated as constraint networks are an example). For other networks, Theorem 24 overestimates. In [24], we refined that relationship by extending the notion of tightness to the constraints themselves.

Theorem 31. *[24]. If a binary constraint network $\mathcal{R}$ is m-tight, and if the network is strongly $(m + 2)$-consistent, then the network is globally consistent.*

Theorem 32. *[24] If a general constraint network $\mathcal{R}$ is m-tight, and relationally $(m + 2)$-consistent, then the network is globally consistent.*

Theorems 31 & 32 always specify a level of strong consistency that is less than or equal to the level of strong consistency required by Theorem 24. The level of required consistency is equal only when $m = d - 1$ and is less when $m < d - 1$. As well, the theorem can sometimes be usefully applied if $d \geq n - 1$, whereas Theorem 24 cannot.

Example 6. Nadel [21] introduces a variant of the n-queens problem called confused n-queens and uses it to empirically compare backtracking algorithms. In [24], we use Theorem 31 to show that these problems are quite easy (as they require a low level of local consistency to ensure global consistency) and that any empirical results on these problems should be interpreted in this light.

5.4 Constraint looseness and local consistency

In [23], we presented a simple sufficient condition that estimates the inherent level of strong k-consistency of a binary constraint network.

Theorem 33. *[23] If a binary constraint network $\mathcal{R}$ is m-loose and all domains are of size d or less, then the network is strongly $\left(\left\lceil \frac{d}{d-m} \right\rceil\right)$-consistent.*

We now generalize the result to networks with constraints of arbitrary arity.

Theorem 34. *If a general constraint network $\mathcal{R}$ is m-loose and all domains are of size d or less, then the network is relational $\left(\left\lceil \frac{d}{d-m} \right\rceil\right)$-consistent.*

Theorems 33 & 34 provide a lower bound on the actual level of inherent local consistency of a constraint network. Graph coloring problems provide examples where the bound is exact, whereas n-queens problems provide examples where the bound underestimates the true level of local consistency [23].

5.5 Acyclic and causal networks and global consistency

Relational consistency and the DRC_m algorithms can also easily capture the tractable classes of acyclic and causal networks. It is well known that acyclic networks are tractable [19, 8].

Lemma 35. *If a network is acyclic then there exists an ordering of the variables for which each bucket has a single relation.*

Single-bucket networks contain the class of acyclic networks and causal networks. It was shown in [9] that it is possible to discover an ordering of the variables for which each bucket contains a single relation, whenever such an ordering exist. We conclude:

Theorem 36. *Single-bucket networks that are closed under DRC_2 are tractable.*

6 Conclusions

We have shown that different levels of DRC can globally solve different classes of constraint networks:

1. DRC_2 globally solves acyclic and single-bucket, causal relations in polynomial time. It solves, (not globally solves) Horn CNF theories.
2. DRC_3 globally solves closed row-convex networks, bi-valued domain networks, closed 1-tight networks, crossword puzzles, and linear inequalities over finite subsets of the integers. The algorithm is polynomial for binary constraints over finite domains in relational form, and can be exponential otherwise. Algorithm DLE is a linear elimination algorithm that approximates DRC_3 over integers.
3. Algorithm DRC_m globally solves $(m-1)$-valued networks, and closed $(m-2)$-tight networks. The algorithm is polynomial for binary constraints.
4. Algorithm ARC globally solves all networks and is exponential.
5. The complexity of both DRC_m and ARC is exponentially bounded by $w*$, the tree-width of the network over finite domains.

Acknowledgement. We would like to thank Simon Kasif for mentioning Fourier's elimination algorithm to us. This work was partially supported by NSF grant IRI-9157636, By the electrical Power Research institute (EPRI) and by grants from Xerox, Northrop and Rockwell. This work was also supported in part by the Natural Sciences and Engineering Research Council of Canada.

References

1. S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding an embedding in k-trees. *SIAM Journal of Algebraic Discrete Methods*, 8:177–184, 1987.
2. C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30:479–513, 1983.
3. M. C. Cooper. An optimal k-consistency algorithm. *Artif. Intell.*, 41:89–95, 1989.
4. M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7:201–215, 1960.
5. R. Dechter. From local to global consistency. *Artif. Intell.*, 55:87–107, 1992.
6. R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artif. Intell.*, 49:61–95, 1991.
7. R. Dechter and J. Pearl. Network-based heuristics for constraint satisfaction problems. *Artif. Intell.*, 34:1–38, 1988.
8. R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artif. Intell.*, 38:353–366, 1989.
9. R. Dechter and J. Pearl. Directed constraint networks: A relational framework for causal modeling. In *Proc. of the 12th Int'l Joint Conf. on AI*, pages 1164–1170, 1991.
10. R. Dechter and I. Rish. Directional resolution: The Davis-Putnam procedure, revisited. In *Proc. of the 4th Int'l Conf. on Principles of KR&R*, 1994.

11. E. C. Freuder. Synthesizing constraint expressions. *Comm. ACM*, 21:958–966, 1978.
12. E. C. Freuder. A sufficient condition for backtrack-free search. *J. ACM*, 29:24–32, 1982.
13. M. L. Ginsberg, M. Frank, M. P. Halpin, and M. C. Torrance. Search lessons learned from crossword puzzles. In *Proc. of the 8th Nat'l Conf. on AI*, pages 210–215, 1990.
14. P. Jégou. On the consistency of general constraint satisfaction problems. In *Proc. of the 11th National Conf. on AI*, pages 114–119, 1993.
15. L. M. Kirousis. Fast parallel constraint satisfaction. *Artif. Intell.*, 64:147–160, 1993.
16. J-L Lassez and M. Mahler, "On Fourier's algorithm for linear constraints" *Journal of Automated Reasoning*, Vol 9, 1992.
17. A. K. Mackworth. Consistency in networks of relations. *Artif. Intell.*, 8:99–118, 1977.
18. A. K. Mackworth and E. C. Freuder. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artif. Intell.*, 25:65–74, 1985.
19. D. Maier. *The Theory of Relational Databases*. Computer Science Press, 1983.
20. U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Inform. Sci.*, 7:95–132, 1974.
21. B. A. Nadel. Constraint satisfaction algorithms. *Comput. Intell.*, 5:188–224, 1989.
22. P. van Beek. On the minimality and decomposability of constraint networks. In *Proc. of the 10th National Conf. on AI*, pages 447–452, 1992.
23. P. van Beek. On the inherent level of local consistency in constraint networks. In *Proc. of the 12th National Conf. on AI*, pages 368–373, 1994.
24. P. van Beek and R. Dechter. Constraint tightness versus global consistency. In *Proc. of the 4th Int'l Conf. on Principles of KR&R*, pages 572–582, 1994.
25. P. van Beek and R. Dechter. On the minimality and global consistency of row-convex constraint networks. *To appear in J. ACM*, 1995.
26. P. Van Hentenryck, Y. Deville, and C.-M. Teng. A generic arc consistency algorithm and its specializations. *Artif. Intell.*, 57:291–321, 1992.
27. M. Vilain and H. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proc. of the 5th National Conf. on AI*, pages 377–382, 1986.