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. We show the conceptual power of the new definition by showing how it unifies known elimination operators such as resolution in theorem proving, joins in relational databases, and variable elimination for solving linear inequalities. Algorithms for enforcing various levels of relational consistency are introduced and analyzed. We also show the usefulness of the new definition in characterizing relationships between properties of constraint networksand the level of local consistency needed to ensure global consistency.

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 [29, 33] and to the development of algorithms for enforcing local consistency conditions by removing local inconsistencies (e.g., [33, 29, 19, 12, 6]).

In this paper, we present a new definition of local consistency called *relational consistency*[3]. The virtue of the new definition of local consistency is that, firstly, it removes the need for referencing the arity of the constraints when discussing

[3] A preliminary version of this definition appears in [37].

relationships between the properties of the constraints and local consistency. 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, thus allowing 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 relational consistency, like propositional databases, linear equalities and inequalities, and crossword puzzles from general databases requiring higher levels of relational consistency. We also demonstrate the usefulness of the new definition in characterizing relationships between various properties of constraint networks—domain size and acyclicity— and the level of local consistency needed to ensure global consistency.

Following definition and preliminaries (section 2), relational local consistency is defined and algorithms for enforcing such conditions are introduced (section 3). Section 4 shows that the algorithms unify algorithms appearing in propositional databases and linear inequalities. Finally, section 5 describes new associations between constraint properties and relational local consistency needed for global consistency. Discussion and conclusions are given in sections 6 and 7 respectively.

2 Definitions and Preliminaries

Definition 1 (constraint network). [4] A *constraint network* $\mathcal{R}$ is a set of n variables $X = \{x_1, \ldots, x_n\}$, a domain D_i of possible values for each variable x_i, $1 \leq i \leq n$, and a set of t relations $R_{S_1}, \ldots, R_{S_t}$, where $S_i \subseteq X$, $1 \leq i \leq n$. 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. A constraint graph associates each variable with a node and connects any two nodes whose variables appear in the same constraint.

Definition 2 (solution to a constraint network). 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, $1 \leq i \leq n$. 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 order of the variables constrained by a relation is not important; that is, we follow the set-of-mappings formulation of relations (see [34]). The notion of a consistent instantiation of a subset of the variables can be defined in several

[4] Note that all the definitions and algorithms are applicable to relations without the finiteness assumption; a point we make explicit in section 4.2

ways. We use the following definition: an instantiation is consistent if it satisfies all of the constraints that have no uninstantiated variables.

Definition 3 (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. We now introduce the needed operations on constraints adopted from the relational calculus (see [34] for details).

Definition 4 (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 natural join of the two relations. The join of R_{S_1} and R_{S_2} is a relation defined over $S_1 \cup S_2$ containing all the tuples t, satisfying $t[S_1] \in R_{S_1}$ and $t[S_2] \in R_{S_2}$.

Two properties of constraint networks that arise later in the paper are domain size and row convexity.

Definition 5 (k-valued domains). A network of constraints is k-valued if the domain sizes of all variables are bounded by k.

Definition 6 (row convex constraints ([38])). A binary constraint R on a set $\{x_1, x_2\}$ of variables with associated domains D_1 and D_2, is *row convex* if there exists an ordering of D_2 such that, for every $a_1 \in D_1$, the set $\{x_2 \mid (a_1, x_2) \in R\}$ can be ordered such that the elements appear consecutively in the ordering of D_2. An r-ary relation R on a set S of variables $\{x_1, \ldots, x_r\}$ is *row convex* 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. A constraint network is row convex if all its constraints are row convex.

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$\}$, $1 \leq i \leq 4$, and the relations are given by,

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

where $S_1 = \{x_1, x_2, x_3\}$, $S_2 = \{x_2, x_4\}$, and $S_3 = \{x_3, x_4\}$. The projection of R_{S_1} over $\{x_1, x_3\}$, is given by

$$\Pi_{\{x_1,x_3\}} R_{S_1} = \{(\text{a,a}), (\text{a,c}), (\text{a,b}), (\text{b,c}), (\text{b,b}), (\text{b,a}), (\text{c,b}), (\text{c,a}), (\text{c,c})\}.$$

The join of R_{S_2} and R_{S_3} is given by:

$$R_Y = R_{S_2} \bowtie R_{S_3} = \{(\text{a,a,b}), (\text{a,b,b}), (\text{a,c,b}), (\text{b,c,a}), (\text{b,a,c}), (\text{c,c,a}), (\text{c,a,c})\},$$

where $Y = \{x_2, x_3, x_4\}$. The set of all solutions of the network is given by,

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

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,

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

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 [29] defines three properties of networks that characterize local consistency of networks: *node*, *arc*, and *path consistency.* Freuder [19] generalizes this to k-consistency.

Definition 7 (k-consistency; Freuder [19, 20]). 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 the variables without backtracking [9]. 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 8 (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; that is, the network is strong n-consistent relative to the given ordering. An algorithm *globally solves* a problem if it generates a globally solved network.

A globally solved representation is a useful representation of all solutions whenever such a representation is more compact than the set of all solutions.

3.2 Relation-based consistency

In [38], 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 [37] we generalized relational arc- and path-consistency to *relational m-consistency* and used it to specify conditions under which tight binary constraints are globally consistent. In the definition of *relational-consistency*, the relations rather than the variables are the primitive entities. In particular, 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 augment it with the option of having also an explicit reference to a constraint's arity, to allow polynomial algorithms for enforcing those conditions.

Definition 9 (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 relations in $\mathcal{R}$, where $S, T \subseteq X$. We say that R_S is *relationally arc-consistent* relative to a subset of variables $A \subseteq S$ iff any consistent instantiation of the variables in A has an extension to a full tuple in R_S; that is, iff

$$\rho(A) \subseteq \Pi_A(R_S).$$

(Recall that $\rho(A)$ is the set of all consistent instantiations of the variables in A.) A relation R_S is *relationally arc-consistent* if it is relationally arc-consistent relative to every subset $A \subseteq S$. 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 a subset of variables $A \subseteq S \cup T$ iff any consistent instantiation of the variables in A has an extension to the variables in $S \cup T$ that satisfies R_S and R_T simultaneously; that is, iff

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

A pair of relations R_S and R_T is *relationally path-consistent* iff it is relationally path-consistent relative to every subset $A \subseteq S \cup T$. A network is relationally path-consistent iff every pair of relations is relationally path-consistent.

Definition 10 (relational m-consistency). Let $\mathcal{R}$ be a constraint network over a set of variables X, and let $R_{S_1}, \ldots, R_{S_m}$ be m distinct relations in $\mathcal{R}$, where $S_i \subseteq X$. We say that $R_{S_1}, \ldots, R_{S_m}$ are *relationally m-consistent relative to a subset $A \subseteq \bigcup_{i=1}^{m} S_i$* iff any consistent instantiation of the variables in A, has an extension to $\bigcup_{i=1}^{m} S_i$ that satisfies $R_{S_1}, \ldots, R_{S_m}$ simultaneously; that is, if and only if

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

A set of relations $\{R_{S_1}, \ldots, R_{S_m}\}$ is *relationally m-consistent* iff it is relationally m-consistent relative to every $A \subseteq \bigcup_{i=1}^{m} S_i$. A network is relationally m-consistent iff every set of m relations is relationally m-consistent.

Note that if a network is relationally m-consistent then it is also relationally m'-consistent for every $m' \leq m$. Relational arc- and path-consistency correspond to relational 1- and 2-consistency, respectively.

We next refine the definition of relational consistency to be restricted to subsets of bounded size. This restriction is similar to the original restriction used for variable-based local consistency. In *relational (i,m)-consistency* defined below, m always indexes the cardinality of a set of relations and i corresponds to the constraint's arity tested for local consistency.

Definition 11 (relational (i,m)-consistency). A set of relations $\{R_{S_1}, \ldots, R_{S_m}\}$ is *relationally (i,m)-consistent* iff it is relationally m-consistent relative to every subset A *of size* i, $A \subseteq \bigcup_{i=1}^{m} S_i$. A network is relationally (i,m)-consistent iff every set of m relations is relationally (i,m)-consistent. A network is strong relational (i,m)-consistent iff it is relational (j,m)-consistent for every $j \leq i$. Strong relational (n,m)-consistency is identical to relational m-consistency.

The relational based definition of arc-consistency given in [30] is identical to relational (1,1)-consistency.

Definition 12 (directional relational consistency). Given an ordering of the variables, $o = x_1, \ldots, x_n$, a network is *m-directionally relationally consistent* iff for every l, every subset of relations $\{R_{S_1}, \ldots, R_{S_m}\}$ whose largest index variable is x_l, and for every subset $A \subseteq \{x_1, \ldots, x_{l-1}\}$, every consistent assignment to A can be extended to x_l while satisfying all the relevant constraints in $\{R_{S_1}, \ldots, R_{S_m}\}$ simultaneously. Directional relational (resp., strong) (i,m)-consistency is defined accordingly, by restricting the cardinality of A to i.

Revisiting the definition of a globally solved problem:

Definition 13 (globally solved). A problem is globally solved iff there is a known ordering along which the problem is e-directionally relationally consistent, where e is the maximum number of constraints.

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 $D_i = \{\text{a,b,c}\}$, $1 \leq i \leq 4$, 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 $R_{2,3,4,5}$. It is not even (1, 1)-consistent since the value $x_2 = b$ is consistent, but is not consistent relative to $R_{2,3,4,5}$. 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 $R_1 = R_5 = \{\text{b}\}$ (namely, if we enforce (1,2)-consistency) the set of solutions of the network does not change, and the network is both relationally arc- and path-consistent. The reason being that all the variables' domains have a singleton value and therefore, the set of solutions over every subset of variables will contain a single tuple only; the one that is extended to a full solution.

By definition, relational k-consistency implies relational (i, k)-consistency for $i \leq k - 1$, which, for binary constraints, implies strong variable-based k-consistency. The virtue in the relational definition (relative, for instance, to the one based on the dual graph [22]), is that it is easy to work with; it can be incorporated naturally into algorithms for enforcing desired levels of relational consistency.

Below we present algorithm Relational-Consistency or $RC_{(i,m)}$, a brute-force algorithm for enforcing relational (i, 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 denotes the universal relation over A. Algorithm RC_m is the unbounded version of algorithm $RC_{(i,m)}$ in which the recorded constraints arity is not restricted.

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

1. **repeat**
2. $\quad Q \leftarrow \mathcal{R}$
3. $\quad$ **for** every m relations $R_{S_1}, \ldots, R_{S_m} \in Q$
4. $\quad$ and every subset A_i of size i, $A_i \subseteq \bigcup_{j=1}^{m} S_j$ **do**
5. $\quad\quad R_{A_i} \leftarrow R_{A_i} \cap \Pi_{A_i}(\bowtie_{j=1}^{m} R_{S_j})$
6. $\quad\quad$ **if** R_{A_i} is the empty relation
7. $\quad\quad\quad$ **then** exit and return the empty network
8. **until** $Q = \mathcal{R}$

We call the operation in Step 5 *extended composition*, since it generalizes the composition operation defined on binary relations.

Definition 14 (extended composition). The extended composition of relation R_{S_1}, $\ldots$, R_{S_m} relative to a subset of variables $A \subseteq \bigcup_{i=1}^{m} S_i$, denoted

$EC_A(R_{S_1}, \ldots, R_{S_m})$ is defined by:

$$EC_A(R_{S_1}, \ldots, R_{S_m}) = \Pi_A \bowtie_{i=1}^{m} R_{S_i}$$

When the operator is applied to m relations, it is called extended m-composition. If the projection operation is restricted to a set of size i, it is called extended (i, m)-composition.

Algorithm $RC_{(i,m)}$ computes the closure of $\mathcal{R}$ with respect to extended (i, m)-composition. Its complexity is $O(exp(i \cdot m))$ (see Theorem 16) which is clearly computationally expensive for large i and m though it can be improved in a manner parallel to the improvements of path-consistency algorithms [31].

As with variable-based local-consistency, we can improve the efficiency of enforcing relational consistency by enforcing it only along a certain direction. In Figure 1 we present two versions of algorithm DIRECTIONAL-RELATIONAL-CONSISTENCY, $DRC_{(i,m)}$, (DRC_m, respectively) which enforces directional relational (i, m)-consistency (m-consistency, respectively) 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 *(i,m)-directional extension* (*m-directional extension*, respectively) of $\mathcal{R}$, denoted $E_{(i,m)}(R)$ ($E_m(R)$, respectively). Given an ordering of the variables, the algorithm partitions the relations into buckets. In the bucket of x_j it places all the relations whose largest indexed variable is x_j. Buckets are subsequently processed in descending order, and each is closed under the extended (i, m)-composition relative to subsets that exclude the bucket's variable. The resulting relations are placed in lower buckets. Since the operation of extended composition computes constraints that eliminate certain variables it is often called an *elimination operator*. Indeed, as we discuss later, algorithm DRC_m belongs to the class of variable elimination algorithms.

In addition to the main operation of extended composition we propose two optional steps of *simplification* and *instantiation*. These steps are targeted to provide a more efficient implementation and allow the identification of some tractable classes. The simplification step ensures that each bucket contains relations defined on distinct subsets of variables that are not included in each other. The instantiation step exploits the property that whenever one of the relations in the bucket is a singleton tuple we need not perform the full extended m-composition. Instead we can restrict each relation to those tuples that are consistent with the singleton tuple and move each restricted relation to its appropriate buckets. This is equivalent to applying extended 2-composition between each relation and the singleton relation. This special case-handling for instantiation exploits the computational effect of *conditioning* as described in [8, 15].

In step 7 of the algorithm, if the size of $\bigcup_{t=1}^{j} S_t - \{x_p\}$ is smaller than i, we apply the operation to $A = \bigcup_{t=1}^{j} S_t - \{x_p\}$. In step 8, if more than one relation is recorded on the same subset of variables, a subsequent simplification step will combine all such relations into one. Algorithm DRC_m is identical to $DRC_{(i,m)}$ except that constraints are recorded on *all* the variables in the bucket excluding

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

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

Fig. 1. Algorithm $DRC_{i,m}$

x_p; that is, step 7 is modified to:

7a. **for** $A \leftarrow \bigcup_{t=1}^{j} S_t - \{x_p\}$ **do**

Theorem 15. *Let $\mathcal{R}$ be a network processed by $DRC_{(i,m)}$ (DRC_m, respectively) along ordering o, then the directional extension $E_{(i,m)}(\mathcal{R})$ ($E_m(\mathcal{R})$, respectively) is directionally relationally (i,m)-consistent (m-consistent, respectively) relative to o.*

Proof. Clear.

Theorem 16. *The complexity of $DRC_{(i,m)}$ is $O(n^i(k \cdot n)^{(im)})$ where k bounds the domain sizes and n is the number of variables. The complexity of $RC_{(i,m)}$ is $O((n \cdot k)^i \cdot n^i(k \cdot n)^{(im)})$*

Proof. The main step of the algorithm (lines 5-8) relates to the processing of a bucket. The number of relations in each bucket is bounded by $e + n^i$ where e

is size of the initial set of relations and $O(n^i)$ bounds the number of possible relations of arity i out of n variables. The new relations are of size k^i at the most since they are recorded on at most i variables only. The number of subsets of size m out of n^i relations (assuming $e \leq O(n^i)$) is $O(n^{im})$. Performing an m-way join when each relation is of size at most k^i takes $O(k^{im})$, leading to an overall complexity of $O((n \cdot k)^{im})$. Applying a projection over all subsets of size i (step 8) adds a factor of n^i leading to an overall bound of $O(n^i(n \cdot k)^{im})$. The complexity of $RC_{(i,m)}$ can be derived similarly. One loop of the algorithm (steps 2-7) may require $O(n^i(n \cdot k)^{im})$ using a similar analysis. Since the number of loops is bounded by the total number of tuples that can be removed in all the i-ary constraints, which is $O(n^i k^i)$, the result follows. □

The complexity of the nonrestricted version of the algorithms, DRC_m, is not likely to be polynomial even for $m = 2$ since, as we will see, it can solve NP-complete problems. 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 [11], also known as *tree-width* [1, 2].

Definition 17 (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 chordal graph[5]. This is normally accomplished by picking a variable ordering $o = x_1, ..., x_n$, then, moving from x_n to x_1, recursively connecting all the neighbors of x_i that precede it in the ordering. The *induced width* (or *tree width*) of this ordered graph, denoted $w^*(o)$, is the maximal number of earlier neighbors in the resulting graph of each node. The maximal cliques in the newly generated chordal graph form a *clique-tree* and may serve as the subproblems in a procedure called *tree-clustering* [12]. The size of the smallest induced width over all the graph's clique-tree embeddings is the *induced width*, w^* of the graph.

It is known that finding the induced width of a graph is NP-complete [2], nevertheless every ordering of the variables o, yields a simple to compute upper bound denoted $w^*(o)$ (see [12]). The complexity of DRC_m along o can be bounded as a function of $w^*(o)$ of its constraint graph. Specifically [12],

Theorem 18. *The time complexity and size of the network generated by DRC_m along ordering o is $O(nm \cdot (2^m k^3)^{(w^*(o)+1)})$. In particular, the time complexity of DRC_2 is $O((4k^3)^{(w^*(o)+1)})$.*

Proof. Observe that the number of variables mentioned in any bucket is at most $w^*(o)+1$, and thus the number of relations in a bucket is bounded by $O(2^{w^*(o)+1})$ and the number of subsets of size m is $O(2^{(w^*(o)+1)m})$. Also, the number of tuples in each relation is bounded by $k^{w^*(o)+1}$. The complexity of an m-way join of relations of size $k^{w^*(o)+1}$ can be bounded by $O(m \cdot k^{2(w^*(o)+1)})$ since the size

[5] A graph is chordal if every cycle of size 4 or more has a chord.

of the relation resulting from every pair-wise join is still bounded by $k^{w^*(o)+1}$ and thus, m consecutive joins do not multiply but only add. Projection adds a factor of $k^{w^*(o)+1}$. Consequently, the overall complexity is $O((nm) \cdot 2^{(w^*(o)+1)m} \cdot k^{3(w^*(o)+1)})$ which equals the claim. □

The only case for which DRC_m is tractable occurs when $m = 1$.

Lemma 19. *The complexity of DRC_1 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.*

Proof. We have n buckets to process. Each bucket will not contain more then e relations, at any time. The reason is that extended 1-composition involves projections and intersections only, which add only a linear number of constraints and which takes $O(t \cdot e)$ steps. Simplification of a bucket takes $O(e^2 \cdot t^2)$ yielding the result. □

Example 3. Crossword puzzles have been used experimentally in evaluating backtracking algorithms for solving constraint networks [21]. We use an example puzzle to illustrate algorithm DRC_2 (see Figure 2). 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,

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

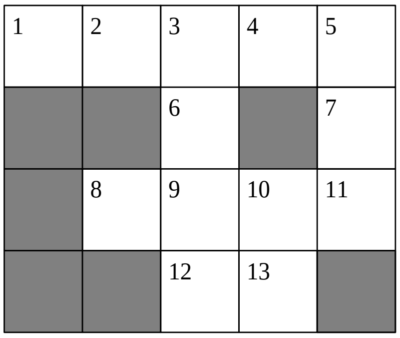

Fig. 2. A crossword puzzle

Let us perform a few iterations of DIRECTIONAL-RELATIONAL-CONSISTENCY, with m equal to 2 and o as the ordering of the variables $x_{13}, x_{12}, \ldots, x_1$. Thus, x_1 is the highest variable in the ordering and x_{13} is the lowest. The bucket for x_1 contains the single relation $R_{1,2,3,4,5}$. Processing bucket$_1$ adds the relation,

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

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,

$$R_{3,4,5} = \Pi_{3,4,5}(R_{2,3,4,5}) = \{(\text{S,E,S}), (\text{S,E,R}), (\text{E,E,T}), (\text{A,I,L}), (\text{E,E,R})\},$$

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

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

to the bucket of variable x_4. The bucket for x_4 now contains the relation $R_{4,5,6,9,12}$. Processing $bucket_4$ adds the relation,

$$R_{5,6,9,12} = \{(\text{S,A,M,E}), (\text{R,A,M,E}), (\text{T,A,R,N}), (\text{L,R,O,N}), (\text{R,A,R,N})\}.$$

The bucket for x_5 contains now the relations $R_{5,6,9,12}$, and $R_{5,7,11}$. Processing bucket$_5$ adds the relation,

$$R_{6,7,9,11,12} = \{(\text{A,E,R,N,N}), (\text{A,U,M,N,E}), (\text{A,U,R,N,N}), (\text{R,E,O,T,N})\}.$$

The bucket for x_6 contains only the newly generated relation $R_{6,7,9,11,12}$. Processing bucket$_6$ adds the relation,

$$R_{7,9,11,12} = \{(\text{E,R,N,N}), (\text{U,M,N,E}), (\text{U,R,N,N}), (\text{E,O,T,N})\},$$

to the bucket for x_7. Processing bucket$_7$ adds the relation,

$$R_{9,11,12} = \{(\text{R,N,N}), (\text{M,N,E}), (\text{O,T,N})\},$$

to the bucket of x_9. The bucket of x_8 contains only the original relation $R_{8,9,10,11}$, and when processed it adds the relation,

$$R_{9,10,11} = \{(\text{I,K,E}), (\text{R,O,N}), (\text{E,E,T}), (\text{A,R,N}), (\text{A,M,E})\}.$$

The bucket for x_9 contains the relations $R_{9,10,11}$, $R_{9,11,12}$. Processing bucket$_9$ adds the relation,

$$R_{10,11,12} = \{(\text{O,N,N})\}.$$

The bucket for x_{10} contains the relations $R_{10,11,12}$, and $R_{10,13}$. Processing bucket$_{10}$ adds the empty relation. Since the empty relation was derived, the algorithm stops and reports that the network is inconsistent.

Finally, we propose algorithm ADAPTIVE-RELATIONAL-CONSISTENCY (ARC) which is the relational counter-part of algorithm adaptive-consistency [11]. Like algorithm DRC_m, it processes the buckets in order from last to first. When processing the bucket of x_j, it applies extended composition relative to *all* the relations in that bucket, and with respect to the whole set of variables appearing in the bucket excluding x_j. It then places the resulting relation in its appropriate bucket. Algorithm ARC^i is a restricted version of ARC that records relations of arity i only. It is identical to ARC accept that step 4 and 5 are modified to record constraints on subsets of size i at the most. Algorithm ARC generates a globally-solved problem and it can be viewed as a compilation algorithm since it yields a representation from which the set of solutions can be recovered in linear time. It is identical to DRC_m when m is not bounded. For brevity, we omit the full steps of simplification and instantiations.

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

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 **do**
3.   simplify,
4.   instantiate,
5.   **for** all the relations $R_{S_1}, \ldots, R_{S_j}$ in $bucket_p$ **do**
6. $A \leftarrow \bigcup_{i=1}^{j} S_i - \{x_p\}$
7. $R_A \leftarrow R_A \cap \Pi_A(\Join_{i=1}^{j} R_{S_i})$
8.   **if** R_A is not the empty relation **then**
9.   add R_A to its appropriate bucket
10.   **else** exit and return the empty network
11. **return** $E_o(R) = bucket_1 \cup bucket_2 \cup \cdots \cup bucket_n$

Theorem 20. *Algorithm* ADAPTIVE-RELATIONAL-CONSISTENCY *(ARC) globally solves any constraint network. The complexity of the algorithm when processed along ordering o is bounded by $O(n \cdot (2k^3)^{w^*(o)+1})$.*

Proof. The algorithm is clearly generating a backtrack-free representation. The number of relations in each bucket will increase to at most $2^{w^*(o)+1}$ relations. The arity of each relation is bounded by $w^*(o)+1$ and thus its size is bounded by $O(k^{w^*(o)+1})$. Consequently the overall complexity is bounded by the cost of joining at most $O(2^{w^*(o)+1})$ relations of size $O(k^{w^*(o)+1})$ which, when adding a factor of $O(k^{w^*(o)+1})$ for projection, can be bounded by $O(n \cdot 2^{w^*(o)+1} \cdot k^{3(w^*(o)+1)})$. □

Finally we can show that some NP-complete problems are solved by DRC_2.

Theorem 21. *Crossword puzzles can be globally solved by DRC_2 in any variable ordering and its complexity is $O(n \cdot k^{3(w^*(o)+1)})$.*

Proof. Let $\mathcal{R}$ be a crossword puzzle instance. We will show that the buckets of the network generated by ARC have at most two relations. Therefore, for such problems ARC reduces to DRC_2. Since ARC generates a backtrack-free problem it follows that so will DRC_2. We will now prove that there are at most two relations in each bucket of the crossword puzzle at any time during processing by ARC. Let us annotate each variable in a constraint by a $+$ if it appears in a horizontal word and by a $-$ if it appears in a vertical word. Clearly, in the initial specification each variable appears in at most two constraints and each annotated variable appears in just one constraint (with that annotation). We show that this property is maintained throughout the algorithm's performance. It could be the case that the two annotated variables will appear in the same constraint. The annotation of the variables in the constraint resulting from extended 2-composition inherits the annotation in the parent constraints. If a variable appeared with annotation "+" in one, and annotation "−" in the other, its annotation in the resulting constraint will be "+,−". The claim can be proved by induction on the processed buckets. Assume that after processing buckets $x_n, \ldots, x_i$ *all* the constraints appearing in the union of all the buckets from $bucket_{i-1}$ to $bucket_1$, satisfy that each annotated variable appears in at most one constraint. When processing $bucket_{i-1}$, since it contains only two constraints (otherwise it will contain multiple annotations of variable x_{i-1}), it generates a single new constraint. Assume that the constraint is added to the bucket of x_j. Clearly, if x_j is annotated positively (respectively negatively) in the added constraint, $bucket_j$ cannot contain already a constraint with a positive (respectively, negative) annotation of x_j. Otherwise, it means that before processing bucket $i-1$, there were two constraints with positive (respectively negative) annotation of x_j, one in the bucket of x_{i-1} and one in the bucket of x_j, which contradicts the induction hypothesis. A very similar argument can be applied to the multiple annotation case. The complexity of DRC_2 for the crossword puzzles is bounded by $O(n \cdot k^{3(w^*(o)+1)})$ thus reducing the base of the exponent by a factor of $2^{w^*(o)}$ relative to the general bound of DRC_2. $\Box$

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.

4.1 Variable elimination in propositional CNF theories

We denote propositional symbols, also called *variables*, by uppercase letters $P, Q, R, \ldots$, propositional literals (i.e., $P, \neg P$) by lowercase letters $p, q, r, \ldots$, and disjunctions of literals, or *clauses*, by $\alpha, \beta, \ldots$. A *unit clause* is a clause of size 1. The notation $(\alpha \vee T)$, when $\alpha = (P \vee Q \vee R)$ 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 $\varphi = \{\alpha_1, \ldots, \alpha_t\}$ 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 2-composition of R_A and R_B relative to $A \cup B - \{Q\}$, $Q \in A \cap B$. It is easy to see that pair-wise resolution is equivalent to extended 2-composition.

Lemma 22. *The* resolution *operation over two clauses* $(\alpha \vee Q)$ *and* $(\beta \vee \neg Q)$, *results in a clause* $(\alpha \vee \beta)$ *satisfying:* $models(\alpha \vee \beta) = EC_{Q'}(models(\alpha), models(\beta))$.

Proof. Clear.

In [35] we have shown that row-convex relations that are closed under extended 2-composition can be globally solved by DRC_2. Observe that any bi-valued relation is row-convex, therefore, since *CNF* theories are bi-valued, DRC_2, if applied to the relational representation of a *CNF* theory, will decide the problem's satisfiability and generate a globally solved representation. From the above lemma, extended 2-decomposition can be applied to the *CNF* representation directly and therefore, transformation to a relational representation can be avoided.

Replacing extended 2-composition by resolution and the instantiation step by unit resolution in DRC_2, results in algorithm *Directional Resolution (denoted DR)* which is the core of the well known Davis Putnam algorithm for satisfiability [7, 14]. Applying the same exchange within $DRC_{(i,2)}$ yields algorithm *bounded directional resolution* (BDR_i) which is a polynomial approximation of DR [14]. As is well known and as also follows from our theory, algorithm directional resolution globally solves any *CNF* theory.

DIRECTIONAL-RESOLUTION (φ, o)

Input: A *CNF* theory φ, an ordering $o = Q_1, \ldots, Q_n$ of its variables.

Output: A decision of whether φ is satisfiable. if it is, a theory $E_o(\varphi)$, equivalent to φ, else an empty directional extension.

1. Initialize: generate an ordered partition of clauses into buckets. $bucket_1$, $\ldots$, $bucket_n$, where $bucket_i$ contains all clauses whose highest literal is Q_i.
2. **for** $i \leftarrow n$ **downto** 1 **do**
3.   **if** there is a unit clause **then**
           apply unit-resolution and place the resolvents in their right bucket
           if the empty clause was generated, theory is not satisfiable
4.   **else** resolve each pair $\{(\alpha \vee Q_i), (\beta \vee \neg Q_i)\} \subseteq bucket_i$.
           if $\gamma = \alpha \vee \beta$ is empty, return $E_o(\varphi) = \{\}$, theory is not satisfiable
           else determine the index of γ and add it to the appropriate bucket.
5. **return** $E_o(\varphi) \leftarrow \bigcup_i bucket_i$

Incorporating resolution into DRC_1 yields algorithm unit propagation. The operation of extended 1-composition in DRC_1 will have no effect since projections on clauses generate universal relations. The only relevant steps are the simplification and instantiation. The simplification step, if included, allows resolution involving non-unit clauses as long as the variables appearing in one clause are contained in the other clause. The instantiations step translates to unit resolution.

As in the general case, DR generates a globally solved representation and its complexity can be bounded exponentially as a function of the induced width w^* of the *CNF* theory. The graph of a *CNF* theory associates propositional symbols with nodes and connects two nodes if their associated symbols appear in the same clause.

4.2 Variable elimination in linear inequalities

In database theory, a k-ary relation r is a finite set of tuples and a database is a finite set of relations. However, the relational calculus and algebra can be developed without the finiteness assumptions for relations. We will use the term unrestricted relation, for finite or infinite sets of points in a k-dimensional space [24]. In particular, it was shown that relational calculus is identical to relational algebra for countable domains and that relational algebra for infinite relations is exactly the same as for finite relations [25][6]. Therefore, the relational framework we have presented applies as is to infinite relations. In this section we will demonstrate the applicability of our results to the special case of linear inequalities over infinite domains like the Rationals as well as over finite and infinite subsets of the Integers.

Let us consider the class of linear inequalities where a constraint between r variables or less is 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 rational constants. For example, the conjunction $(3x_i + 2x_j \leq 3) \wedge (-4x_i + 5x_j \leq 1)$ is an allowed constraint between variables x_i and x_j. A network with constraints of this form can be formulated as a linear program where the domains are infinite Rational, or Integers, or finite subsets of integers restricted by unary linear inequalities. We will show first that over the Rationals the standard operation of variable elimination is equivalent to extended 2-composition while this equivalence is not maintained over the integers. Let us denote by $sol(\alpha)$ the unrestricted relation of tuples from the domain satisfying a set of linear inequalities, α. We define the elimination operation as follows:

Definition 23 (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 24. *$sol(elim_r(\alpha, \beta)) \supseteq EC_r(sol(\alpha), sol(\beta))$ when the domains are the Integers. However, over the Rationals $sol(elim_r(\alpha, \beta)) = EC_r(sol(\alpha), sol(\beta))$.*

[6] We thank Manolis Koubarakis for pointing to us the extension to infinite domains.

Proof. It is easy to see that if a_r and b_r have the same sign (both are positive or both are negative), then for any assignment to $x_1, \ldots, x_{i-1}$ there is always a value for x_r that extends $x_1, \ldots, x_{i-1}$ and that satisfies both α and β. Therefore, the extended composition produces the universal relation. Assume now that a_r and b_r have 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 *rational value* of x_r in a way that satisfies both α and β. It is unclear, though, that there exists an integer extension to x_r which is the reason for partial containment for the integers. □

In [35] we have shown that linear inequality constraints over finite sets of integers are row-convex and therefore can be globally solved by DRC_2 using their relational form. The definition of row-convexity can be extended to infinite domains without any modification. This implies that linear inequalities over the Rationals that are relationally 2-consistent are globally solved and consequently linear inequalities can be globally solved by DRC_2.

Incorporating linear elimination into DRC_2 (when the constraints are presented as linear inequalities) results in algorithm *Directional Linear Elimination* (abbreviated DLE) which is the well known Fourier elimination algorithm (see [28]). Indeed, as dictated by our theory and as is already known the algorithm decides the solvability of any set of linear inequalities over the Rationals.

DIRECTIONAL-LINEAR-ELIMINATION (φ, o)

Input: A *set of linear inequalities* φ, an ordering $o = x_1, \ldots, x_n$ of its variables.

Output: A decision of whether φ is satisfiable. **if** it is, a theory $E_o(\varphi)$, equivalent to φ, else an empty directional extension.

1. Initialize: generate an ordered partition of the inequalities into buckets.
2. **for** $i \leftarrow n$ **downto** 1 **do**
3.   **if** x_i has one value in its domain **then**
4.   substitute the value into each inequality in the bucket and put the resulting inequality in the right bucket.
4.   **else** for each pair $\{\alpha, \beta\} \subseteq bucket_i$, compute $\gamma = elim_i(\alpha, \beta)$
if γ has no solutions, return $E_o(\varphi) = \{\}$, theory is not satisfiable
else add γ to the appropriate bucket.
5. **return** $E_o(\varphi) \leftarrow \bigcup_i bucket_i$

Example 4.

$$\varphi(x_1, x_2, x_3, x_4) = \{(1)\ 5x_4 + 3x_2 - x_1 \leq 5,\ (2)\ x_4 + x_1 \leq 2,\ (3)\ -x_4 \leq 0,$$
$$(4)\ x_3 \leq 5,\ (5)\ x_1 + x_2 - x_3 \leq -10,\ (6)\ x_1 + 2x_2 \leq 0\}.$$

Initially, $bucket_4 = \{5x_4 + 3x_2 - x_1 \leq 5,\ x_4 + x_1 \leq 2,\ -x_4 \leq 0\}$, $bucket_3 = \{\ x_3 \leq 5,\ x_1 + x_2 - x_3 \leq -10\}$ and $bucket_2 = \{x_1 + 2x_2 \leq 0\}$. Processing $bucket_4$, applying elimination relative to x_4 over inequalities (1) (3), and (2) (3), respectively, results in: $3x_2 - x_1 \leq 5$, placed into $bucket_2$, and $x_1 \leq 2$, placed into $bucket_1$. Processing $bucket_3$ next, eliminates x_3 from (4) and (5), yielding $x_1 + x_2 \leq -5$, placed into $bucket_2$ and processing $bucket_2$ adds no new inequality. We can now generate a backtrack-free solution in the following way. Select a value for x_1 from its domains satisfying the unary inequalities in $bucket_1$. After selecting assignments to $x_1, \ldots, x_{i-1}$ select a value for x_i satisfying all the inequalities in $bucket_i$. This is easy since all the constraints are unary once the values of $x_1, \ldots, x_{i-1}$ are determined.

Theorem 25. *DLE (Fourier elimination) globally solves a set of linear inequalities over the Rationals*[7]*.*

Proof. It is known that the Fourier elimination algorithm decides the consistency of a set of linear inequalities over the rationals. Since DRC_2 globally solves a set of row-convex constraints, since linear inequalities are row-convex and are closed under extended 2-composition, and since DLE is equivalent to DRC_2, the claim follows. □

Linear inequalities over the integers: When the domains are the Integers the algorithm is no longer guaranteed to decide consistency since linear elimination is not identical to extended 2-composition. If the empty relation is generated by DLE, the problem is indeed inconsistent, else, the problem may or may not be consistent. Nevertheless, the representation generated by DLE could be useful since it is a backtrack-free representation relative to the rationals, of a superset of the sought-for integer solutions. From such a representation an integer solution may be extracted using backtrack search that may enjoy a substantial amount of pruning.

Complexity of DLE Algorithm DLE is generally exponential since it may record an exponential number of inequalities. If the domains are finite, the finite relational representation can be used (in which case $DLE = DRC_2$), and in this case the complexity can be bounded using the notion of induced width. Otherwise, DLE's complexity may be worst-case exponential even when the induced width w^*, is bounded. The reason is that an exponential number of inequalities may need to be recorded on the same subset of variables. One cannot "intersect" two inequalities and replace them by one. In other words, linear inequalities are not closed under intersection while relations are.

[7] The result holds also for the Reals, however since relational algebra was extended for countable domains only it does not follow from the general theory and needs to be proved directly.

Case of binary inequalities: When the linear inequalities are over pairs of variables only, algorithm DLE, as presented here is still exponential. However it was shown to have a polynomial implementation over the Rationals that uses a special data structure that bounds the number of inequalities over any pair of variables and leads to a polynomial algorithm [18]. Over the integers the binary linear problem is NP-complete [27]. A more restricted case of binary monotone inequalities of the form $ax_i - bx_j \leq c$, where a, b, c positive integers, was shown to be weakly NP-complete since there exists a pseudo-polynomial algorithm [18]. A polynomial algorithm that globally solves the problem over the rationals is given in [17]. For bounded integer domains the general binary linear problem can be expressed in a relational form and since DRC_2 is polynomial over binary constraints, the class can be solved in polynomial time relative to the maximal range of the integer domains. In summary,

Theorem 26. *Algorithm DLE is exponential even for binary inequalities and even for bounded induced width. For finite domains DRC_2 is applicable. Its complexity for binary constraints is polynomial (in the input and the maximum domain range), and is exponentially bounded by the induced width, for non-binary constraints.* □

There are additional special classes for which DLE is polynomial. One case is the class of *simple temporal constraints*. Those are unary and binary constraints of the form $X - Y \leq a$. Algorithm DLE reduces, in this case, to the shortest path algorithm presented in [10]. The algorithm is polynomial since the number of inequalities produced is bounded (in this simple case at most two inequalities are needed between any pair of variables) and since the class is closed under linear elimination. The linear elimination operator over the integers, coincides with extended 2-composition in this case, and therefore, DLE is complete for simple temporal constraints over the integers as well. Note, that although this is a subclass of monotone inequalities, tractability of DLE over this class does not follow from [18] whereby a special implementation was required.

Theorem 27. *Algorithm DLE is polynomial over the class of unary and binary inequalities of the form $X - Y \leq a$, $X \leq b$. The algorithm globally solves such inequalities, over the Integers (if a and b are integers), the Rationals and the Reals.*

Proof. Over the Integers and the Rationals, global consistency follows from the global consistency of DRC_2. In this case DLE is complete since simple temporal inequalities over the integers are closed under extended 2-composition and intersection. the proof is given in [10].

Case of zero-diversity theories: Propositional *CNF*s as well as linear inequalities share an interesting syntactic property: It is easy to recognize whether applying extended 2-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 [14] 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 [20] 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. In particular binary trees can be solved by arc-consistency [31]. Dechter and Pearl [11] provide an adaptive scheme where the level of local consistency is adjusted on a node-by-node basis. Dechter and Pearl [12] generalize the results on trees to hyper-trees which are called acyclic in the database community [4].

For work on identifying properties of the constraints, Montanari [33] shows that path consistency is sufficient to guarantee that a binary network is globally consistent if the relations are monotone. Dechter [9] 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 constraints in the network [37, 36]. Van Hentenryck, Deville, and Teng [39] 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 implicational constraints [26, 23] and to row-convexity [38].

Finally, for work that falls into both classes, Dechter and Pearl [13] 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.

Most of these relationships were formulated initially using the variable-based definition of local-consistency. Reference to constraints was indirect via the constraint's arity as a parameter. Recently, we have shown that these relationships can be generalized using relational consistency and that they lead to a characterization of classes of problems that can be solved by a restricted level m of DRC_m. The general pattern is as follows. We present a sufficient condition showing that a network satisfying a property p, and having a corresponding level of relational 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. We have recently demonstrated the use of our definition for two properties: row-convexity and tightness. We have shown that,

Theorem 28. *[38]. A networks of relations that are row-convex and are relational 2-consistent are globally consistent.*

Consequently, a network of row-convex relations that are closed under extended 2-composition can be globally-solved by DRC_2. Similarly, we have shown that:

Theorem 29. *[37] If a network of constraints is m-tight and $m+2$-relational consistent, than it is globally consistent.*

Consequently, whenever a set of m-tight relations is closed under extended $m+2$-composition it can be solved by DRC_{m+2}. The notion of m-tightness is defined as follows. A binary relation is m-tight if every value of one variable is consistent with at most m values of the second variable. A general relation is m-tight if every tuple on all the variables excluding one, has an extension in the constraint to the missing variable using at most m values.

In this section we apply the definition of relational consistency to relationships involving properties such as the size of the domains, acyclicity and causality.

5.1 Domain size and global consistency

In [9], we have shown that:

Theorem 30. *[9] 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 algorithm can be stated more coherently.

Theorem 31. *A k-valued constraint network $\mathcal{R}$, that is k-relationally-consistent is globally consistent.*

Proof. We prove the theorem by showing that relationally k-consistent k-valued networks are relationally $(k+i)$-consistent for any $i \geq 1$. According to the definitions, we need to show that, if there are relations $R_{S_1}, \ldots, R_{S_{k+i}}$, all sharing variable x_t, and if $\bar{x}$ is a locally consistent tuple defined over $\bigcup_{i=1}^{k+i} S_i - \{x_t\}$, then there is a value a of x_t such that $(\bar{x}, a)$ belongs to the joined relation $R_{S_1} \bowtie, \ldots, \bowtie R_{S_{k+i}}$. With each value j, in the domain of x_t we associate a subset A_j that contains all those relations in $\{R_{S_1}, \ldots, R_{S_{k+i}}\}$ that are consistent with the assignment $x_t = j$. Since variable x_t may take on k possible values $1, 2, \ldots, k$ we get k such subsets, $A_1, \ldots, A_k$. We claim that there must be at least one set, say A_1, that contains all the constraints $\{R_{S_1}, \ldots, R_{S_{k+i}}\}$. If this were not the case, each subset A_j would be missing some member, say $R'_{S'_j}$, which means that the partial tuple $\bar{x}' = \Pi_A(\bar{x})$, $A = \bigcup_{i=1}^{k} S'_i - \{x_t\}$, is locally consistent, namely it belongs to ρ_A, but it cannot be consistently extended to a value of x_t while satisfying the k relations $R'_{S'_1}, \ldots, R'_{S'_k}$. This leads to a contradiction because as a subset of $\bar{x}$, $\bar{x}'$ is locally consistent, and from the assumption of relational k-consistency, this tuple should be extensible by any additional variable including x_t. $\Box$

Since the domains do not increase by extended k-composition we get:

Theorem 32. *Any k-valued network $\mathcal{R}$ can be globally solved by DRC_k.*

Example 5. From Theorem 32, bi-valued networks can be globally solved by DRC_2. In particular, propositional *CNF*s can be globally solved by DRC_2. As we have seen, in this case, the operator of extended 2-composition takes the form of pair-wise resolution yielding algorithm directional resolution [14].

5.2 Acyclicity, causality and global consistency

Relational consistency and the DRC_m algorithms can also capture the tractable classes of acyclic and causal networks. It is well known that acyclic networks are tractable [32, 12].

Definition 33 (acyclic networks). A network of constraints is acyclic if it has a chordal constraint graph and if each maximal clique is associated with a single constraint.

It is easy to see that:

Lemma 34. *If a network is acyclic then there exists an ordering of the variables for which each bucket has a single relation.*

Causal networks include acyclic networks. They were defined in order to capture the ease of some tasks in physical systems, such as projection.

Definition 35 (causal networks [13]). A constraint network is causal relative to an ordering $o = x_1, ..., x_n$ iff it is globally solved (i.e., backtrack-free).

Definition 36 (causal relations [13]). A constraint is called causal if its projection on any subset of variables generates a universal relation.

Lemma 37. *[13] A single-bucket network relative to ordering o whose constraints are causal, is causal relative to o.*

Finally, it is easy to see that:

Theorem 38. *Single-bucket networks that are closed under DRC_1 are tractable.*

Proof. Since each bucket contains a single relation throughout processing, DRC_1 is equivalent to ARC and therefore, complete. Since DRC_1 is polynomial, the claim follows. □

Since acyclic networks are single-buckets and closed under DRC_1, and since single-bucket causal relations are closed under DRC_1, and, since CNF formulas as well as linear inequalities are causal relations, we conclude:

Corollary 39. *Algorithm DRC_1 is complete for: 1. Acyclic networks, 2. single-bucket causal relations, and in particular for single-bucket CNF's and linear inequalities. For the latter two classes this is a special case of the zero-diversity class.*

6 Discussion

The algorithms we present in this paper belong to the class of variable elimination algorithms, formulated recently within the *bucket elimination* framework [15], which generalize non-serial dynamic programming [5]. We have recently shown how a collection of probabilistic and combinatorial optimization tasks can be formulated within this framework [15]. Such algorithms were also presented for various graph-based tasks by [1, 3]. All the algorithms possess similar properties of compiling a theory into a backtrack-free one (or greedy) and their complexity is dependent on the same graph properties. Specifically they all have a complexity bound which is exponential in the induced-width of some graph.

Another common property often overlooked of all such algorithms, is that they also require space exponential in the induced width. We have recently demonstrated how a method of conditioning can be incorporated into the bucket-elimination scheme to allow trading space for time. The special case-handling of singleton values that we had introduced (i.e., instantiation) permits this extension [16] and will lead to similar time-space tradeoffs.

Since the algorithms may be quite time demanding, unless the problem is very sparse, practical considerations call for the use of approximations. Polynomial approximation algorithms such as $DRC_{(i,m)}$ could be useful and may be extended to optimization and probabilistic inference as well.

7 Conclusions

We focused on a new definition of local consistency called *relational consistency*. This definition is relational-based, in contrast with previous definitions which were variable-based. We presented algorithms, *Directional Relational Consistency (DRC_m)*, enforcing relational consistency using a general composition operator which unifies resolution for CNF theories, variable elimination in linear inequalities and the project-join operator in relational databases. We also show that relational consistency is useful in characterizing relationships between properties of constraint networks and the level of local consistency needed to ensure global consistency.

Specifically, we have shown that different levels of DRC can globally solve different classes of constraint networks:

1. DRC_1 globally solves acyclic and single-bucket, causal relations in polynomial time.
2. DRC_2 globally solves bi-valued domain 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 is exponential otherwise. Algorithm DLE (or Fourier elimination) is a linear elimination algorithm equivalent to DRC_2 over the rationals, and approximates DRC_2 over integers. The resolution algorithm of Davis-Putnam is equivalent to DRC_2.
3. Algorithm DRC_m globally solves m-valued networks. The algorithm is polynomial for binary constraints.
4. Algorithm ARC globally solves all networks.
5. The complexity of both DRC_m and ARC is exponentially bounded by w^*, the induced-width (tree-width) of the network over finite domains.
6. We introduced a class of polynomial directional relational consistency algorithms $DRC_{(i,m)}$ that approximate DRC_m. The algorithms are complete when $i \geq w^*(o)$.

All the algorithms we present belong to the family of variable elimination algorithms that are widely applicable to deterministic reasoning tasks, to optimization problems and to probabilistic inference [16, 15].

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

References

1. S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability — a survey. *BIT*, 25:2–23, 1985.
2. 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.
3. S. Arnborg, and A. Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete and applied Mathematics* 23 (1989) 11-24.
4. C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30:479–513, 1983.
5. U. Bertele and F. Brioschi. *Nonserial Dynamic Programming*, Academic Press, New York, 1972.
6. M. C. Cooper. An optimal k-consistency algorithm. *Artif. Intell.*, 41:89–95, 1989.
7. M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7:201–215, 1960.

8. R. Dechter. Enhancement schemes for constraint processing incorporating, backjumping, learning and cutset decomposition. *Artif. Intell.*, 41:273–312, 1990.
9. R. Dechter. From local to global consistency. *Artif. Intell.*, 55:87–107, 1992.
10. R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artif. Intell.*, 49:61–95, 1991.
11. R. Dechter and J. Pearl. Network-based heuristics for constraint satisfaction problems. *Artif. Intell.*, 34:1–38, 1988.
12. R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artif. Intell.*, 38:353–366, 1989.
13. 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.
14. 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*, 134–145, 1994.
15. R. Dechter. Bucket elimination: a unifying framework for probabilistic inference. In *Proceedings of Uncertainty in Artificial Intelligence (UAI-96)*, 1996.
16. R. Dechter. Topological properties of time-space tradeoff. In *Proceedings of Uncertainty in Artificial Intelligence (UAI-96)*, 1996.
17. D. Q. Goldin and P. C. Kanellakis, Constraint query algebras. In *Constraints* 1,1-41,1996.
18. D. S. Hochbaum and J. Naor, Simple and Fast algorithms for linear integer programs with two variables per inequality. *SIAM J. of Computing* 23:6:1179-1192, 1994.
19. E. C. Freuder. Synthesizing constraint expressions. *Comm. ACM*, 21:958–966, 1978.
20. E. C. Freuder. A sufficient condition for backtrack-free search. *J. ACM*, 29:24–32, 1982.
21. 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.
22. P. Jégou. On the consistency of general constraint satisfaction problems. In *Proc. of the 11th National Conf. on AI*, pages 114–119, 1993.
23. M.C. Cooper, D,A, Cohen and P.G. Jeavons. Characterizing tractable constraints *Artificial Intelligence* 65, 347-361, 1994.
24. P.C. Kanellakis, G.M. Kuper and P. Z. Revesz. Constraint Query languages. Proc. 9th ACM PODS, 299-313, 1990.
25. P. Kanellakis, *Elements of Relational Database Theory*, Handbook of Theoretical Computer Science, Chapter 17, Vol, B, J. van Leeuwen editor, North-Holland, 1990.
26. L. M. Kirousis. Fast parallel constraint satisfaction. *Artif. Intell.*, 64:147–160, 1993.
27. J. C. Lagarias, "The computational complexity of simultaneous Diophantine approximation problems" *SIAM Journal on Computing*, Vol 14, No. 1 (1985), pp. 196-209.
28. J-L Lassez and M. Mahler, "On Fourier's algorithm for linear constraints" *Journal of Automated Reasoning*, Vol 9, 1992.
29. A. K. Mackworth. Consistency in networks of relations. *Artif. Intell.*, 8:99–118, 1977.
30. A. K. Mackworth. On reading sketch maps. *International joint conference on Artificial Intelligence(IJCAI-77)*, Cambridge, Mass. 1977, pp. 587-606.

31. 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.
32. D. Maier. *The Theory of Relational Databases.* Computer Science Press, 1983.
33. U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Inform. Sci.*, 7:95–132, 1974.
34. J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Vol. 1.* Computer Science Press, 1988.
35. P. van Beek. On the minimality and decomposability of constraint networks. In *Proc. of the 10th National Conf. on AI*, pages 447–452, 1992.
36. 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.
37. 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.
38. P. van Beek and R. Dechter. On the minimality and global consistency of row-convex constraint networks. *Journal of the ACM*, 42:543-561, 1995.
39. P. Van Hentenryck, Y. Deville, and C.-M. Teng. A generic arc consistency algorithm and its specializations. *Artif. Intell.*, 57:291–321, 1992.

This article was processed using the LaTeX macro package with LLNCS style