# Constraint Restrictiveness versus Local and Global Consistency

Peter van Beek
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2H1
vanbeek@cs.ualberta.ca

Rina Dechter
Department of Computer and Information Science
University of California, Irvine
Irvine, California, USA 92717
dechter@ics.uci.edu

**Abstract**

Constraint networks are a simple representation and reasoning framework with diverse applications. In this paper, we identify two new complementary properties on the restrictiveness of the constraints in a network—*constraint tightness* and *constraint looseness*—and we show their usefulness for estimating the level of local consistency needed to ensure global consistency, and for estimating the level of local consistency present in a network. In particular, we present a sufficient condition, based on constraint tightness and the level of local consistency, that guarantees that a solution can be found in a backtrack-free manner. The condition can be useful in applications where a knowledge base will be queried over and over and the preprocessing costs can be amortized over many queries. We also present a sufficient condition for local consistency, based on constraint looseness, that is straightforward and inexpensive to determine. The condition can be used to estimate the level of local consistency of a network. This in turn can be used in deciding whether it would be useful to preprocess the network before a backtracking search, and in deciding which local consistency conditions, if any, still need to be enforced if we want to ensure that a solution can be found in a backtrack-free manner. Two definitions of local consistency are employed in characterizing the conditions: the traditional variable-based notion and a new definition of local consistency called *relational* consistency. New algorithms for enforcing relational consistency are introduced and analyzed[1].

[1]Some of the results reported in this paper have previously appeared at KR-94 [32] and AAAI-94 [31].

# Contents



# Notation

| | |
|---|---|
| $\mathcal{R}$ | a constraint network |
| $x_1, \ldots, x_n$ | the variables in a constraint network |
| $X$ | the set of all variables in a constraint network |
| $D_1, \ldots, D_n$ | the domains of the variables in a constraint network |
| $S_1, \ldots, S_t$ | the subsets of variables over which the relations are defined |
| $R$ | a relation (possibly subscripted with the set of variables over which the relation is defined or the indices of the variables) |
| $Y$ | a subset of the variables in a constraint network |
| $Y_I$ | an instantiation of the variables in the set of variables $Y$ |
| $(a_1, \ldots, a_n)$ | an instantiation of the variables $\{x_1, \ldots, x_n\}$, i.e., an assignment of $a_i \in D_i$ to $x_i$ |
| $n$ | the number of variables in a constraint network |
| $d$ | the size of the largest domain in a constraint network |
| $o$ | an ordering of the variables |
| $Y_I[S]$ | the tuple consisting of only the components of $Y_I$ that correspond to the variables in $S$ |
| $\rho(Y)$ | the set of all consistent instantiations of the variables in $Y$ |
| $\sigma_{Y_I}(R)$ | the selection of those tuples in $R$ that agree with $Y_I$ |
| $\Pi_Y(R)$ | the projection of the relation $R$ on the subset $Y$ |
| $\bowtie$ | the join operator of the relational database model |

# 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 spite of the simplicity of the framework, many interesting problems can be formulated as constraint networks, including graph coloring [25], scene labeling [34], natural language parsing [23], and temporal reasoning [1].

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 [16, 21, 25] and to the development of algorithms for enforcing local consistency conditions by removing local inconsistencies (e.g., [6, 11, 14, 21, 25, 34]).

Local consistency has proven to be an important concept in the theory and practice of constraint networks for primarily two reasons. First, a common method for finding solutions to a constraint network is to first preprocess the network by enforcing local consistency conditions, and then perform a backtracking search. The preprocessing step can reduce the number of dead ends reached by the backtracking algorithm in the search for a solution. With a similar aim, local consistency techniques can be interleaved with backtracking search. The effectiveness of using local consistency techniques in these two ways has been studied empirically (e.g., [10, 7, 17, 18, 19, 28]). Second, much previous work has identified conditions for when a certain level of local consistency is sufficient to guarantee a network is globally consistent or to guarantee a solution can be found in a backtrack-free manner (e.g., [9, 11, 15, 16, 25, 33]).

In this paper, we identify two new complementary properties on the restrictiveness of the constraints in a network—*constraint tightness* and *constraint looseness*—and we show their usefulness for estimating the level of local consistency needed to ensure global consistency, and for estimating the level of local consistency present in a network. In particular, we present the following results.

We present a relationship between the tightness of the constraints, the arity of the constraints, and the level of local consistency sufficient to ensure global consistency. Specifically, in any constraint network where the constraints have arity $r$ or less and the constraints have tightness of $m$ or less, if the network is strongly $((m+1)(r-1)+1)$-consistent, then the network is globally consistent. Informally, a constraint network is strongly $k$-consistent if any consistent instantiation of any $k-1$ or fewer variables can be extended consistently to any additional variable. Also informally, given an $r$-ary constraint and an instantiation of $r-1$ of the variables that participate in the constraint, the parameter

$m$ is an upper bound on the number of instantiations of the $r$th variable that satisfy the constraint. In general, such sufficient conditions, bounding the level of local consistency that guarantees global consistency, are important in applications where constraint networks are used for knowledge base maintenance and there will be many queries against the knowledge base. Here, the cost of preprocessing will be amortized over the many queries. They are also of interest for their explanatory power, as they can be used for characterizing the difficulty of problems formulated as constraint networks.

We present a sufficient condition, based on the looseness of the constraints and on the size of the domains of the variables, that gives a lower bound on the inherent level of local consistency of a binary constraint network. The bound is straightforward and inexpensive to determine. In contrast, all but low-order local consistency is expensive to verify or enforce as the optimal algorithms are $O(n^k d^k)$, where $k$ is the level of local consistency and $d$ is the size of the domains [6, 29]. The bound is tight for some constraint networks but not for others. Specifically, in any constraint network where the domains are of size $d$ or less, and the constraints have looseness of $m$ or greater, the network is strongly $(\lceil d/(d-m) \rceil)$-consistent[2]. The parameter $m$ can be viewed as a lower bound on the number of instantiations of a variable that satisfy the constraints. The condition based on constraint looseness is useful in two ways. First, it can be used in deciding which low-order local consistency techniques will *not* change the network and thus are not useful for processing a given constraint network. For example, we use our results to show that the $n$-queens problem, a widely used test-bed for comparing backtracking algorithms, has a high level of inherent local consistency. As a consequence, it is generally fruitless to preprocess such a network. Second, it can be used in deciding which local consistency conditions, if any, still need to be enforced if we want to ensure that a solution can be found in a backtrack-free manner.

We also present a new definition of local consistency called *relational consistency*. The virtue of this definition is that, firstly, it allows expressing the relationships between the restrictiveness 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. Finally, it allows identifying those formalisms for which consistency can be decided by enforcing pairwise consistency, like propositional databases and linear equalities and inequalities, from general databases requiring higher levels of local consistency.

[2] $\lceil x \rceil$, the ceiling of $x$, is the smallest integer greater than or equal to $x$.

## 2 Definitions and Preliminaries

We begin with some needed definitions.

**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.

Given a tuple representing an instantiation of only some of the variables in a constraint network, the notion of a consistent instantiation can be defined in primarily two ways. The difference in the definitions lies in how we treat the case where only some of the variables over which a relation is defined are instantiated. 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; that is, the new relation that consists of the tuples of* $R_{S_1}$ *and* $R_{S_2}$ *combined on all their common variables. 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 $S_1 \cup S_2$.*

Following Montanari [25], 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 $a$th element of $D_i$ and the $b$th element of $D_j$ is not permitted; a one entry means that the pair is permitted. Two concepts central to this paper are the tightness and the looseness of constraints.

**Definition 4 ($m$-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 5 ($m$-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.*

Informally, an $r$-ary relation is $m$-tight ($m$-loose) if every tuple of $r-1$ values can be extended in at most (at least) $m$ ways.

**Definition 6 ($m$-tight and $m$-loose general constraints)** *An $r$-ary relation $R$ on a set $S$ of variables $\{x_1, \ldots, x_r\}$ is $m$-*tight *($m$-*loose*) 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 $m$-tight ($m$-loose).*

**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 domains of the variables are $D_i$ = {a,b,c} and the relations are given by,

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

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,

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

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,

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

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 and 1-loose. 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 consistency definitions.

## 3.1 Variable-based consistency

Mackworth [21, 22] defines three properties of networks that characterize local consistency of networks: *node*, *arc*, and *path consistency*. Freuder [14] generalizes this to $k$-consistency.

**Definition 7 ($k$-consistency; Freuder [14, 15])** *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 $k$th 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 [9].

| | | | |
|---|---|---|---|
| Q | | | |
| | | | |
| | Q | | |
| | | | |

(a)

| | | | |
|---|---|---|---|
| Q | | | |
| | | Q | |
| | | | |
| | Q | | |

(b)

Figure 1: (a) not 3-consistent; (b) not 4-consistent

**Example 2.** We illustrate the definition of $k$-consistency using the well-known $n$-queens problem. The problem is to find all ways to place $n$-queens on an $n \times n$ chess board, one queen per column, so that each pair of queens does not attack each other. One possible constraint network formulation of the problem is as follows: there is a variable for each column of the chess board, $x_1, \ldots, x_n$; the domains of the variables are the possible row positions, $D_i = \{1, \ldots, n\}$; and the binary constraints are that two queens should not attack each other. Consider the constraint network for the 4-queens problem. It can be seen that the network is 2-consistent since, given that we have placed a single queen on the board, we can always place a second queen such that the queens do not attack each other. However, the network is not 3-consistent. For example, given the consistent placement of two queens shown in Figure 1a, there is no way to place a queen in the third column that is consistent with the previously placed queens. Similarly the network is not 4-consistent (see Figure 1b).

### 3.2 Relation-based consistency

In [33], 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. We now generalize relational arc- and path-consistency to *relational $m$-consistency.* In the definition of *relational $m$-consistency*, the relations rather than the variables are the primitive entities. As we shall see in subsequent sections, this allows expressing the relationships between the restrictiveness of the constraints and local consistency in a way that avoids an explicit reference to the arity of the constraints.

**Definition 8 (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.*

**Definition 9 (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. Verifying relational $m$-consistency can be exponential even for relational arc-consistency, if the arity of the constraints is not bounded. In general, relational arc-consistency is $O(end^r)$, where $e$ is the number of constraints, $n$ is the number of variables, $d$ is the size of the domains, and $r$ is the arity of the constraints. When $r$ is a constant, verifying relational arc-consistency is polynomial; otherwise it is not.

**Example 3.** Consider the following constraint network over the set of variables $\{x_1, x_2, x_3, x_4, x_5\}$. 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 $R_2 = R_3 = R_4 = \{$a$\}$ and $R_1 = R_5 = \{$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. In general, the definition of relational $m$-consistency is similar but not identical to that of variable-based $m$-consistency over the dual representation of the problem in which the constraints are the variables, their allowed tuples are their respective domains and two such constraint-variables are constrained if they have variables in common. The virtue in this new explicit definition (relative to the one based on the dual graph) is that it is simpler to work with, it uses known notations from the relational database model, and it can be incorporated naturally into algorithms for enforcing desired levels of relational $m$-consistency. Below we present algorithm Relational-Consistency, a brute-force algorithm for enforcing strong relational $m$-consistency on a network $\mathcal{R}$.

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

**repeat**
$Q \leftarrow \mathcal{R}$
**for** every $m-1$ relations $R_{S_1}, \ldots, R_{S_{m-1}} \in Q$ (not necessarily distinct) and every $x \in \bigcap_{i=1}^{m-1} S_i$
**do** $A \leftarrow \bigcup_{i=1}^{m-1} S_i - \{x\}$
$R_A \leftarrow R_A \cap \; \Pi_A(\bowtie_{i=1}^{m-1} R_{S_i})$
**if** $R_A$ is the empty relation
**then** exit and return the empty network
**until** $Q = \mathcal{R}$

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 $A$. The algorithm takes any $m-1$ relations that may or may not be relationally $m$-consistent and enforces relational $m$-consistency by tightening the relation among the appropriate subsets of variables. We call the operation in Step 5 of the algorithm *extended m-composition*, since it generalizes the composition operation defined on binary relations. Algorithm RELATIONAL-CONSISTENCY computes the closure of $\mathcal{R}$ with respect to extended $m$-composition.

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.

While enforcing *variable-based* $m$-consistency can be done in polynomial time, it is unlikely that relational $m$-consistency can be achieved tractably since even for $m = 3$ it solves the NP-complete problem of propositional satisfiability (see Example 8). A more direct argument suggesting an increase in time and space complexity is the fact that the algorithm may need to record relations of arbitrary arity. 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 which enforces strong relational $m$-consistency on a network $\mathcal{R}$, relative to a given ordering $o$, of the variables $x_1, x_2, \ldots, x_n$. We call the network generated by the algorithm the *directional closure of* $\mathcal{R}$.

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 for** every $j$ relations $R_{S_1}, \ldots, R_{S_j}$ in $bucket_p$, $j \leftarrow 1$ **to** $m-1$
4.         **do** $A \leftarrow \bigcup_{i=1}^{j} S_i - \{x_p\}$
5.             $R_A \leftarrow R_A \cap \Pi_A(\bowtie_{i=1}^{j} R_{S_i})$
6.             **if** $R_A$ is not the empty relation
7.                 **then** add $R_A$ to its appropriate bucket
8.                 **else** exit and return the empty network

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* [2]). 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 [3], nevertheless every ordering of the variables $o$, yields a simple to compute upper bound denoted $w^*(o)$ (see [12]). The complexity of Directional-Relational-Consistency along $o$ can be bounded as a function of $w^*(o)$ of its constraint graph. Specifically, it can be shown, using similar theorems on directional consistency algorithms reported earlier [12], that the time complexity and size of the network generated by Directional-Relational-Consistency along ordering $o$ is $O(\exp(mw^*(o)))$.

**Example 4.** Crossword puzzles have been used in experimentally evaluating backtracking algorithms for solving constraint networks [18]. We use an example puzzle (taken from [8]) to illustrate algorithm Directional-Relational-Consistency (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,

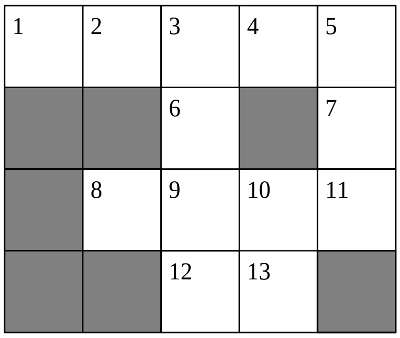



Figure 2: A crossword puzzle

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

Let us perform three iterations of Directional-Relational-Consistency, with $m$ equal to 3 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,

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

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,

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

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.

# 4 Constraint Tightness vs Global Consistency

In this section, we present relationships between the tightness of the constraints and the level of local consistency sufficient to ensure that a network is globally consistent. After reviewing previous work, we present our results for binary constraint networks, and then generalize the results to networks with constraints of arbitrary arity.

## 4.1 Related work

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. Here we review only the literature for constraint networks with finite domains.

For work that falls into the class of identifying topological properties, Freuder [15, 16] 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. As a special case, if the constraint graph is a tree, arc consistency is sufficient to ensure a solution can be found without backtracking. 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 databases in the database community [4].

For work that falls into the class of identifying properties of the constraints (the class into which the present work falls), Montanari [25] shows that path consistency is sufficient to guarantee that a binary network is globally consistent if the relations are monotone. Van Beek and Dechter [33] show that if the relations of a path consistent binary network are row convex, the network is globally consistent. 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. She proves the following result.

**Theorem 1 (Dechter [9])** *Any $d$-valued $r$-ary constraint network that is strongly $(d(r-1)+1)$-consistent is globally consistent. In particular, any $d$-valued binary constraint network that is strongly $(d+1)$-consistent is globally consistent.*

For some networks, Dechter's theorem 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, Dechter's theorem overestimates. Our results should be viewed as an improvement on Dechter's theorem. In particular, by taking into account the tightness of the constraints, our results always specify a level of strong consistency that is less than or equal to the level of strong consistency required by Dechter's theorem.

## 4.2 Tightness: Binary constraint networks

The following lemma is needed in the proof of the main result for constraint networks with binary constraints and in a later proof of the result generalized to constraint networks with constraints of arbitrary arity. The lemma is really about the "tightness" of constraints and the sufficiency of a certain level of consistency. We state the lemma in more colloquial terms to make the proof more understandable.

**Lemma 1** *Suppose there are fan clubs that like to meet and talk about famous people, and the following conditions.*

1. *There are $n$ fan clubs and $d$ famous people.*
2. *Each fan club meets and talks about at most $m$, $m < d$, famous people.*
3. *For every set of $m+1$ or fewer fan clubs, there exists at least one famous person that every club in the set talks about.*

*Then, there must exist at least one famous person that every fan club talks about.*

**Proof.** The proof is by contradiction and uses a proof technique discovered by Dechter for Theorem 1. Assume to the contrary that no such famous person exists. Then, for each famous person, $f_i$, there must exist at least one fan club that does not talk about $f_i$. Let $c_i$ denote one of the fan clubs that does not talk about $f_i$. By construction, the set $c = \{c_1, c_2, \ldots, c_d\}$ is a set of fan clubs for which there does not exist a famous person that every club in the set talks about (every candidate $f_i$ is ruled out since $c_i$ does not talk about $f_i$). For every possible value of $m$, this leads to a contradiction.

**Case 1** ($m = d - 1$): The contradiction is immediate as $c = \{c_1, c_2, \ldots, c_d\}$ is a set of fan clubs of size $m + 1$ for which there does not exist a famous person that every club in the set talks about. This contradicts condition (3).

**Case 2** ($m = d-2$): The nominal size of the set $c = \{c_1, c_2, \ldots, c_d\}$ is $m+2$. We claim, however, that there is a repetition in $c$ and that the true size of the set is $m+1$. Assume to the contrary that $c_i \neq c_j$ for $i \neq j$. Recall $c_i$ is a club that does not talk about $f_i$, $i = 1, \ldots, d$ and consider $\{c_1, c_2, \ldots, c_{d-1}\}$. This is a set of $m+1$ fan clubs so by condition (3) there must exist an $f_i$ that every club in the set talks about. The only possibility is $f_d$. Now consider $\{c_1, \ldots, c_{d-2}, c_d\}$. Again, this is a set of $m + 1$ fan clubs so there must exist an $f_i$ that every club in the set talks about. This time the only possibility is $f_{d-1}$. Continuing in this manner, we can show that fan club $c_1$ must talk about exactly $m + 1$ famous people. This contradicts condition (2). Therefore, it must be the case that $c_i = c_j$ for some $i \neq j$. Thus, the set $c$ is of size $m + 1$ and this contradicts condition (3).

**Case 3** ($m = d - 3$), …, **Case d-1** ($m = 1$): The remaining cases are similar. In each case we argue that (i) there are repetitions in the set $c = \{c_1, c_2, \ldots, c_d\}$, (ii) the true size of the set $c$ is $m + 1$, and (iii) a contradiction is derived by appealing to condition (3).

Thus, there exists at least one famous person that every fan club talks about. □

We now state the theorem for binary constraint networks.

**Theorem 2** *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.*

**Proof.** We show that any network with $\leq m$ ones in every row that is strongly $(m+2)$-consistent is $(m+2+i)$-consistent for any $i \geq 1$. Suppose that variables $x_1, \ldots, x_{m+1+i}$ can be consistently instantiated with values $a_1, \ldots, a_{m+1+i}$. To show that the network is $(m + 2 + i)$-consistent, we must show that there exists at least one instantiation, $a_{m+2+i}$, of variable $x_{m+2+i}$ such that

$$(a_j, a_{m+2+i}) \in R_{j,m+2+i} \qquad j = 1, \ldots, m + 1 + i$$

is satisfied. Let $v_j$ be the (0,1)-vector given by row $a_j$ of the (0,1)-matrix $R_{j,m+2+i}$, $j = 1, \ldots, m+1+i$ (see Figure 3 for an illustration; the $v_j$ are shown boxed). The one entries in the $v_j$ are the allowed instantiations of $x_{m+2+i}$, given the instantiations $a_1, \ldots, a_{m+1+i}$. That there exists a consistent instantiation of $x_{m+2+i}$ follows from Lemma 1 where (i) $a_1, \ldots, a_{m+1+i}$ are the fan clubs, (ii) the domain elements of $x_{m+2+i}$, are the famous people, (iii) the one entries in the $v_j$'s are the famous people that fan club $a_j$ talks about, and (iv) condition (3) of Lemma 1 follows from the assumption of strong $(m+2)$ consistency. Therefore, from Lemma 1 it follows that there exists at least one instantiation of $x_{m+2+i}$ that satisfies all the constraints simultaneously. Hence, the network is $(m+2+i)$-consistent. □

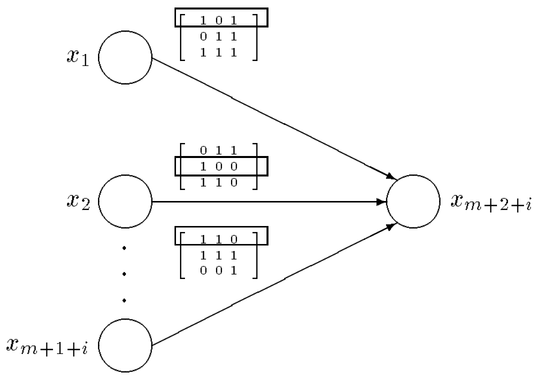



Figure 3: Instantiating $x_{m+2+i}$

Theorem 2 always specifies a level of strong consistency that is less than or equal to the level of strong consistency required by Dechter's theorem (Theorem 1). 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 Dechter's theorem cannot.

As the following example illustrates, both $r$, the arity of the constraints, and $m$ can change if the level of consistency required by the theorem is not present and must be enforced. The parameter $r$ can only increase; $m$ can decrease, as shown below, but also increase. The parameter $m$ will increase if all of the following hold: (i) there previously was no constraint between a set of variables, (ii) enforcing a certain level of consistency results in a new constraint being recorded between those variables and, (iii) the new constraint has a larger $m$ value than the previous constraints.

**Example 5.** Nadel [26] introduces a variant of the $n$-queens problem called confused $n$-queens. The problem is to find all ways to place $n$-queens on an $n \times n$ chess board, one queen per column, so that each pair of queens *does* attack each other. One possible constraint network formulation of the problem is as follows: there is a variable for each column of the chess board, $x_1, \ldots, x_n$; the domains of the variables are the possible row positions, $D_i = \{1, \ldots, n\}$; and the binary constraints are that two queens should attack each other.

The problem is worth considering, as Nadel [26] uses confused $n$-queens in an empirical comparison of backtracking algorithms for solving constraint networks. Thus it is important to analyze the difficulty of the problems to set the empirical results in context. As well, the problem is interesting in that it provides an example where Theorem 2 can be applied but Dechter's theorem can not (since $d \geq n-1$). Independently of $n$, each row and column of the (0,1)-matrix representation of the constraints has at most 3 ones. Hence, the networks are 3-tight and the theorem guarantees that if the network for the confused $n$-queens problem is strongly 5-consistent, the network is globally consistent.

First, suppose that $n$ is even and we attempt to either verify or achieve this level of strong consistency by applying successively stronger local consistency algorithms. Kondrak [20] has shown that the following analysis holds for all $n$, $n$ even.

1. Applying an arc consistency algorithm results in no changes as the network is already arc consistent.

2. Applying a path consistency algorithm does tighten the constraints between the variables. Once the network is made path consistent, each row has $\leq 2$ ones. Now the theorem guarantees that if the constraint network is strongly 4-consistent, the network is globally consistent.

3. Applying a 4-consistency algorithm results in no changes as the network is already 4-consistent. Thus, the network is strongly 4-consistent and therefore also globally consistent.

Second, suppose that $n$ is odd. This time, after applying path consistency, the networks are still 3-tight and it can be verified that the networks are not 4-consistent. Enforcing 4-consistency would require non-binary constraints, hence Theorem 2 no longer applies. We take this example up again in the next section where the results are generalized to non-binary constraints. There we show that recording 3-ary constraints is sufficient.

Recall that Nadel [26] uses confused $n$-queens problems to empirically compare backtracking algorithms for finding all solutions to constraint networks. Nadel states that these problems provide a "non-trivial test-bed" [26, p.190]. We believe the above analysis indicates that these problems are quite easy and that any empirical results on these problems should be interpreted in this light. Easy problems potentially make even naive algorithms for solving constraint networks look promising. To avoid this potential pitfall, backtracking algorithms should be tested on problems that range from easy to hard. In general, hard problems are those that require a high level of local consistency to ensure global consistency. Note also that these problems are trivially satisfiable.

**Example 6.** The graph $k$-colorability problem can be viewed as a problem on constraint networks: there is a variable for each node in the graph; the domains of the variables are the possible colors, $D = \{1, \ldots, k\}$; and the binary constraints are that two adjacent nodes must be assigned different colors. Graph $k$-colorability provides examples of networks where both Theorems 1 and 2 give the same bound on the sufficient level of local consistency (since $d = k$ and $m = d - 1$). Further, as Dechter [9] shows, the bound is tight. For example, consider coloring a complete graph on five nodes with four colors. The network is 3-tight and strongly 4-consistent, but not strongly 5-consistent and not globally consistent. Hence, when $m = d - 1$, the level of local consistency specified by Theorem 2 is as strong as possible and cannot be lowered.

We can also construct examples to show that Theorem 2 is as strong as possible for all $m < d - 1$. This can be done by "embedding" graph coloring constraints into the constraints for the new network. For example, consider the network where the domains are $D = \{1, \ldots, 5\}$ and the constraints between all variables is given by,

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

The inner $3 \times 3$ matrix is the 3-coloring constraint. The network is 2-tight and strongly 3-consistent, but not strongly 4-consistent and not globally consistent.

## 4.3 Tightness: General constraint networks

The results for binary constraint networks can be generalized to networks with constraints of arbitrary arity. We first generalize the results using variable-based local consistency and then generalize the results using relation-based local consistency.

We now state the general theorem for variable-based local consistency.

**Theorem 3** *If an $r$-ary network $\mathcal{R}$ is $m$-tight, and if the network is strongly $((m+1)(r-1)+1)$-consistent, then the network is globally consistent.*

**Proof.** Let $k = (m+1)(r-1)+1$. We show that any network with relations that are $m$-tight that is strongly $k$-consistent is $(k+i)$-consistent for any $i \geq 1$. Let $Y_I = (a_1, \ldots, a_{k+i-1})$ be a consistent instantiation of a set $Y$ of $k+i-1$ variables and let $x_{k+i}$ be an arbitrary new variable. We will show that there exists an instantiation $a_{k+i}$ of $x_{k+i}$ such that the extended tuple $(a_1, \ldots, a_{k+i-1}, a_{k+i})$ is consistent. This means that any relation $R_S \in \mathcal{R}$ defined over variable $x_{k+i}$ and a non-empty subset of variables from $\{x_1, \ldots, x_{k+i-1}\}$ should be satisfied. Let $Y_I[S]$ be the partial tuple of $Y_I$ that is restricted to the set $S$ over which $R_S$ is defined. We call this tuple a *constraint-tuple*. Since all the constraints are $m$-tight, constraint $R_S$ will allow $Y_I[S]$ to be extended by at most $m$ values of $x_{k+i}$. Each such constraint-tuple, $Y_I[S]$ can be regarded as a fan club, with its allowed values in $x_{k+i}$ relative to $R_S$ as the famous people discussed by the fan club. Therefore, condition (2) of Lemma 1 is satisfied. Also, condition (3) of Lemma 1 is satisfied, since the length of each constraint-tuple is $r-1$ or less, the requirement of strong $(m+1)(r-1)+1$-consistency, ensures that any set of up to $(m+1)$ constraint-tuples (overlapping or not), has a consistent extension in $x_{k+i}$. Therefore, from Lemma 1 it follows that there is a common value of $x_{k+i}$ that satisfies all the constraints simultaneously. $\Box$

**Example 7.** Consider again the confused $n$-queens problem discussed in Example 5. There we saw that, after enforcing path consistency, the networks are 3-tight, for $n$ odd. Enforcing 4-consistency requires 3-ary constraints. Adding the necessary 3-ary constraints does not change the value of $m$; the networks are still 3-tight. Hence, by Theorem 3, if the networks are strongly 9-consistent, the networks are globally consistent. Kondrak [20] has shown that recording 3-ary constraints is sufficient to guarantee the networks are strongly 9-consistent for all $n$, $n$ odd. Hence, independently of $n$, the networks are globally consistent once strong 4-consistency is enforced.

We now show how the concept of relation-based local consistency can be used to alternatively describe Theorem 3.

**Theorem 4** *If a constraint network $\mathcal{R}$ is $m$-tight, and if the network is strongly relationally $(m+2)$-consistent, then the network is globally consistent.*

**Proof.** Assume that the network is relationally $(m+2)$-consistent. Let $Y_I = (a_1, \ldots, a_{i-1})$ be a consistent instantiation of a set $Y$ of $i-1$ variables, $i > m+2$. We show that for any $x_i$, there exists an instantiation $a_i$ of $x_i$ such that the extended tuple $(a_1, \ldots, a_{i-1}, a_i)$ is consistent. This means that any relation $R_S \in \mathcal{R}$ defined over variable $x_i$ and a non-empty subset of variables from $\{x_1, \ldots, x_{i-1}\}$, should be satisfied. Since all of the constraints are $m$-tight, the number of values of $x_i$ that, together with $Y_I[S]$, are allowed by $R_S$ does not exceed $m$. Also, strong relational $(m+2)$-consistency implies that any subset of $m+1$ or fewer constraints can be consistently extended by $x_i$. Consequently, due to Lemma 1 there is a value $a_i$ such that the tuple $(a_1, \ldots, a_{i-1}, a_i)$ satisfies all the constraints simultaneously. $\Box$

As an immediate corollary of Theorem 4, if we know that the result of applying Relational-Consistency$(\mathcal{R}, m)$ will be that all of the relations will be $(m-2)$-tight, we can guarantee *a priori* that the algorithm will return an equivalent, globally consistent network. Algorithm Relational-Consistency computes the closure of $\mathcal{R}$ with respect to extended $m$-composition. We can conclude that:

**Corollary 1** *For any network $\mathcal{R}$ whose closure under extended i-composition, for $i = 2, \ldots, m$, is an $(m-2)$-tight network,* Relational-Consistency*$(\mathcal{R}, m)$ computes an equivalent globally consistent network.*

**Proof.** Follows immediately from Theorem 4 and from the fact that algorithm Relational-Consistency generates a strongly relationally $m$-consistent network. □.

**Example 8.** Let the domains of the variables be of size two. Relations formed over domains of size two are 1-tight and closed under extended 2- and 3-composition. Thus, by Corollary 1, bi-valued networks can be solved by Relational-Consistency$(\mathcal{R}, 3)$. In particular, the satisfiability of propositional *CNFs* can be decided by Relational-Consistency$(\mathcal{R}, 3)$. Here the extended composition operation (Step 5 of the algorithm) takes the form of pair-wise resolution [13]. A different derivation of the same result is already given by [9, 33].

While Relational-Consistency is incomplete for deciding consistency in general, it is complete for $(m-2)$-tight relations that are closed under extended $m$-composition. In fact, it is sufficient to require directional $(m-2)$-tightness relative to the ordering used; that is, requiring that if $x_i$ appears before $x_j$ in the ordering then any value of $x_i$ will be $(m-2)$-tight relative to $x_j$ but not vice-versa. For example, functional relations are always 1-tight for orderings that put input variables before output variables.

**Definition 10 (directionally $m$-tight)** *A binary constraint $R_{ij}$ is* directionally $m$-tight *with respect to an ordering of the variables $o = (x_1, \ldots, x_n)$, if $x_i$ appears before $x_j$ in the ordering and every row of the (0,1)-matrix representation of the constraint has at most $m$ ones. An $r$-ary relation $R$ on a set $S$ of variables is* directionally $m$-tight *if for the subset of $r-2$ variables $Y \subseteq S$ that are lowest in the ordering, and for every instantiation $Y_I$ of the variables in $Y$, the binary relation $\Pi_{(S-Y)}(\sigma_{Y_I}(R))$ is directionally $m$-tight.*

Algorithm Directional-Relational-Consistency computes the directional closure of $\mathcal{R}$ with respect to extended $m$-composition. We can conclude that:

**Theorem 5** *For any network $\mathcal{R}$, whose directional closure under extended $m$-composition is directionally $(m-2)$-tight,* Directional-Relational-Consistency*$(\mathcal{R}, m, o)$ will either decide that the network is inconsistent or else compute an equivalent network of $\mathcal{R}$ that is backtrack-free along the ordering $o$.*

**Proof.** Clearly the directional closure of $\mathcal{R}$ is equivalent to $\mathcal{R}$. What needs to be shown is that the directional closure relative to $o$ is backtrack-free along the ordering $o$. We prove this by induction on $o$. Without loss of generality, assume that the ordering is $o = x_1, \ldots, x_n$. For the first variable $x_1$ (recall that $x_1$ is processed last by the algorithm) there must be a value in the domain of $x_1$ that is allowed since otherwise the domain will be empty and the directional closure will be empty. Assume now that we have already consistently instantiated the first $i-1$ variables as $Y_I = (a_1, \ldots, a_{i-1})$. Let $x_i$ be the next variable. We claim that every $m-1$ or fewer *applicable* constraints in the directional closure, each defined on $x_i$ and a subset of $\{x_1, \ldots, x_{i-1}\}$, must have a common extension to $x_i$. Suppose to the contrary that there exists a set of $j$, $1 \leq j \leq m-1$, applicable constraints $R_{S_1 \cup \{x_i\}}, \ldots, R_{S_j \cup \{x_i\}}$ that do not have a common extension to $x_i$. However, the extended composition over these $j$ relations when the bucket of $x_i$ was processed generated a relation over $S_1 \cup \cdots \cup S_j$ that should have been consulted when testing $Y_I$'s consistency and should have disallowed $Y_I$, yielding a contradiction. Thus, using the fact that every set of $m-1$ or fewer applicable relations has a common extension to $x_i$, and that the relations are directionally $(m-2)$-tight, Lemma 1 guarantees that they all have a common extension to $x_i$. $\Box$.

**Example 9.** Consider again the crossword puzzle discussed in Example 4. All of the constraints are 1-tight with the exception of $R_{5,7,11}$, which is 2-tight. For the ordering $o = x_1, \ldots, x_{13}$, however, the constraint $R_{5,7,11}$ is directionally 1-tight: $x_5$ is the lowest variable in the ordering and we can verify that for every instantiation $a_5$ of $x_5$, the binary relation $\Pi_{7,11}(\sigma_{x_5=a_5}(R_{5,7,11}))$ is 1-tight. Therefore, the network is directionally 1-tight. According to Theorem 5, enforcing relational 3-consistency will generate a backtrack-free network along the ordering $o$, provided the tightness of the network does not increase as a result.

# 5 Constraint Looseness vs Local Consistency

In this section, we present a sufficient condition, based on the looseness of the constraints and on the size of the domains of the variables, that gives a lower bound on the inherent level of local consistency of a binary constraint network. After reviewing previous work, we present our results for binary constraint networks, and then generalize the results to networks with constraints of arbitrary arity.

## 5.1 Related work

It is known that some classes of constraint networks already possess a certain level of local consistency and therefore algorithms that enforce this level of local consistency will have no effect on these networks. For example, Nadel [26] observes that an arc consistency algorithm never changes a constraint network formulation of the $n$-queens problem, for $n > 3$. Dechter [9] observes that constraint networks that arise from the graph $k$-coloring problem are inherently strongly $k$-consistent. Our results characterize what it is about the structure of the constraints in these networks that makes these statements true.

## 5.2 Looseness: Binary constraint networks

We now present a simple condition that estimates the inherent level of strong $k$-consistency of a binary constraint network. The condition is sufficient but not necessary for local consistency.

**Theorem 6** *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.*

**Proof.** We show that the network is $k$-consistent for all $k \leq \lceil d/(d-m) \rceil$. Suppose that variables $x_1, \ldots, x_{k-1}$ can be consistently instantiated with values $a_1, \ldots, a_{k-1}$. To show that the network is $k$-consistent, we must show that there exists at least one instantiation $a_k$ of variable $x_k$ that satisfies all the constraints,

$$(a_i, a_k) \in R_{ik} \qquad i = 1, \ldots, k-1$$

simultaneously. We do so as follows. The instantiations $a_1, \ldots, a_{k-1}$ restrict the allowed instantiations of $x_k$. Let $v_i$ be the (0,1)-vector given by row $a_i$ of the (0,1)-matrix $R_{ik}$, $i = 1, \ldots, k-1$. Let $\text{zero}(v_i)$ be the set that lists the positions of the zeros in vector $v_i$. The zero entries in the $v_i$'s are the forbidden instantiations of $x_k$, given the instantiations $a_1, \ldots, a_{k-1}$. No consistent instantiation of $x_k$ exists if and only if $\text{zero}(v_1) \cup \cdots \cup \text{zero}(v_{k-1}) = \{1, \ldots, d\}$. Now, the key to the proof is that all the $v_i$ contain at least $m$ ones. In other words, each $v_i$ contains at most $d - m$ zeros. Thus, if

$$(k-1)(d-m) < d,$$

it cannot be the case that $\text{zero}(v_1) \cup \cdots \cup \text{zero}(v_{k-1}) = \{1, \ldots, d\}$. (To see that this is true, consider the "worst case" where the positions of the zeros in any vector do not overlap with those of any other vector. That is, $\text{zero}(v_i) \cap \text{zero}(v_j) = \emptyset$, $i \neq j$.) Thus, if

$$k \leq \left\lceil \frac{d}{d-m} \right\rceil,$$

all the constraints must have a non-zero entry in common and there exists at least one instantiation of $x_k$ that satisfies all the constraints simultaneously. Hence, the network is $k$-consistent. □

Theorem 6 always specifies a level of local consistency that is less than or equal to the actual level of inherent local consistency of a constraint network. That is, the theorem provides a lower bound. Graph coloring problems provide examples where the theorem is exact, whereas $n$-queens problems provide examples where the theorem underestimates the true level of local consistency.

**Example 10.** Consider again the well-known $n$-queens problem discussed in Example 2. The problem is of historical interest but also of theoretical interest due to its importance as a test problem in empirical evaluations of backtracking algorithms and heuristic repair schemes for finding solutions to constraint networks (e.g., [17, 19, 24, 26]). For $n$-queens networks, each row and column of the (0,1)-matrix representation of the constraints has $d-3 \leq m \leq d-1$ ones, where $d = n$. Hence, Theorem 6 predicts that $n$-queens networks are inherently strongly ($\lceil n/3 \rceil$)-consistent. Thus, an $n$-queens constraint network is inherently arc-consistent for $n \geq 4$, inherently path consistent for $n \geq 7$, and so on, and we can predict where it is fruitless to apply a low order consistency algorithm in an attempt to simplify the network (see Table 1). The actual level of inherent consistency is $\lfloor n/2 \rfloor$ for $n \geq 7$. Thus, for the $n$-queens problem, the theorem underestimates the true level of local consistency.

Table 1: Predicted ($\lceil n/3 \rceil$) and actual ($\lfloor n/2 \rfloor$, for $n \geq 7$) level of strong local consistency for $n$-queens networks

| $n$ | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| pred. | 2 | 2 | 2 | 3 | 3 | 3 | 4 | 4 | 4 | 5 | 5 | 5 |
| actual | 2 | 2 | 2 | 3 | 4 | 4 | 5 | 5 | 6 | 6 | 7 | 7 |

The reason Theorem 6 is not exact in general and, in particular, for $n$-queens networks, is that the proof of the theorem considers the "worst case" where the positions of the zeros in any row of the constraints $R_{ik}, i = 1, \ldots, k-1$, do not overlap with those of any other row. For $n$-queens networks, the positions of some of the zeros do overlap. However, given only the looseness of the constraints and the size of the domains, Theorem 6 gives as strong an estimation of the inherent level of local consistency as possible as examples can be given where the theorem is exact.

**Example 11.** Graph $k$-colorability provides examples where Theorem 6 is exact in its estimation of the inherent level of strong $k$-consistency (see Example 6 for the constraint network formulation of graph coloring). As Dechter [9] states, graph coloring networks are inherently strongly $k$-consistent but are not guaranteed to be strongly $(k+1)$-consistent. Each row and column of the constraints has $m = d-1$ ones, where $d = k$. Hence, Theorem 6 predicts that graph $k$-colorability networks are inherently strongly $k$-consistent.

**Example 12.** We can also construct examples, for all $m < d-1$, where Theorem 6 is exact. For example, consider the network where, $n = 5$, the domains are $D = \{1, \ldots, 5\}$, and the binary constraints are given by,

$$R_{ij} = \begin{bmatrix} 0\,1\,1\,1\,1 \\ 0\,0\,1\,1\,1 \\ 1\,0\,0\,1\,1 \\ 1\,1\,0\,0\,1 \\ 1\,1\,1\,0\,0 \end{bmatrix}, \; 1 \leq i < j \leq n,$$

and $R_{ji} = R_{ij}^T$, for $j < i$. The network is 3-loose and therefore strongly 3-consistent by Theorem 6. This is exact, as the network is not 4-consistent.

We conclude this section with some discussion on what Theorem 6 contributes to our intuitions about hard classes of problems (in the spirit of, for example, [5, 35]). Hard constraint networks are instances which give rise to search spaces with many dead ends. The hardest networks are those where many dead ends occur deep in the search tree. Dead ends, of course, correspond to partial solutions that cannot be extended to full solutions. Thus, networks where the constraints are loose are good candidates to be hard problems since loose networks have a high level of inherent strong consistency and strong $k$-consistency means that all partial solutions are of at least size $k$.

Computational experiments we performed on random problems provide evidence that loose networks can be hard. Random problems were generated with $n = 50$, $d = 5, \ldots 10$, and $p, q = 1, \ldots, 100$, where $p/100$ is the probability that there is a binary constraint between two variables, and $q/100$ is the probability that a pair in the Cartesian product of the domains is in the constraint. The time to find one solution was measured. In the experiments we discovered that, given that the number of variables and the domain size were fixed, the hardest problems were found when the constraints were as loose as possible without degenerating into the trivial constraint where all tuples are allowed. That networks with loose constraints would turn out to be the hardest of these random problems is somewhat counter-intuitive, as individually the constraints are easy to satisfy. These experimental results run counter to Tsang's [30, p.50] intuition that a single solution of a loosely constrained problem "can easily be found by simple backtracking, hence such problems are easy," and that tightly constrained problems are "harder compared with loose problems." As well, these hard loosely-constrained problems are not amenable to preprocessing by low-order local consistency algorithms, since, as Theorem 6 states, they possess a high level of inherent local consistency. This runs counter to Williams and Hogg's [35, p.476] speculation that preprocessing will have the most dramatic effect in the region where the problems are the hardest.

## 5.3 Backtrack-free binary networks

Given an ordering of the variables in a constraint network, backtracking search works by successively instantiating the next variable in the ordering, and backtracking to try different instantiations for previous variables when no consistent instantiation can be given to the current variable. Previous work has identified conditions for when a certain level of local consistency is sufficient to ensure a solution can be found in a backtrack-free manner (see Section 4.1). Sometimes the level of inherent strong $k$-consistency guaranteed by Theorem 6 is sufficient, in conjunction with these previously derived conditions, to guarantee that the network is globally consistent and therefore a solution can be found in a backtrack-free manner without preprocessing. Otherwise, the estimate provided by the theorem gives a starting point for applying local consistency algorithms.

In this section, we use constraint looseness to identify two new classes of backtrack-free binary networks. First, we give a condition for a network to be inherently globally consistent. Second, we give a condition, based on a directional version of the looseness property, for an ordering to be backtrack-free. We also give an efficient algorithm for finding an ordering that satisfies the condition, should it exist.

We begin with a corollary of Theorem 6.

**Corollary 2** *If a binary constraint network $\mathcal{R}$ is $m$-loose, all domains are of size $d$ or less, and $m > \frac{n-2}{n-1}d$, the network is globally consistent.*

**Proof.** By Theorem 6, the network is strongly $n$-consistent if $\lceil d/(d-m) \rceil \geq n$. This is equivalent to, $d/(d-m) > n-1$ and rearranging for $m$ gives the result. $\square$

The condition in the corollary is not empty only whenever $d$ is greater than or equal to $n-1$. As one example, consider a constraint network with $n = 5$ variables that has domains of at most size $d = 10$ and constraints that are 8-loose. The network is globally consistent and, as a consequence, a solution can be found in a backtrack-free manner. Another example is networks with $n = 5$, domain sizes of $d = 5$, and constraints that are 4-loose.

Global consistency implies that all orderings of the variables are backtrack-free orderings. Sometimes, however, there exists a backtrack-free ordering when only much weaker local consistency conditions hold. Freuder [15] identifies a relationship between the *width* of an ordering of the variables and the level of local consistency sufficient to ensure an ordering is backtrack-free.

**Definition 11 (width; Freuder [15])** *Let $o = (x_1, \ldots, x_n)$ be an ordering of the variables in a binary constraint network. The* width of a variable*, $x_i$, is the number of binary constraints between $x_i$ and variables previous to $x_i$ in the ordering. The* width of an ordering *is the maximum width of all variables.*

**Theorem 7 (Freuder [15])** *An ordering of the variables in a binary constraint network is backtrack-free if the level of strong $k$-consistency of the network is greater than the width of the ordering.*

Dechter and Pearl [11] define a weaker version of $k$-consistency, called directional $k$-consistency, and show that Theorem 7 still holds. Both versions of

$k$-consistency are, in general, expensive to verify, however. Dechter and Pearl also give an algorithm, called adaptive consistency, that does not enforce a uniform level of local consistency throughout the network but, rather, enforces the needed level of local consistency as determined on a variable by variable basis. We adapt these two insights, directionality and not requiring uniform levels of local consistency, to a condition for an ordering to be backtrack-free.

**Definition 12 (directionally $m$-loose)** *A binary constraint is* directionally $m$-loose *if every row of the (0,1)-matrix representation of the constraint has at least $m$ ones, where $0 \le m \le d-1$.*

**Theorem 8** *An ordering of the variables $o = (x_1, \ldots, x_n)$ in a binary constraint network $\mathcal{R}$ is backtrack-free if $\left\lceil \frac{d}{d-m_{ij}} \right\rceil > w_j$, $1 \le i < j \le n$, where $w_j$ is the width of variable $x_j$ in the ordering, and $m_{ij}$ is the directional looseness of the (non-trivial) constraint $R_{ij}$.*

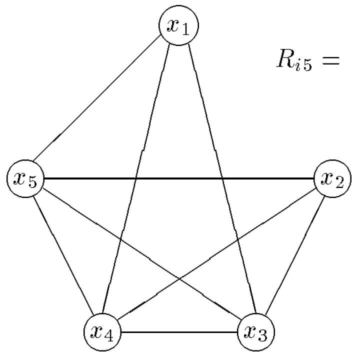
**Proof.** Similar to the proof of Theorem 6. □

An algorithm for finding such a backtrack-free ordering of the variables, should it exist, is given below.

FINDORDER($\mathcal{R}, n$)
1. $I \leftarrow \{1, 2, \ldots, n\}$
2. **for** $p \leftarrow n$ **downto** 1 **do**
3. find a $j \in I$ such that, for each $R_{ij}$, $i \in I$ and $i \neq j$, $\lceil d/(d - m_{ij}) \rceil > w_j$, where $w_j$ is the number of constraints $R_{ij}$, $i \in I$ and $i \neq j$, and $m_{ij}$ is the directional looseness of $R_{ij}$
 (if no such $j$ exists, report failure and halt)
4. put variable $x_j$ at position $p$ in the ordering
5. $I \leftarrow I - \{j\}$

**Example 13.** Consider the network in Figure 4. The network is 2-consistent, but not 3-consistent and not 4-consistent. Freuder [15], in connection with Theorem 7, gives an algorithm for finding an ordering which has the minimum width of all orderings of the network. Assuming that the algorithms break ties by choosing the variable with the lowest index, the minimal width ordering found is $(x_5, x_4, x_3, x_2, x_1)$, which has width 3. Thus, the condition of Theorem 7 does not hold. In fact, this ordering is not backtrack-free. For example, the instantiation $x_5 = 1$, $x_4 = 3$, $x_3 = 5$ is a dead end, as there is no consistent instantiation for $x_2$. The ordering found by procedure FINDORDER is $(x_4, x_3, x_2, x_1, x_5)$, which has width 4. It can be verified that the condition of Theorem 8 holds. For example, $w_1$, the width at variable $x_1$, is 2, and the constraints $R_{41}$ and $R_{31}$ are both 3-loose, so $\left\lceil \frac{d}{d-3} \right\rceil = 3 > w_1 = 2$. Therefore all solutions of the network can be found with no backtracking along this ordering.

$$R_{ij} = \begin{bmatrix} 0\,0\,1\,1\,1 \\ 1\,0\,0\,1\,1 \\ 1\,1\,0\,0\,1 \\ 1\,1\,1\,0\,0 \\ 1\,1\,1\,1\,0 \end{bmatrix}, \ i = 1,2; \ j = 3,4$$




$$R_{i5} = \begin{bmatrix} 0\,1\,1\,1\,1 \\ 1\,0\,1\,1\,1 \\ 1\,1\,0\,1\,1 \\ 1\,1\,1\,0\,1 \\ 1\,1\,1\,1\,0 \end{bmatrix}, \ i = 1, \ldots, 4$$

$$R_{34} = \begin{bmatrix} 0\,0\,1\,1\,0 \\ 0\,1\,1\,0\,1 \\ 1\,1\,0\,1\,1 \\ 1\,0\,1\,1\,0 \\ 0\,1\,1\,0\,0 \end{bmatrix}$$

$$R_{ji} = R_{ij}^T, \ j < i$$

Figure 4: Constraint network for which a backtrack-free ordering exists

## 5.4 Looseness: General constraint networks

The results for binary constraint networks can be generalized to networks with constraints of arbitrary arity. We first generalize the results using variable-based local consistency and then generalize the results using relation-based local consistency.

We now state the general theorem for variable-based local consistency. Let binomial$(k, r)$ be the binomial coefficients, the number of possible choices of $r$ different elements from a collection of $k$ objects. If $k < r$, then binomial$(k, r)$ $= 0$.

**Theorem 9** *If a constraint network $\mathcal{R}$ is $m$-loose, all domains are of size $d$ or less, and all constraints are of arity $r$, then the network is strongly $k$-consistent, where $k$ is the minimum value such that the following inequality holds,*

$$\text{binomial}(k-1, r-1) \geq \left\lceil \frac{d}{d-m} \right\rceil - 1.$$

**Proof.**

*Part 1.* Suppose that variables $x_1, \ldots, x_{k-1}$ can be consistently instantiated with values $a_1, \ldots, a_{k-1}$. To show that the network is $k$-consistent, we must show that there exists at least one instantiation $a_k$ of variable $x_k$ that satisfies all the applicable constraints. This means that any relation $R_S \in \mathcal{R}$ defined over variable $x_k$ and a non-empty subset of variables from $\{x_1, \ldots, x_{k-1}\}$ should be satisfied. Since all the constraints are $m$-loose, each applicable constraint $R_S$ will allow at least $m$ different values for $x_k$. Thus, in a manner similar to that of the proof of Theorem 6, we can show that any set of $\lceil d/(d-m) \rceil - 1$ or fewer constraints will have at least one instantiation of $x_k$ that satisfies all the constraints simultaneously.

*Part 2.* Now, given that any set of $\lceil d/(d-m) \rceil - 1$ or fewer constraints have a common extension, the level of strong $k$-consistency is the minimum number of distinct variables that can occur among this many constraints. Each of the constraints will be over a different subset of variables and in each of the constraints one of the variables is fixed to be $x_k$. The binomial coefficients binomial$(k-1, r-1)$ tell us the minimum number of variables that are needed in order to specify the remaining $r-1$ variables in each constraint subject to the condition that each constraint must be over a different subset of variables. □

The theorem is restricted in its applicability as it requires that all of the constraints are of equal arity $r$. No equivalent result was found for general $r$-ary networks, where the constraints have arity $r$ or less. Nevertheless, the theorem does apply to some interesting classes of networks.

**Example 14.** Consider a formula in 3-CNF which can be viewed as a constraint network where each variable has the domain {true, false} and each clause corresponds to a constraint defined by its models. The domains are of size two and all constraints are of arity 3 and are 1-loose. The minimum value of $k$ such that the inequality in Theorem 9 holds is when $k = 3$. Hence, the networks are strongly 3-consistent.

We now show how the concept of relation-based local consistency can be used to alternatively describe Theorem 9.

**Theorem 10** *If a constraint network $\mathcal{R}$ is m-loose and all domains are of size d or less, then the network is strongly relational* $\left( \left\lceil \frac{d}{d-m} \right\rceil \right)$*-consistent.*

**Proof.** Same as Part 1 of proof of Theorem 9. □

# 6 Conclusions and Future Work

We identified two new complementary properties on the restrictiveness of the constraints in a network: *constraint tightness* and *constraint looseness*. Constraint tightness was used, in conjunction with the level of local consistency, in a sufficient condition that guarantees that a solution to a network can be found in a backtrack-free manner. The condition can be useful in applications where a knowledge base will be queried over and over and the preprocessing costs can be amortized over many queries. Constraint looseness was used in a sufficient condition for local consistency. The condition is straightforward and inexpensive to determine and can be used to estimate the level of strong local consistency of a network. This in turn can be used in deciding whether it would be useful to preprocess the network before a backtracking search, and in deciding which local consistency conditions, if any, still need to be enforced if we want to ensure that a solution can be found in a backtrack-free manner.

We also showed how constraint tightness and constraint looseness are of interest for their explanatory power, as they can be used for characterizing the difficulty of problems formulated as constraint networks and for explaining why some problems that are "easy" locally, are difficult globally. We showed that when the constraints are tight, networks may require less preprocessing in order to guarantee a backtrack-free solution and that when the constraint are loose, networks may require much more search effort in order to find a solution. As an example, the confused $n$-queens problem, which has tight constraints, was shown to be easy to solve as it is backtrack-free after enforcing only low-order local consistency conditions. As another example, many instances of crossword puzzles are also relatively easy, as the constraints are tight. On the other hand, scheduling problems involving resource constraints can be quite hard, as the constraints are inequality constraints and thus are loose.

A property of constraints proposed by Nudel [27], which is related to constraint tightness and constraint looseness, counts the number of ones in the entire constraint. Nudel uses this count, called a compatibility count, in an effective variable ordering heuristic for backtracking search. We plan to examine whether $m$-tightness and $m$-looseness can be used to develop even more effective domain and variable ordering heuristics. We also plan to examine how the looseness property can be used to improve the average case efficiency of local consistency algorithms. The idea is to predict whether small subnetworks already possess some specified level of local consistency, thus potentially avoiding the computations needed to enforce local consistency on those parts of the network. We also plan to examine how to efficiently implement the directional relational consistency algorithm and we are currently exploring an *adaptive* version of the algorithm that takes into account the width of a node in dynamically deciding what level of relational $m$-consistency to enforce.

## References


[1] J. F. Allen. Maintaining knowledge about temporal intervals. *Comm. ACM*, 26:832–843, 1983.

[2] S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability — a survey. *BIT*, 25:2–23, 1985.

[3] 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.

[4] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30:479–513, 1983.

[5] P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the really hard problems are. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 331–337, Sydney, Australia, 1991.

[6] M. C. Cooper. An optimal k-consistency algorithm. *Artificial Intelligence*, 41:89–95, 1989.

[7] R. Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41:273–312, 1990.

[8] R. Dechter. Constraint networks. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence, 2nd Edition*, pages 276–285. John Wiley & Sons, 1992.

[9] R. Dechter. From local to global consistency. *Artificial Intelligence*, 55:87–107, 1992.

[10] R. Dechter and I. Meiri. Experimental evaluation of preprocessing techniques in constraint satisfaction problems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 271–277, Detroit, Mich., 1989.

[11] R. Dechter and J. Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34:1–38, 1988.

[12] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.

[13] R. Dechter and I. Rish. Directional resolution: The Davis-Putnam procedure, revisited. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Bonn, Germany, 1994.

[14] E. C. Freuder. Synthesizing constraint expressions. *Comm. ACM*, 21:958–966, 1978.

[15] E. C. Freuder. A sufficient condition for backtrack-free search. *J. ACM*, 29:24–32, 1982.

[16] E. C. Freuder. A sufficient condition for backtrack-bounded search. *J. ACM*, 32:755–761, 1985.

[17] J. Gaschnig. Experimental case studies of backtrack vs. waltz-type vs. new algorithms for satisficing assignment problems. In *Proceedings of the Second Canadian Conference on Artificial Intelligence*, pages 268–277, Toronto, Ont., 1978.

[18] M. L. Ginsberg, M. Frank, M. P. Halpin, and M. C. Torrance. Search lessons learned from crossword puzzles. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 210–215, Boston, Mass., 1990.

[19] R. M. Haralick and G. L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–313, 1980.

[20] G. Kondrak, 1993. Personal Communication.

[21] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[22] A. K. Mackworth. Constraint satisfaction. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*. John Wiley & Sons, 1987. Use updated reference Mackworth92.

[23] H. Maruyama. Structural disambiguation with constraint propagation. In *Proceedings of the 28th Conference of the Association for Computational Linguistics*, pages 31–38, Pittsburgh, Pennsylvania, 1990.

[24] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 17–24, Boston, Mass., 1990.

[25] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Inform. Sci.*, 7:95–132, 1974.

[26] B. A. Nadel. Constraint satisfaction algorithms. *Computational Intelligence*, 5:188–224, 1989.

[27] B. Nudel. Consistent-labeling problems and their algorithms: Expected-complexities and theory-based heuristics. *Artificial Intelligence*, 21:135–178, 1983.

[28] P. Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9:268–299, 1993.

[29] R. Seidel. On the complexity of achieving k-consistency. Department of Computer Science Technical Report 83-4, University of British Columbia, 1983. Cited in: A. K. Mackworth 1987.

[30] E. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, 1993.

[31] P. van Beek. On the inherent level of local consistency in constraint networks. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 368–373, Seattle, Wash., 1994.

[32] P. van Beek and R. Dechter. Constraint tightness versus global consistency. In *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, pages 572–582, Bonn, Germany, 1994.

[33] P. van Beek and R. Dechter. On the minimality and global consistency of row-convex constraint networks. *Accepted for publication in J. ACM*, 1994.

[34] D. Waltz. Understanding line drawings of scenes with shadows. In P. H. Winston, editor, *The Psychology of Computer Vision*, pages 19–91. McGraw-Hill, 1975.

[35] C. P. Williams and T. Hogg. Using deep structure to locate hard problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 472–477, San Jose, Calif., 1992.