Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

FROM LOCAL TO GLOBAL CONSISTENCY

Rina Dechter

July 1991
CSD-910046

In Proceeding, CSCSI-90, May 1990, pp. 231-237. R-135

FROM LOCAL TO GLOBAL CONSISTENCY*

Rina Dechter**

Computer Science Department

Technion - Israel Institute of Technology

Haifa 32000, Israel

e-mail: dechter@techsel.bitnet

Abstract

In reasoning tasks involving the maintenance of consistent databases (so called ''constraint networks''), it is customary to enforce local consistency conditions in order to simplify the subsequent construction of a globally coherent model of the data. In this paper we present a relationship between the sizes of the variables' domains and the level of local consistency sufficient to ensure global consistency. The results are presented for binary constraint networks first, and then are generalized to higher order constraints. An interesting result emerging from this relationship is that any relation on bi-valued variables which is not representable by a network of binary constraints cannot be represented by networks with hidden variables, regardless of the number of hidden variables allowed.

*This work was supported in part by the National Science Foundation, Grant #IRI-88215522 and by the Air Force Office of Scientific Research, Grant #AFOSR-80-0136 while the author was visiting the cognitive systems lab at UCLA.

**The author is currently with the Information and Computer Science department at UC-Irvine.

1. Introduction

One of the major forces shaping resource-bounded reasoning stem from the requirement of local computation, namely, from the need to consider only a few data items at any inference step and to avoid the decision where to store intermediate results. All realistic models of human reasoning invoke this notion of locality in one form or another. For example, spreading activation in conceptual memories is grounded in the notion that activity spreads locally among conceptually neighboring entities, but does not leap toward remotely designated addresses.

In reasoning tasks involving the maintenance of consistent set of information items (so called ''constraints''), the principle of locality has been embodied in techniques that enforce local consistency among groups of related variables. The rationale being that such local consistency will simplify the task of constructing a globally coherent model of the data. For example, the scene labeling scheme of Waltz [25] often leads to globally consistent objects, although each step examines only neighboring edges and vertices. Truth Maintenance System [9] often sacrifice completeness by limiting the inferential steps to those involving constraint-propagation [18] Such local techniques share the computational advantages mentioned earlier: there are only a few data items participating in each inference step, these items bear meaningful conceptual relationships to one another, partial results are stored exactly where they will be useful, computational steps can be performed in any order, and there is no need to remember which part of the knowledge has been processed and which part has not.

For a collection of constraints to be *globally consistent* means that it is completely explicit, namely the set of ''partial solutions'' to each subset of the constraints can always be extended to a ''full solution'' that satisfies all the constraints. This property makes globally-consistent ''constraint networks'' very attractive computationally. They admit greedy search algorithms which are guaranteed to generate a solution without any deadends (i.e., backtrack-free [12,4]). Our wish, therefore, is to enforce global consistency using as local a computation as possible.

So far all known conditions under which local consistency would entail global consistency involved topological properties of the network representing the interactions among the data items (so called ''constraint networks'') [12,5,7]. In this paper we consider additional attributes of the problem specification. We give a general relationship between the sizes of the

variables' domains, their constraints' arity and the level of local consistency required for ensuring global consistency. The results are presented for binary constraint networks first (Section 3), and then are generalized to higher arity constraints (Sections 4 and 5). Sections 6 and 7 provide examples and concluding remarks.

2. Definitions and Preliminaries

A **constraint-network (CN),** R, consists of a set of n **variables** $X_1, \ldots, X_n$, each associated with a finite **domain** of values, $D_1,\ldots,D_n$, and a set of **constraints** $\{C_1, \ldots, C_t\}$. A constraint C_i consists of two parts: 1). a subset of variables, $var(C_i) = \{X_{i_1},\ldots,X_{i_{j(i)}}\}$ called the **c-set** and 2). a relation, $rel(C_i)$ defined on this subset:

$$rel(C_i) = rel(X_{i_1},\ldots,X_{i_{j(i)}}) \subseteq D_{i_1} \times \cdots \times D_{i_{j(i)}}. \qquad (1)$$

$rel(C_i)$ consists of all simultaneous value assignments to the variables of $var(C_i)$ that are restricted by C_i alone. The **scheme** of R is the set of all its c-sets. A **solution** to R is an assignment of value to each variable such that all the constraints are satisfied. We say that the network, R, **represents** the relation, $rel(R)$ consisting of all its solutions. Namely,

$$rel(R) = \{(X_1 = x_1, \ldots, X_n = x_n) \mid \forall\, C_i\,,\ \Pi_{var(C_i)} rel(R) \subseteq rel(C_i)\} \qquad (2)$$

where $\Pi_U \rho$ stands for the projection(1) of a subset of U variables on the relation ρ. A relation, ρ, satisfying $\rho = rel(R)$ is said to be **decomposable** by R and, alternatively, R is said to be a **network decomposition** of ρ.

Typical tasks defined in connection with networks of constraints are to determine whether or not a solution exists, to find one or all solutions, and to determine whether an instantiation of some subset of variables is a part of a global solution, etc. These tasks are collectively called **constraint satisfaction problems (CSPs).**

A **binary constraint network** is one in which every c-set involves at most two variables. In this case the network can be associated with a constraint graph, where each node represents a variable and the arcs connect nodes whose variables are explicitly constrained. An r-ary CN involves constraints with arity r or less. Figure 1 shows the constraint graph of a binary con-

(1) The projection of a relation ρ on a subset of variables $U = U_1,\ldots,U_l$ is given by $\Pi_U(\rho) = \{x_u = (x_{u_1},\ldots,x_{u_l}) \mid \exists\ \bar{x} \in \rho,\ \bar{x}$ is an extension of $x_u\}$.

straint network where each node represents a variable having values $\{a, b, c\}$ and each link is associated with a strict lexicographic order (where $X_i < X_j$ iff $i < j$). (The domains and the constraints are shown explicitly on some of the links.)

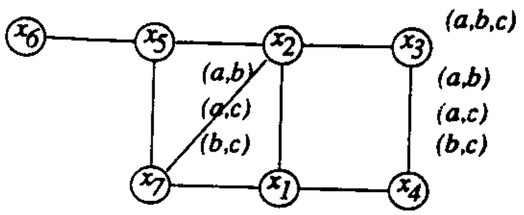

Figure 1: An example of a binary constraint network

Given a network of constraints, R, any subset, X, of its variables, determines a subnetwork R_X that contains all those constraints in R whose c-sets are restricted to variables in X. In the following paragraphs we define the notions of **local, relative** and **global consistency** which are central to this paper. Throughout the paper, n stands for the number of variables, k, for the number of values and r for the constraints' arity.

Definition 1: A partial instantiation of variables $(X_1 = x_1, ..., X_i = x_i)$ is **locally consistent** if it satisfies all the constraints in the subnetwork restricted to the participating variables.

Definition 2: Given two subnetworks R_X and R_Y s.t. $X \subseteq Y$, subnetwork R_X is **consistent relative** to subnetwork R_Y if any partial instantiation of variables in R_X which is *locally consistent* can be extended to a consistent instantiation of R_Y.

Definition 3: (global-consistency)

a. A locally consistent **instantiation** $(X_1 = x_1, ..., X_i = x_i)$ is **globally consistent** if it is *consistent relative* to the overall network.

b. A **subnetwork** R_X is **globally consistent** if all its partial assignments are *globally con-*

sistent.

c. A **network** of constraints is **globally consistent** if all its subnetworks are *globally consistent.*

Looking, for instance, at the network of Figure 1, we see that the assignment $\alpha = (X_1{=}a, X_2{=}b, X_7{=}c)$ is *locally consistent.* However, the network, $R_{\{X_1X_2X_7\}}$, is **not** *consistent relative* to $R_{\{X_1X_2X_5X_7\}}$, since the consistent assignment α cannot be consistently extended to any value of X_5. Moreover, in this case most subnetworks are not *globally consistent* since the overall network is inconsistent, thus representing the empty relation.

We will now redefine the notion of ***i*–consistency** [12] in terms of *relative consistency.*

Definition 4:

a. A network of constraints is said to be ***i*–consistent** if every subnetwork R_{i-1} of size $i-1$ is *consistent relative to* any subnetwork R_i that contains R_{i-1}. (Equivalently, any locally consistent instantiations can be extended by any i^{th} variable.) A network is **strong *i*–consistent** if it is *j*-consistent for every $j = 1,2,...i$.

b. Networks that are 2-consistent and 3-consistent are called **arc-consistent** and **path-consistent**, respectively [17, 21].

c. A binary constraint network is said to be the **minimal network** [21] if **every** subnetwork of size two is globally consistent.

The notion of a globally consistent network is a generalization of Montanari's decomposability property which was defined for the minimal network. Since a *globally consistent* network is completely explicit, its solutions can be generated in a backtrack-free manner in any ordering chosen [12, 4]. Clearly, a network is *i*-consistent for all *i* iff it is *globally consistent.*

When a network is not *i*-consistent, consistency-enforcing algorithms can be used to get it to the required consistency level [10, 3]. A brute-force algorithm for enforcing consistency of level *i* can be bounded by $O(n^{2i}k^{2i}i^r)$, and $\Omega((nk)^i i^r)$, as well as by $O((2nk)^{2i})$ and $\Omega((nk)^i)$ for unbounded *r* (see appendix for more details). Optimal algorithms for realizing the lower bound are available in [3]. For simplicity, we will use the brute-force upper bound complexity in our

analysis.

3. Local Consistency in Binary Networks

3.1 Multi-valued networks

In this and in the following sections we present a relationship between the number of values in the domain of each variable and the level of local consistency that can ensure global-consistency. Theorem 1 presents the relationship for binary networks while Theorem 5 extends it to arbitrary networks. For simplicity and without loss of generality, we assume that all variables have the same size, k, for their domains.

Theorem 1: Any k-valued binary constraint network that is strong $(k+1)$-consistent is also globally consistent.

Proof: We will prove the theorem by showing that strong $(k+1)$-consistent networks are $(k+i+1)$-consistent for any $i \geq 1$. We need to show that, if $\bar{x} = (x_1, x_2, ..., x_{k+i})$ is any locally consistent subtuple of the subset of variables $\{X_1, X_2, ..., X_{k+i}\}$, and if X_{k+i+1} is any additional variable, then there is an assignment x_{k+i+1} to X_{k+i+1} that is consistent with $\bar{x}$. Variable X_{k+i+1} may take on k possible values $\{1,2,...,k\}$ and, accordingly, we can define k sets $A_1, ..., A_k$ where A_j is the set of all assignments in $\bar{x}$ that are consistent with the assignment $X_{k+i+1} = j$. Since the network is in particular 2-consistent, and since all constraints are binary, the value x_{k+i+1} must be consistent with each assignment in the set $\{x_1, x_2, ..., x_{k+i}\}$ separately, thus, every assignment in $\bar{x}$ must appear in the set $A_1 \cup A_2, \ldots, \cup A_k$. We claim that there must be at least one set, say A_1, that spans all members of $\{x_1, ..., x_{k+i}\}$. If this were not the case, each set A_j would be missing some member, say x'_j, which means that the tuple $\bar{x'} = (x'_1, x'_2, ..., x'_k)$ would not be consistent with any one value of X_{k+i+1}. This leads to a contradiction because $\bar{x'}$, being a subset of $\bar{x}$, is locally consistent, and from the assumption of strong $(k+1)$-consistency, this tuple should be extensible by any additional variable. Note that we need not assume that the $\{x'_j\}$'s are distinct because **strong** $(k+1)$-consistency renders the argument applicable to subtuples $\bar{x'}$ of length less than k.

Assume now, without loss of generality that A_1 spans all $k+i$ values. This means that each assignment $X_j = x_j$, $j \in \{1, ..., k+i\}$ is consistent with the assignment $X_{i+k+1} = 1$, hence, we found a value $(1 \in D_{k+i+1})$ that is consistent with $\bar{x}$.

□

We may be tempted to conclude from Theorem 1 that any k-valued network can be solved polynomially by enforcing strong $(k+1)$-consistency. However, enforcing $(k+1)$-consistency may require the addition of non-binary constraints for which the theorem does not apply. We can, however, conclude a weaker result.

Corollary 1: A k-valued binary network, in which all *induced* constraints (generated by enforcing strong-$(k+1)$-consistency) are decomposable into binary constraints, can be solved in $O((nk)^{2(k+1)})$ steps.

Proof: One can enforce strong $(k+1)$-consistency in $O((nk)^{2(k+1)})$ steps [10, 17]. Then, each recorded constraint is decomposed to its minimal network (by projecting the constraint on all subsets of two variables). Since we assumed that such a binary network represents each induced constraint precisely, the intersection of all the binary constraints results in an equivalent binary network of constraints which is strong $(k+1)$ consistent. Consequently, from Theorem 1 we can conclude that the resulting network is globally consistent. Since a globally consistent network can be solved in a backtrack-free manner, the cost of generating a solution is the cost of verifying that a partial instantiation of variables satisfies all the constraints. In binary networks this can be accomplished by $O(n^2)$ constraint checks.

□

An interesting special case arises in the context of the k-colorability problem: given a set of k colors and a graph, assign a color to each node in the graph such that adjacent nodes will have different colors. This problem is clearly an instance of a k-valued binary constraint network. Moreover, since there are always enough colors (i.e., k) to consistently extend any locally consistent coloring of $k-1$ nodes (or less) by a color to one additional node, the problem is inherently **strong k-consistent**. Theorem 1 implies that if only we could extend the consistency level from k to $k+1$, the problem would become globally consistent and, hence, polynomially solvable. However,

Lemma 1: [16] A k-colorability problem is $k+1$ consistent iff each node has at most $k-1$ neighbors, namely the graph degree is bounded by k.

Proof: Clearly, if each node's degree is less then k, the problem is $k+1$-consistent. Assume now that we have a $k+1$-consistent problem and a node X having k neighbors. Assigning a unique

color to each neighbor is a locally consistent assignment which is not extensible to X, thus resulting in a contradiction.

□

The colorability example strengthen the result in Theorem 1, showing that the level of local consistency needed for ensuring global consistency should be at least $k+1$ - no lower level would suffice. This follows from observing that this instance of a k-valued binary network, is strong k-consistent while **not** globally consistent (unless it has a degree bounded by k). We can conclude, therefore, that:

Theorem 2: The (worse-case) level of strong local consistency required to ensure global-consistency for k-valued binary networks is at least $k+1$.

□

3.2 Bi-valued networks

Another interesting special case occurs when all variables are bi-valued, that is, $k = 2$. This is the only value of k for which the network remains binary after enforcing the required level of $(k+1)$-consistency. According to Theorem 1, bi-valued networks require strong 3-consistency (i.e., path-consistency) in order to be globally consistent and this level of consistency can be enforced by adding binary constraints only. Thus, we have the following corollary:

Corollary 2: A strong 3-consistent bi-valued binary network is minimal.

Proof: Theorem 1 implies that the 3-consistent problem is globally consistent, and since only binary constraints are involved, it is minimal.

It follows that: □

Theorem 3:

1. The minimal network of a bi-valued binary constraint network can be obtained in $O(n^3)$ steps.

2. A bi-valued binary constraint network can be solved in $O(n^3)$ steps.

Proof: The minimal network is obtained by enforcing strong 3-consistency which takes $O(n^3)$ steps [17]. Since the minimal network is *globally consistent*, a solution can be generated

without encountering deadends, therefore, requiring only $O(n^2)$ additional steps.[2]

□

Corollary 2 has a surprising implication regarding the expressive power of hidden variables in bi-valued binary constraint networks. Given a bi-valued relation ρ, we can generate the minimal network of ρ [21] by taking the projection of ρ on each pair of variables. The set of all solutions of this *minimal network* always contains ρ and it is the best binary network approximation of ρ. When the minimal network represents ρ exactly, we say that ρ is **binary-network-decomposable.** In general, a relation ρ is not binary-network decomposable, and the question is whether it can be decomposed using a larger network containing auxiliary bi-valued variables, or, equivalently, whether ρ can be a projection of some other relation **that is** binary-network decomposable. In [8] we showed that if a bi-valued relation is not network-decomposable, adding any number of auxiliary bi-valued variables will not remedy the situation. We will provide here an alternative proof based on Theorem 1.

Definition 5: Let ρ be any n-ary bi-valued relation over a set of variables $X = \{X_1, ..., X_n\}$. We say that relation ρ is h**–network–decomposable** if there exist h additional bi-valued variables $Y = \{Y_1, ..., Y_h\}$ for which there is a binary network $R(X,Y)$ defined over $X \cup Y$ s.t. $\rho = \Pi_X rel(R(X,Y))$. The additional variables needed for decomposition will be called **hidden variables.**

Theorem 4: A bi-valued relation that is not network-decomposable is also not h-network-decomposable, for any h.

Proof: Suppose the contrary, that ρ is a bi-valued relation that is not network-decomposable over variables $X = \{X_1, ..., X_n\}$, and let $Y = \{Y_1, ..., Y_h\}$ be a set of hidden variables such that there is a relation ρ' over $X \cup Y$ satisfying $\rho = \Pi_X \rho'$ and ρ' is network-decomposable. Since ρ' is network-decomposable, its minimal network, M, must be a binary network decomposition of ρ' and since it is minimal, it is known to be strong 3-consistent [21]. The subnetwork, M_X, of M, restricted to the subset of variables, X, is a constraint subnetwork. According to Theorem 1, since M is bi-valued strong 3-consistent binary network, any tuple which is consistent in any of its subnetwork is globally consistent. In particular, partial solutions of M_X are globally consistent

(2) Note that finding a solution to a bi-valued binary network is equivalent to the 2-satisfiability problem [13] which is known to be linear. Thus part 2 of Theorem 2 does not provide the tightest bound for this task. However, we do not know of any better algorithm for the task of generating the minimal network.

and, hence, $rel(M_X)$ is the projection of the set of all solutions ($rel(M)$) on the set of variables X, namely:

$$rel(M_X) = \Pi_X rel(M), \tag{3}$$

or, substituting, $\rho' = rel(M)$,

$$rel(M_X) = \Pi_X \rho'. \tag{4}$$

The assumption $\rho = \Pi_X \rho'$ yields,

$$\rho = rel(M_X) \tag{5}$$

which means that, M_X is an exact network decomposition of ρ, thus contradicting our initial supposition.

□

4. Local Consistency in General Networks

4.1 Constraints with higher arity

In this subsection we generalize Theorem 1 to networks having constraints of arbitrary arity.

Theorem 5: Any k-valued r-ary constraint network that is strong $(k(r-1)+1)$-consistent is globally consistent.

Proof: We will show that strong $(k(r-1)+1)$-consistency implies that the network is also $(k(r-1)+i+1)$-consistent for any $i > 0$. We need to show that, if $\bar{x} = (x_1, x_2, ..., x_{k(r-1)+i})$ is a locally consistent subtuple over variables $\{X_1, X_2, ..., X_{k(r-1)+i}\}$, and if $X_{k(r-1)+i+1}$ is any additional variable, then there is an assignment $x_{k(r-1)+i+1}$ to $X_{k(r-1)+i+1}$ that is consistent with $\bar{x}$.

Since the network has constraints of arity r or less it means that such assignment $X_{k(r-1)+i+1} = x_{k(r-1)+i+1}$ has to be independently consistent with each subset of $r-1$ assignments of the set $\{x_1, x_2, ..., x_{k(r-1)+i}\}$. The consistency of such sets is verified via the relevant constraints, having arity r or less, which are defined on the variable $X_{k(r-1)+i+1}$ and on the corresponding subset of $r-1$ variables from $\{X_1, X_2, ..., X_{k(r-1)+i}\}$. In other words, all the constraints having arity r or less, that involve variable $X_{k(r-1)+i+1}$ have to be verified in checking the consistency of any extension. Each such constraint can be uniquely determined by a subset of $r-1$ or less variables from the set $\{X_1, ..., X_{k(r-1)+i}\}$ on which it is defined.

Variable $X_{k(r-1)+i+1}$ may take on k possible values $\{1,2,...,k\}$ and, accordingly, we define k sets $A_1,...,A_k$ as follows. A_j is the set of all all subsets of assignments in $\bar{x}$ of size less or equal to $(r-1)$ that are locally consistent with the assignment $X_{k(r-1)+i+1} = j$. Note that each such subset of assignments must be locally consistent, and, since the network is strongly $(k(r-1)+1)$-consistent, and, in particular, strong r-consistent, any such partial assignment tuple must have at least one matching value in $X_{k(r-1)+i+1}$. Consequently, all partial assignments from $\bar{x}$, having length $(r-1)$ or less must appear in the set $A_1 \cup A_2, \ldots, \cup A_k$. The participation of an assignment tuple t in a set A_j means that all the constraints involving both the tuple's variables and variable $X_{k(r-1)+i+1}$ have to be satisfied. Let us denote by S the set of all constraints that may need to be verified, by $S_j(t)$ the subset of constraints involved in the consistency verification of a tuple t with the value j of $X_{k(r-1)+i+1}$, and by $S(j)$ all the constraints that are relevant to all tuples in A_j. Clearly, $S(j) = \bigcup_{t \in A_j} S_t(j)$ and $S = \bigcup_j S(j)$.

We claim that there must be at least one set, say A_1, that requires that **all the constraints** in S will be verified, namely that $S = S(1)$, or else, each set, A_j, must have a constraint $C \in S$ s.t. $C \notin S(j)$. Let $var(C) = \{X_{1j},...,X_{lj},X_{k(r-1)+i+1}\}$, $l \le r-1$, denote the constraint's *c-set* that is not verified in A_j. (Remember that constraints in S are identified by the subset of variables on which they are defined.) This means that the assignment $\bar{x}_j = (X_{1j}=x_{1j},...,X_{lj}=x_{lj})$ is not in A_j. From this it can be concluded that the assignment which results from concatenating all these *excluded* sub-assignments, $\bar{x}_1\bar{x}_2 \cdots \bar{x}_k$, is not consistent with any of the values of $X_{k(r-1)+i+1}$. However, the length of this assignment is less or equal to $k(r-1)$, and since we assumed strong $(k(r-1)+1)$-consistency, it must be consistent with at least one value, hence yielding a contradiction. Assume now, without loss of generality that $S = S(1)$. As a result, all partial tuples of $\bar{x}$ of size $r-1$ or less are consistent with the value "1" of $X_{k(r-1)+i+1}$ and we, consequently, found a value $(1 \in D_{k(r-1)+i+1})$ which is consistent with $\bar{x}$.

□

4.2 Classes of tractability

Theorem 5 supplies us with the ability to pose useful conjectures for certain families of constraint networks. When the maximum number of values is k, and the maximum constraint arity is r, we may try to prove that strong $(k(r-1)+1)$-consistency can be achieved using constraints with arity not exceeding r. If this is proved, Theorem 5 ensures that the resulting network is globally consistent hence, tractable.

The ''level of local consistency'' which is sufficient for global consistency can serve as a parameter for classifying problem instances into different classes of tractability. If we have a k-valued binary network, for instance, we can apply strong $(k+1)$-consistency to it, and if this is achievable using binary constraints only, we know that the problem is tractable and we say that it belongs to **class 0**. However, if the newly induced constraints are not ''binary decomposable'', they may be of arity k, at worse, thus resulting in a k-valued k-ary constraint network. To make this new problem globally consistent, we are advised by Theorem 5 to enforce $(k(k-1)+1)$-strong consistency. If this is achievable with k-ary constraints only, we know the problem is globally consistent and we say it is in **class 1**. Else, the resulting problem may have $k(k-1)$-ary constraints and, accordingly, we need to enforce now $k(k(k-1)-1)+1$ strong-consistency and so on. Continuing in this fashion, we can define higher classes of tractability, depending on the number of times we have to increase the consistency level until achieving global consistency.

In summary, we can divide k-valued binary constraint problems having n variables into approximately $\log_k n+1$ complexity classes. Class 0 contains those problems that can be made $(k+1)$-consistent with only binary constraints, class 1 contains problems that can be made $(k(k-1)+1)$-consistent using constraints of arity k or less, and, in general:

Definition 6: A k-valued r-ary constraint network is in class i if it can be made (K_i+1)-consistent via constraints of arity K_{i-1} or less, where K_i is defined by

$$K_i = k^i(r-1) - k^{i-1} - k^{i-2} - \dots - k. \tag{6}$$

Theorem 6: Any constraint network in class i can be solved in $O((2nk)^{2(K_i+1)})$ steps.

Proof: We will show that if a problem is in class i, we can make it globally consistent by enforcing strong (K_i+1)-consistency using $O((2nk)^{2(K_i+1)})$ steps (see appendix). As a result, for a bounded k and bounded i the problem is tractable, thus concluding the proof.

Since the problem is in class i, we know that it can be made strong (K_i+1)-consistent with constraints of arity K_{i-1} or less. From Theorem 5 it follows that since the resulting problem (after enforcing (K_i+1)-consistency) has k values and at most K_{i-1}-ary constraints, it needs be only $(k(K_{i-1}-1)+1)$-consistent to ensure global consistency. By substituting the expression for K_{i-1} (equation (6)), we get:

$$k(K_{i-1}-1)+1 = k(k^{i-1}(r-1) - k^{i-2} - k^{i-3} - \ldots - k-1)+1 \tag{7}$$
$$= k^{i}(r-1) - k^{i-1} - k^{i-2} - \ldots - k+1 = K_i + 1$$

but since this level of consistency (i.e., K_i+1) has already been enforced on the network, we know that the problem is globally consistent.

□

It is natural to ask what does it take to determine the tractability of a given problem, and we suspect that this **membership** problem is exponential. However, verifying that the problem is in a given bounded class is polynomial.

Theorem 7: Deciding whether a problem is in class i takes $O((2nk)^{2(K_i+1)})$ steps.

Proof: First, we run the problem through a strong (K_i+1)-consistency algorithm(3) which takes $O((2nk)^{2(K_i+1)}))$. The resulting problem may have some constraints of arity K_i. For each such constraint we need to determine whether it is expressible using constraint of arity not exceeding K_{i-1}. Let C be one such constraint. We then generate a network, $R_C^{(i)}$ of K_{i-1}-ary constraints by projecting C on each subset of K_{i-1} variables, an operation which is bounded by the cardinality of C, $O(k^{K_i})$. We then have to solve the resulting network $R_C^{(i)}$ in order to check if it has the same solution set as the original relation C. This corresponds to solving a constraint network problem having K_i variables and k values, which takes $O(k^{K_i})$ steps. The number of constraints, like C, that we may need to process is $O(n^{K_i})$ hence, the overall complexity is $O((2nk)^{2(K_i+1)})$.

□

Clearly, the most useful class is *class 0*; it is both the most tractable and the easiest to recognize. We now define a subclass of *class* 0 which has a specially useful feature and which is frequently encountered. As already defined, a problem is in *class*0 if it can be made $(k(r-1)+1)$-strong-consistent with r-ary constraints. If this level of consistency is achieved by enforcing only $(r+1)$-consistency, we know that the problem is globally consistent, since $(r+1)$-consistency, by definition, records at most r-ary constraints. Accordingly, we define:

Definition 7: A k-valued r-ary network that can be made globally consistent by enforcing $(r+1)$-strong-consistency, is called **regular**.

(3) Note that definition 5 is a procedural definition, based on a property of the constraints recorded by the consistency enforcing algorithm. A more ''declarative'' definition would have rendered the membership problem intractable.

Lemma 2: The complexity of regular networks is $O((2nk)^{2(r+1)})$.

□

Theorem 8: Membership in the *regular* class can be determined in $O((2nk)^{k(r-1)+2})$.

Proof: We first perform $(r+1)$ strong-consistency which is bounded by $(2nk)^{2(r+1)}$. Then we have to check if the resulting network is $(r+2)$-consistent, $(r+3)$-consistent, and so on, until we reach $(k(r-1)+1)$ consistency. If all these levels of local consistency are verified, we know, based on Theorem 5, that the problem is globally consistent. The verification of all level of local consistency between $r+2$ to $k(r-1)+1$ is bounded by

$$\sum_{i=r+2}^{k(r-1)+1} (2nk)^i = O((2nk)^{k(r-1)+2}) \tag{8}$$

□

Clearly, *regular* networks are more tractable (once recognized) and also are easier to recognize then general *class* 0 problems. Examples of such networks are provided in Section 6.

In the next subsection we will show that the complexity can be further reduced by requiring only directional consistency.

5. Directional Consistency

The notion of directional consistency was introduced in [5] as a speed up device which weakens local consistency while still ensuring global consistency along a given ordering. It allows all the constraints to be processed just once, resulting in a brute-force complexity bound of $O(n^{i+1}k^i 2^i)$ (see appendix). This weaker consistency condition can be enforced more cheaply while at the same time ensuring backtrack-free search using the selected ordering.

Definition 8: A network of constraints is said to be directional i-consistent, w.r.t. an ordering $d = X_1, ..., X_n$, if every subnetwork of size $i-1$ which is locally consistent can be consistently extended by any variable that succeeds all the subnetwork's variables in the ordering d. A network of constraints is **directional globally consistent** w.r.t. ordering d, if it is directional-i-consistent for every i.

Following is the directional version of Theorem 5.

Theorem 9: Any k-valued r-ary constraint network that is directional strong $(k(r-1)+1)$-consistent w.r.t. d is also directional globally consistent.

Proof: Same as the proof of Theorem 5.

□

For the sake of completeness we present an algorithm *adaptive-consistency* that enforces *directional consistency* to any desirable level. The algorithm can be defined in terms of a procedure **adaptive(***level*,*d***)** [6] that enforces directional strong ($level$+1)-consistency along an ordering d, where *level* is a parameter indicating the highest cardinality of constraints that are recorded, and d is an ordering $X_1, \ldots, X_n$, on the set of variables.

```
adaptive(level, $X_1, \ldots, X_n$)
Begin
1. for i=n to 1 by -1 do
2. Compute PARENTS($X_i$) /* PARENTS($X_i$) is the set of variables
   that are currently adjacent to and precede $X_i$ in $d$.
3. Perform new-record( level, $X_i$, PARENTS($X_i$))
4. for level$\geq$2, connect all elements in PARENTS($X_i$) (if they are not yet connected)
End
```

Procedure new-record(*level*, *var*, *set*) records only constraints of size less or equal to *level* from the subset *set* and is defined as follows:

```
new-record(level, var, set)
Begin
1. if level $\leq$ |set| then
2.    for every subset S in set, s.t |S | = level do
3.        record-constraint(var,S)
4.    end
5. else do record-constraint(var,set)
End
```

The procedure **record-constraint(***var*,*set***)** generates and records those tuples of variables in *set* that are consistent with at least one value of *var*. Clearly, the algorithm enforces a directional strong ($level$+1)-consistency, and its complexity is bounded by both $O((nk)^{level+1}(level+1)^r)$, and $O((2nk)^{level+1})$. The extension to *adaptive-consistency* is made by recording constraints on

all the parent-set (modifying line 3 of *adaptive*), and its complexity is bounded by both $O(k^{W^*+1} W^{*r})$ and $O((2k)^{W^*+1})$ where W^* is the maximum parent size resulting from applying the algorithm. For details see appendix and [7].

Given an ordering d of an r-ary k-valued constraint network, a problem is in **directional class** i if directional strong (K_i+1)-consistency can be enforced using constraints of arity K_{i-1} or less. As we saw, the cost of enforcing this level of directional consistency by *adaptive* (K_i+1,d) is $O((2nk)^{K_i+1})$ compared to $O((2nk)^{2(K_i+1)})$ in the undirectional version(4). We can also conclude that:

Theorem 10: Given an ordering of the variables, d, the membership of a problem in *directional class i* can be determined in $O((2nk)^{K_i+1})$ steps.

□

6. Examples

Example 1: Tasks of reasoning with time and space provide many examples of constraint networks having special local consistency properties. Allen's algebra [1], for example, defines thirteen possible relations between time intervals and provides a transitivity table for their propagation. This algebra can be described as a traditional constraint network where the variables are the relationships between two intervals, each having 13 values, and the transitivity table defines ternary constraints on triplets of variables. Having $k = 13$ and $r = 3$, we conclude (Theorem 5) that **if** we can enforce 27-strong-consistency without introducing higher than ternary constraints, we can generate a globally consistent network using a polynomial algorithm of degree 27. However, since Allen's algebra is known to be NP-complete, this is not a realistic assumption.

Example 2: A second example comes from temporal constraints limited to relationships between time points. Vilain and Kautz [24] suggested that if we consider three temporal relations between any two time points $\{<,=,>\}$ and define the transitive relationship induced by such relations, we get a simpler network that can be made globally consistent using a 3-consistent algorithm. This claim was later corrected by van Beek and Cohen [22], showing that 4-consistency is necessary for global consistency.

(4) Note, that although the optimal consistency-enforcing algorithms are asymptotically the same as the directional consistency, the "constants" which are neglected in the asymptotic analysis differ significantly in favor of the directional algorithms.

This $PA^{\neq}$ algebra (as termed by van Beek et al.) can be described as a traditional constraint network, where the variables are the relationships between two points and the transitivity table defines ternary constraints. This yields a constraint network with $k = 3$, $r = 3$. Theorem 5 suggests that if such a network is 7-strong-consistent, it is globally consistent. To make this observation useful, one has to show that it is feasible to enforce 7-consistency with ternary constraints. It can be shown that 6-consistency for our formulation can be achieved using ternary constraints, and that it implies 4-consistency in the $PA^{\neq}$, which, in turns, implies global consistency [22], and in particular, 7-consistency. It is still unclear whether the network version of this $PA^{\neq}$ algebra is a *regular* network.

Example 3: If we further simplify the point algebra and consider only two labels between time points $\{<,>\}$ (one may argue that equality never really happens), we get a ternary 2-valued constraint network. According to Theorem 5, strong 5-consistency is sufficient to ensure global consistency and, being a subset of $PA^{\neq}$ we believe that 5-consistency is achievable with ternary constraints. And, again, whether this is a regular problem is, yet, unknown.

Example 4: Consider a temporal constraint network, denoted INT^*, where the variables' domains are finite set of integers and the constraints between pairs of variables are taken from the set $\{<,\leq,=,>,\geq\}$, with no inequalities. This is a binary constraint network. Thus, if $(k+1)$-consistency can be enforced with binary constraints we have a tractable problem (Theorem 1). We will show, however, that this is a *directional-regular* problem, namely, given a certain ordering any level of directional local consistency is enforceable by directional 3-consistency. In fact, for these networks even directional 2-consistency suffices.

We associate a given problem with a directed acyclic graph as follows: the nodes denotes variables and for each constraint of the form $X < Y$ or $X \leq Y$, we direct an arc from X to Y and label it with the constraint's inequality. Cycles in this graph can be easily detected (using a connected component algorithm). If a cycle contains a strict inequality, we can conclude that there is no solution. Otherwise, we can conclude that all variables on the cycle have the same value and they can be collapsed to one, thus resulting in an acyclic directed graph.

Theorem 11: Networks in class INT^*, having an acyclic graph representation, can be made directional globally consistent using directional 2-consistency.

Proof: Given an acyclic graph representation of the problem, we can generate an ordering, d, of

the nodes which is consistent with the partial order dictated by the graph. We will show that the problem is **directional-regular** along the ordering d by showing that that enforcing directional-2-consistency is sufficient to render it $(k+1)$-directional-consistent and, therefore, directional-globally consistent (Theorem 9). Suppose we want to enforce directional $(k+1)$-consistency using algorithm *adaptive-consistency* (with *level=k*). At each step we have a parent node P, a set of child nodes $C_1, \ldots, C_t$ and a set of t constraints. Each constraint is between a C_i and P, stating either $C_i < P$ or $C_i \leq P$. We claim that by enforcing directional 2-consistency on each C_i separately, we achieve, automatically, any higher level of consistency, thus $(k+1)$-consistency. This is so since any value remaining in C_i's domain after enforcing *directional−2−consistency* with P, has the property that it is less or equal some value p_i in P. Thus, any value assignment to all $C_1, \ldots, C_t$ from these restricted domains is extensible by the $\max_i \{p_i\}$ in P.

□

van Beek [23] and independently Meiri et. al [19]. have treated the **full** point-algebra problem using similar techniques and showed same complexity bounds. However, once integer domains are introduced the full algebra (containing the ≠ relation) becomes intractable.

Example 5: A different family of constraint networks arises in the domain of scene labeling. Huffman [15] and Clowes [2] developed a basic labeling scheme for blocks world picture graphs. Given a basic labeling set: + (convex), - (concave), -> (occluding, object on arrowhead side), and a standard set of simplifying assumptions on scene content, the physically realizable junction labeling are just those shown in Figure 2. Freuder [11] provided algorithms for labeling this restricted set while Waltz [25] explored a richer label set.

A network composed of junctions in Figure 1 can be viewed in two ways: Each line can be viewed as a variable having three values while the constraints are binary or ternary depending on whether two or three lines intersect. From this view, Theorem 5 states that if a given network is also 7-strong-consistent, it is already globally consistent. The second view treats each junction as a variable, each variable has 3 or 4 values (the number of possible labeling combination of a junction) and the constraints are binary. Theorem 1 states that if the problem is 5-consistent, it is globally consistent. The second view, therefore, provided a much weaker consistency demand for guaranteeing global-consistency. Nevertheless, since this problem is known to be NP-complete, we know that there are problem instances where such consistency level cannot be enforced without increasing the constraints arity.

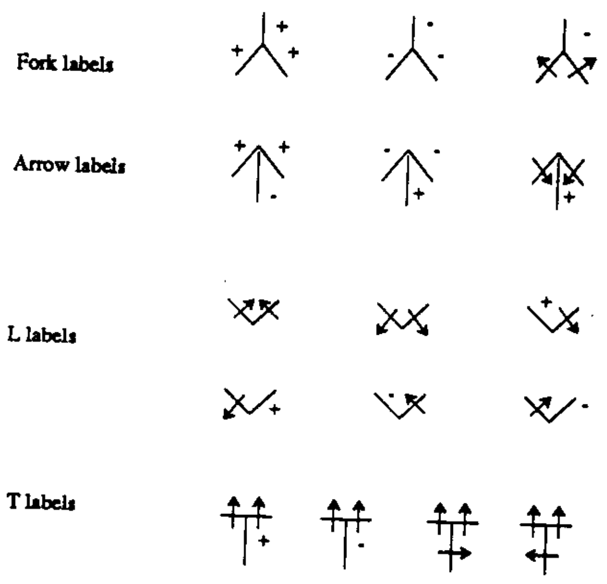

Figure 2: Junction Labels

We conclude with two additional examples of *regular* classes. One, mentioned earlier, is the class of bi-valued binary constraint networks. In this case the set of *class 0* problems and *regular* problems coincide since $k(r-1)+1 = r+1$ when k=2 and r=2. A different class are those termed ''distributive'' by Montanari [21]. He showed that when the constraints satisfy the *distributivity* property, 3-consistency will make the network globally consistent.

7. Conclusions

A globally consistent network permits the construction of a consistent solution in linear time. We showed that the amount of local consistency required for achieving global consistency is dependent on the product of two parameters: the number of values in each variable and the constraint-arity. The complexity of achieving the required local consistency is exponential in this product.

Based on this parameter we introduced a tractability classification of constraint networks, where each k-valued r-ary network falls into one of $\log_k n$ classes. A problem in class i can be solved in $O((2nk)^{2(k^i(r-1))})$ steps, and deciding whether a problem belongs to class i takes also $O((2nk)^{2(k^i(r-1))})$ steps. A special class of tractable problems, called *regular* was identified and special examples were presented.

As a consequence of our main theorems (1 and 5), we showed that if a bi-valued relation is not representable by a binary constraint network, it cannot be helped by any number of hidden variables. This could be viewed as a generalization to Peirce's relation thesis. Peirce claimed that any relation could be decomposable into ternary relationships if the domain of the hidden variable is not bounded [14]. Here we showed that when one limits the domain into two values only, no expressiveness is gained. In another paper [8] we show that if the hidden variables have three or more values in their domains, then any relation is h-network-decomposable for some h.

Acknowledgement

I would like to thank Itay Meiri for his assistance in formulating Theorem 5 and his careful comments on a final version of the manuscript. I would also like to thank Judea Pearl for his useful comments and Caroline Ehrlich for proofreading the paper.

Appendix: The complexity of brute-force consistency algorithms

Following is a worse-case analysis of a brute-force algorithm for enforcing i-consistency. Most of the analysis in the literature is for the special cases of i=2 or i=3 and for binary constraint networks [17,20]. Since we deal with general networks we will express the complexity as a function of the constraints' arity as well, while assuming that the consistency level, i, is greater than the constraint's arity, r. Following is a description of the i-consistency algorithm, which is a generalization of Mackworth's $PC-1$ algorithms [17].

Brute-force-consistency BFC(i).

```
1. Begin
2. repeat until there is no change
3.  for each subnetwork, $R_{i-1}$, of size $i-1$ on variables $X_1, \ldots, X_{i-1}$ do
4.    for each $X \in R - R_{i-1}$ do
5.      record-constraint($R_{i-1}$,$X$).
6.    end-for
7.  end-for
8. end-repeat
9. end
```

Proposition 1: Algorithm BFC is bounded by $O(n^{2i}k^{2i}i^r)$, and $\Omega((nk)^i i^r)$ steps, and, when r is unbounded it is $O((2nk)^{2i})$ and $\Omega((2nk)^i)$, respectively.

Proof: The algorithm has to process all subnetworks of size i (loop 3-7). The number of such networks is:

$$\binom{n}{i} = O(n^i). \tag{9}$$

On each subnetwork the algorithm records constraints of arity $i-1$ that ensure its consistency relative to any one additional variable (inner loop, lines 4-6). Checking the consistency of one tuple of length $i-1$ against an i^{th} variable may require checking constraints which are defined on every subset of the variables, namely, if the constraints' arity is bounded by r, the number of constraints of size r is bounded by $\binom{i}{r} = O(i^r)$, and otherwise it can be bounded by 2^i.

Since there are at most k^i tuples whose consistency is verified, processing each size-i network is bounded by both

$$O(k^i i^r) \text{ and } O((2k)^i). \tag{10}$$

From (9) and (10) we get that one (3-7) loop is bounded by $O((nk)^i i^r)$ and $O((2nk)^i)$. The number of times these loops are executed (i.e., the number of cycles through loop 2-8) until convergence, is bounded by the number of tuples of length i, namely by is $O((nk)^i)$, (assuming that only one tuple is deleted for each loop). We get, therefore, overall bounds of $O((nk)^{2i} i^{r})$, and $O(2nk)^{2i}$, for the cases that r is bounded and unbounded, respectively.

A lower bound for achieving i-consistency is derived by observing that just to verify that the network is i-consistent, require checking all the constraints, a procedure that is equivalent to one (3-7) loop in the *BFC* algorithm, thus resulting in both $\Omega((nk)^i i^r)$ and $\Omega((2nk)^i)$.

□

Directional consistency algorithms [5] allow all subnetworks to be processed exactly once, hence we can show that:

Proposition 2: i-directional consistency is both $O(n^{i+1} k^i i^r)$ and $O(n^{i+1} k^i 2^i)$ for bounded and unbouded r, respectively.

Proof: The algorithm process the constraints in a decreasing order of d each time making the subnetwork restricted to variables $1,2,...,j-1$, i-directional-consistent w.r.t. variable j. Since there are at most $O(n^i)$ subnetworks, and since each should be processed by $O(k^i i^r)$ steps, we get that making the network i-directional consistent w.r.t. one variable is $O((nk)^i i^r)$ and (if r is not bounded) $O((2nk)^i)$.

□

Proposition 3: The complexity of *adaptive-consistency* is $O(k^{W^*+1} i^r)$, and $O((2k)^{W^*+1})$, for bounded and unbouded r, respectively.

Proof: We have to process at most n constraints, each of arity bounded by W^* which yield the above bounds. For details see [7].

References

[1] Allen, J.F, ''Maintaining knowledge about temporal intervals,'' *Communications of the ACM,* Vol. 26, No. 11, November, 1983, pp. 832-843.

[2] Clowes, M. B., ''On seeing things,'' *Artificial Intelligence,* Vol. 2, 1971, pp. 79-116.

[3] Cooper, M.C., ''An optimal k-consistency algorithm,'' *Artificial Intelligence Journal,* Vol. 41, No. 1, November, 89, pp. 89-95.

[4] Dechter, R., ''Studies in the use and generation of huristics,'' UCLA, Los Angeles California, 1985. Ph.d thesis.

[5] Dechter, R. and J. Pearl, ''Network-based heuristics for constraint-satisfaction problems,'' *Artificial Intelligence,* Vol. 34, No. 1, December, 1987, pp. 1-38.

[6] Dechter, R. and I. Meiri, ''Experimental evaluation of preprocessing techniques in constraint satisfaction,'' in *Proceedings of the 11th Intl. Conf. on AI (IJCAI-89,* Detroit, Michigan: August, 1989.

[7] Dechter, R. and J. Pearl, ''Tree clustering for constraint networks,'' in *Artificial Intelligence,* 1989, pp. 353-366.

[8] Dechter, R., ''On the expressiveness of networks with hidden variables,'' in *Proceedings AAAI-90,* Boston, MA: August, 1990.

[9] Doyle, J., ''A truth maintenance system,'' *Artificial Intelligence,* Vol. 12, 1979, pp. 231-272.

[10] Freuder, E.C., ''Synthesizing constraint expression,'' *Communication of the ACM,* Vol. 21, No. 11, 1978, pp. 958-965.

[11] Freuder, E.C., ''On the knowledge required to label a picture graph,'' *Artificial Intelligence,* Vol. 15, No. 1, 1980, pp. 1-17.

[12] Freuder, E.C., ''A Sufficient condition for backtrack-free search,'' *Journal of the ACM,* Vol. 29, No. 1, January 1982, pp. 24-32.

[13] Garey, M.R. and D.S. Johnson, *Computer and Intractability, A guide to NP-Completeness*, San Francisco, California: W.H. Freeman and Company, 1979.

[14] Herzberger, H. G., ''Pierce's remarkable theorem,'' in *Pragmatism and Purpose: Essays Presented to Thomas A. Goudge,* L.W. Sumner, J. G. Slater, F. Wilson, Ed. Toronto, Canada: University of Toronto Press, 1981.

[15] Huffman, D. A., ''Impossible objects as nonsense sentences,'' in *Machine Intelligence,* B Meltzer, D. Michie, Ed. 1971, pp. 195-234.

[16] Kasif, S., *private communication,* October, 1990.

[17] Mackworth, A. K., ''Consistency in networks of relations,'' *Artificial Intelligence,* Vol. 8, No. 1, 1977, pp. 99-118.

[18] McAllester, D. A., ''An outlook on truth-maintenance,'' MIT, Boston, Massachusetts, Tech. Rep. AI Memo No. 551, August, 1980.

[19] Meiri, I. and J. Pearl, ''Faster constraint satisfaction algorithms for temporal reasoning,'' UCLA, Los Angeles, CA,, Tech. Rep. R-151, Cognitive Systems Lab, CS dept., July, 1990.

[20] Mohr, R. and T.C. Henderson, ''Arc and path consistency revisited,'' *Artificial Intelligence,* Vol. 28, No. 2, March, 1986, pp. 225-233.

[21] Montanari, U., ''Networks of constraints: fundamental properties and applications to picture processing,'' *Information Science,* Vol. 7, 1974, pp. 95-132.

[22] van_Beek, P. and R. Cohen, ''Approximation algorithms for temporal reasoning,'' in *Proceedings IJCAI-89,* Detroit, Michigan: 1989.

[23] van_Beek, P., ''Reasoning about qualitative temporal information,'' in *Proceedings Eighth National Conference on Artificial Intelligence (AAAI-90),* Boston, MA: July, 1990, pp. 728-734.

[24] Vilain, M. and H. Kautz, ''Constraint propagation algorithms for temporal reasoning,'' in *Proceedings AAAI-86,* Phila, PA: 1986, pp. 377-382.

[25] Waltz, D., ''Understanding line drawings of scenes with shadows,'' in *The Psychology of Computer Vision,* P. H. Winston, Ed. New York, NY: McGraw-Hill Book Company, 1975.