An Algebraic Approach to General Boolean Constraint Problems

Hans W. Guesgen [1]
Peter B. Ladkin [2]

TR-90-008

March 8, 1990

Abstract

We consider an algebraic approach to the statement and solution of general Boolean constraint satisfaction problems (CSPs). Our approach is to consider partial valuations of a constraint network (including the relational constraints themselves) as sets of partial functions, with the operators of *join* and *projection*. We formulate all the usual concepts of CSPs in this framework, including *k-consistency, derived constraints,* and backtrack-freeness, and formulate an algorithm scheme for *k-consistency* which has the path-consistency scheme in *[Lad-Mad88.2]* as a special case. This algebra may be embedded in the cylindric algebra of Tarski *[HeMoTa71, 85],* via the embedding of *[ImiLip84],* and a connection with relational database operations. CSPs are shown to correspond to conjunctive queries in relational database theory, and we formulate a notion of *equivalence* of CSPs with hidden variables, following *[ChaMer76, Ull80],* and show that testing equivalence is NP-hard.

---

[1] International Computer Science Institute, 1947 Center street, Berkeley, CA 94704-1105

[2] Part of this work was performed while Peter Ladkin was at The International Computer Science Institute, on sabbatical leave from Kestrel Institute. Current address: P.O. Box 7998, Berkeley, CA 94707.

1 Introduction

We consider the general algebraic formulation of general finite Boolean constraint satisfaction problems (CSPs). Suppose $x_1, \ldots . x_n$ are variables. A general Boolean CSP is given by a formula $\bigwedge_{I \subseteq \{1,\ldots,n\}} P_I(\overline{x}_I)$, where $\overline{x}_I$ is the sequence of variables $\langle x_i : i \in I \rangle$. For each i, we have a range of values D_i for x_i in mind, and each predicate symbol P_I denotes a subset S_I of $\bigotimes_{i \in I} D_i$. A *solution* of this CSP is conventionally a tuple of values $\langle a_1, \ldots, a_n \rangle$ such that, for each I, $\langle a_i : i \in I \rangle \in S_I$. For many I we may expect S_I to be a non-informative predicate, i.e. $S_I = \bigotimes_{i \in I} D_i$, so these cases are usually omitted from the formulation of the CSP.

In this paper, we consider the specification of CSPs and their algebraic structure. We find that the concepts that have proven to be useful in describing and solving CSPs may be formulated algebraically, using the concepts of partial function, join and projection of such functions. One advantage to this more formal approach is that algorithms and algorithm schemes may be formally verified, using algebraic laws over these operations. Another advantage is that results such as those of *[LadMad89]*, in which a class of hard examples for parallel path-consistency is presented, are more easily obtainable using a more mathematical approach.

Firstly, we consider the definition of all the usual concepts of CSPs, and we follow this by a simple example to illustrate the use of the concepts in the case of a small finite CSP. Next, we point out the connection with conjunctive query evaluation in relational database theory *[Ull80]*, and adapt a theorem of Chandra and Merlin *[ChaMer76]* to show that computing equivalence of CSPs with hidden variables is NP-hard. We next consider an embedding of CSPs into the cylindric algebra of Tarski *[HeMoTa71,85]*. In the context of relational databases, this embedding was due to Imielinski and Lipski *[ImiLip84]*, and we adapt a structure theorem of theirs to the case of CSPs. We consider an algorithm scheme for computing k-consistency in our framework, and prove it correct. The scheme generalises existing k-consistency algorithms in the same sense in which the path-consistency scheme of *[LadMad88.2]*, which used a relation algebraic formulation for binary networks, generalised existing path-consistency algorithms. Finally, we consider the condition for backtrack-free solutions proposed in *[Fre82]*. We show that it conrresponds to a simple equational condition in our framework.

2 Definitions and Essential Lemmas

In this section we give the definitions that are the basis for our algebraic approach. The definitions are given unillustrated, and in the next section we give an example which we hope will make these more clear.

Definition 1 *1. Suppose A is a finite CSP on variables $x_1 \ldots x_n$, with domains $D_1 \ldots D_n$. Let $1..n = \{1, \ldots, n\}$. The* domain of A *is the set $D = \{1\} \times D_1 \cup \{2\} \times D_2 \cup \ldots \{n\} \times D_n$. (In the case that $D_1 = D_2 = \ldots = D_n = E$, D is just $1..n \times E$).*

2. *An element of D is of the form $\langle k, d \rangle$, and we also denote it by $k \mapsto d$. A* partial relation *R on A is a subset of D. A* partial function *f on A is a partial relation on A that is functional, i.e. for given $k \leq n$ there is at most one pair $k \mapsto d$ in f.*

3. *The* domain of *R is the set $\{k : (\exists d)(k \mapsto d \in R)\}$. The* range of *$R$ is the set $\{d : (\exists k)(k \mapsto d \in R)\}$. Notice that f is always a total function $dom(f) \rightarrow ran(f)$.*

4. *More generally, we call a set D a* domain *if D is a collection of pairs such that $dom(D) = 1..n$ for some n.*

5. *The* restriction of *R to I, $I \lhd R$, where $I \subseteq 1..n$, is the set $\{k \mapsto d : k \in I \wedge k \mapsto d \in R\}$.*

6. *Suppose B is a set of partial functions on D. The* projection of *B on I, $\pi_I(B)$, is $\{I \lhd f : f \in B\}$.*

7. *The* join of *f and g, $f \sqcup g$, is $f \cup g$ if this is functional, else is undefined. The* join of *B and C, sets of partial functions, is the set $\{f \sqcup g : (f \in B) \wedge (g \in C)\}$, and is denoted $B \sqcup C$. The join $B_{a_1} \sqcup B_{a_2} \sqcup \ldots \sqcup B_{a_n}$, is denoted $\bigsqcup_{a \in I} B_a$, where $I = \{a_1, \ldots, a_n\}$, and we shall use other simple conditions on the subscript of $\bigsqcup$ also.*

8. *We use the notation $f : A \rightarrow B$ for total functions f from A to B, and $f : A \nrightarrow B$ for partial functions. A* constraint *is a set of partial functions on A all with the same domain. The* domain of a constraint *is the domain of its members.*

9. *A* constraint network *A is a domain D, along with a set of constraints $\{P_1, \ldots P_m\}$ with domains $I_1, \ldots I_m$ all $\subseteq dom(D)$, such that the I_k are pairwise unequal. We shall denote P_k also as P_I, where $I = I_k$. We denote such a constraint network by $\langle D, P_{I_1}, \ldots P_{I_m} \rangle$. We also use the notation $D_I = \pi_I(D)$, and in particular $D_i = D_{\{i\}}$.*

10. *Suppose $A = \langle D, P_{I_1}, \ldots P_{I_m} \rangle$ is a constraint network. $P_{I_1}, \ldots, P_{I_m}$ are the* explicit constraints *on A. The* derived constraint on *I, $I \subseteq 1..n$, is $C_I = \bigsqcup_{J \subseteq I} P_J$, unless there is nothing in the join, in which case the derived constraint is defined to be $I \lhd D$ (i.e. all possible values). The* solution set *of A is $C_{1..n}$, i.e. the derived constraint on $1..n$. The* partial solution set *on $I \subseteq 1..n$ is another term for the derived constraint on I.*

11. *A is* k-consistent *if and only if $C_I = \pi_I(C_J)$ for all I, J such that $|I| = (k-1)$, $|J| = k$, $I \subset J$.*

It is easy to check of this final definition of *k-consistency* that it is equivalent to Freuder's notion of k-consistency *[Fre78].*

There are a number of easy but essential lemmas that follow from this definition, which we shall make use of in proving later theorems. We state these lemmas here without comment, for our future use.

Lemma 1 *$\sqcup$ is an associative and commutative operation*

Lemma 2 *$C_I = \bigsqcup_{J \subseteq I} C_J$*

Lemma 3 *For any S, $\bigsqcup_{J \subseteq I} \pi_J(S) = \pi_I(S)$*

Lemma 4 *Suppose D_I is a any constraint with domain I, and E a constraint with domain $J \supseteq I$. Then $\pi_I(D_I \sqcup E) = (D_I \sqcup \pi_I(E))$*

Lemma 5 *Suppose $X_i, i \in I$ $Y_i, i \in I$ are all constraints, and for each i, X_i, Y_i have the same domain and $X_i \subseteq Y_i$. Then $\bigsqcup_{i \in I} X_i \subseteq \bigsqcup_{i \in I} Y_i$*

Lemma 6 *Let $\{D_{K_i} : i \in I\}$ be a collection of constraints, with $dom D_{K_i} = K_i$. Then $f \in \bigsqcup_{i \in I} D_{K_i}$ if and only if $\pi_{K_i}(f) \in D_{K_i}$ for each $i \in I$.*

Lemma 7 *For $I \subseteq K$, $P_I \sqcup C_K = C_K$*

Lemma 8 *If $I \subseteq K$, then $\pi_I(\pi_K(D)) = \pi_I(D)$.*

Lemma 9 *Suppose D is a constraint whose domain includes J. Then $\bigsqcup_{I \subseteq J} \pi_I(D) = \pi_J(D)$.*

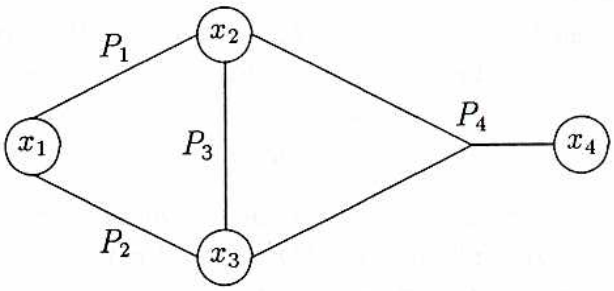

Figure 1: Constraint Network of a Coloring Problem.

3 An Example

We give in this section an illustration of the definitions above.

Consider, for example, a coloring problem represented by the constraint network shown in figure 1 with variables x_1, x_2, x_3, and x_4. Suppose that the domains of x_1, x_2, and x_3 are $\{r, g, y\}$ and that the domain of x_4 is $\{r, g, y, b\}$. We define the *domain* of the network as the union of the domains of each variable indexed with the the corresponding variable (or, rather, the subscript of the variable in this case). (Definition 1.1)

This results in the following domain:

$$\begin{array}{llll} \{\langle 1, r\rangle, & \langle 1, g\rangle, & \langle 1, y\rangle, & \\ \langle 2, r\rangle, & \langle 2, g\rangle, & \langle 2, y\rangle, & \\ \langle 3, r\rangle, & \langle 3, g\rangle, & \langle 3, y\rangle, & \\ \langle 4, r\rangle, & \langle 4, g\rangle, & \langle 4, y\rangle, & \langle 4, b\rangle\} \end{array}$$

Such a domain can also be represented using the notation $k \mapsto d$ instead of the pair notation $\langle k, d\rangle$, and is more appropriate for our purposes. On the basis of Definitions 1.1 - 1.7, the notions of a constraint and a constraint network can be defined in a straightforward way, as in Definitions 1.8 and 1.9.

To illustrate the definition of constraints and constraint network in Definition 1.8 and 1.9, we consider the following subsets of variables in the example:

$$I_1 = \{x_1, x_2\}, I_2 = \{x_1, x_3\}, I_3 = \{x_2, x_3\}, I_4 = \{x_2, x_3, x_4\}$$

These are the domains of the constraints P_1, P_2, P_3, and P_4, respectively, whereas their ranges are subsets of $\{r, g, y, b\}$.

Suppose that the constraints P_1, P_2, and P_3 demand their variables to be unequal. Then P_j is defined as follows:

$$P_j = \{\{k_1 \mapsto d_1, k_2 \mapsto d_2\} : \{k_1, k_2\} = I_j \wedge d_1, d_2 \in \{r, g, y\} \wedge d_1 \neq d_2\}$$

So far, we have defined the notions of variables, domains, constraints, and constraint networks. However, we also need a means to combine constraints in order to compute several kinds of consistencies. This is the purpose of Definitions 1.10 and 1.11, which introduce derived constraints and consistency conditions. Suppose that P_1, P_2, and P_3 are defined as above, and that P_4 is given by:

$$P_4 = \{\{2 \mapsto d_2, 3 \mapsto d_3, 4 \mapsto d_4\} : \begin{array}{l} d_2, d_3 \in \{r, g, y\} \wedge \\ d_4 \in \{r, g, y, b\} \wedge \\ (d_2 = d_4 \vee d_3 = d_4)\} \end{array}$$

Then, the derived constraint on $1..n$ is the following set:

$$\begin{array}{rl} P_{1..n} = \{\langle 1 \mapsto r, 2 \mapsto g, 3 \mapsto y, 4 \mapsto g\rangle, & \langle 1 \mapsto r, 2 \mapsto g, 3 \mapsto y, 4 \mapsto y\rangle \\ \langle 1 \mapsto r, 2 \mapsto y, 3 \mapsto g, 4 \mapsto g\rangle, & \langle 1 \mapsto r, 2 \mapsto y, 3 \mapsto g, 4 \mapsto y\rangle \\ \langle 1 \mapsto g, 2 \mapsto r, 3 \mapsto y, 4 \mapsto r\rangle, & \langle 1 \mapsto g, 2 \mapsto r, 3 \mapsto y, 4 \mapsto y\rangle \\ \langle 1 \mapsto g, 2 \mapsto y, 3 \mapsto r, 4 \mapsto r\rangle, & \langle 1 \mapsto g, 2 \mapsto y, 3 \mapsto r, 4 \mapsto y\rangle \\ \langle 1 \mapsto y, 2 \mapsto r, 3 \mapsto g, 4 \mapsto r\rangle, & \langle 1 \mapsto y, 2 \mapsto r, 3 \mapsto g, 4 \mapsto g\rangle \\ \langle 1 \mapsto y, 2 \mapsto g, 3 \mapsto r, 4 \mapsto r\rangle, & \langle 1 \mapsto y, 2 \mapsto g, 3 \mapsto r, 4 \mapsto g\rangle\} \end{array}$$

This set represents the solutions of the constraint network. Since it is expensive to compute the complete set of solutions, one is often only interested in transforming a given network into an equivalent, k-consistent one. For connections between k-consistency for various values of k, and solutions (so-called *global consistency*) see *[Fre78]*.

4 Connections to Relational Databases

Let $\langle D, P_{I_1}, \ldots P_{I_m}\rangle$ be a constraint network. Then the $P_{I_1}, \ldots P_{I_m}$ can be considered relations in a relational database, whose columns are labelled by the variables $x_1, \ldots, x_n$, alternatively by the integers $1..n$.

Solving the constraint network is equivalent to taking the join of all the relations in the database, equivalently answering the query formed by taking the conjunction of all the primitive relations.

Such conjunctive queries have been considered in *[ChaMer76, see also Ull80].* In particular, let us consider the following problem.

Suppose we consider the equivalence of two CSPs (extensional equivalence, i.e. having the same set of solutions). We extend the notion of representation by CSPs to include the notion of *hidden variables*, i.e. we consider CSPs on variables $x_1, \ldots, x_n, x_{n+1}, \ldots, x_m$ where one is only interested in the set $\pi_{1..n}(C_{1..m})$, i.e. the projection of the solutions onto the first n components only. (The variables $x_{n+1}, \ldots, x_m$ are called the hidden variables). We consider equivalence between two networks which possibly use hidden variables, and even different numbers of hidden variables, in their definition. We give the following definitions informally, but it should be clear from the informal definitions how the formal definitions should read.

Definition 2 *Let a* signature *have its usual meaning, i.e. a collection of predicate symbols, with arities assigned to each predicate symbol. Let a* type *be a signature along with a collection $x_1, .., x_n$ of n variables. Let a constraint network* scheme *be a constraint network, possibly with hidden variables, over a type, i.e. the actual relations constraining each (hyper)edge are not given, but symbols from the type are used instead, and $x_1, .., x_n$ are the explicit (non-hidden) variables. Let an* interpretation *of a constraint be an actual relation assigned to a schematic relation symbol, and an* interpretation *of a scheme be network with an interpretation of each explicit constraint. Let a* joint interpretation *of two or more schemes be interpretations of the schemes wherein the same predicate scheme in each constraint scheme is assigned the same interpretation. Call two schemes* equivalent *if they define the same relation on $x_1, ..., x_n$ for every joint interpretation.*

Theorem 1 *: It is NP-hard to determine whether two network schemes are equivalent*

Sketch of Proof: In [ChaMer76] it is shown that determining the equivalence of two conjunctive queries is NP-hard. It is straightforward to show, using the definitions of *([Ull80])*, that the problem of equivalence of network schemes is translatable into the problem of equivalence of conjunctive queries,

and vice versa.
End of Sketch.

An Algorithm for Equivalence

An algorithm for testing equivalence of conjunctive queries was given in *[Ull80]*, and this may be adapted directly to test equivalence of CSPs with hidden variables, using the translation mentioned above in the previous theorem proof sketch.

5 Connections to Cylindric Algebra

The connections between relational databases and cylindric algebra have been explored in *[ImiLip84]*. In this section, we indicate briefly how these connections work in the case of constraint networks given our definitions above. We assume some knowledge of *[ImiLip84]*, and the concept of cylindric set algebra as in *[HMTAN81]*.

In order to combine constraints on different sets of variables, we introduce the operator h which extends a constraint in all possible ways.

Definition 3 (Embedding) *Let I be a subset of the variables $\{1, \ldots, m\}$, and let C be a constraint over a domain D ($C \subseteq D^m$); then:*

$$h(C) = \{d \in D^m \mid d[I] \in C\}$$

.

For a constraint network, we consider the simultaneous interpretation of all of the constraints via the function h. This gives an embedding into a cylindric set algebra, described in the following theorem.

Theorem 2 *The function h embeds the constraint C into a diagonal-free cylindric set algebra under the interpretation of join as intersection, and projection as cylindrification over the projected variables.*

Sketch of Proof: The construction is essentially due to Imielinski and Lipski [ImiLip84], following the observation that the join and projection operators are those used in relational algebra. Their proof of the equivalent theorem for relational databases adapts directly to our case.
End of Sketch.

We may consider the definition of *derived constraint* in the context of the embedding. Using the information that join corresponds to intersection in the embedding algebra, we have the following lemma.

Lemma 10 *Let I denote a subset of the variables in a constraint network given by $A = \langle D, P_{I_1}, \ldots, P_{I_m} \rangle$ The derived constraint for I is given by*

$$C_I = \pi_I \bigcap_{I_j \subseteq I} h(P_{I_j})$$

There is considerable literature on cylindric algebras and cylindric set algebras *[ImiLip84, HeMoTa71,85, HMTAN81]*. Using the embedding defined above for a constraint network, and the solution set of a network, we may make use of cylindric algebra as a tool for investigating the algebraic structure of general constraint networks in the same way in which relation algebra was used in *[LadMad88.2]* for investigating the structure of binary constraint networks. Relation algebra and cylindric algebra are closely related, via interpretations of various sorts, and are both examples of Boolean Algebras with Operators *[Mad78, JónTar51]*.

6 K-Consistency Algorithm

In this section, we introduce a k-consistency algorithm schema and show its soundness and completeness, i.e. the algorithm really computes a k-consistent network and the solutions for this network are identical with the solutions of the original network. The algorithm schema can be implemented in various ways, in parallel as well as in serial. We repeat the definition of k-consistency from above.

Definition 4 *A is* k-consistent *if and only if $C_I = \pi_I(C_J)$ for all I, J such that $|I| = (k-1)$, $|J| = k$, $I \subset J$.*

Algorithm 1 *1. Let $A = \langle D, P_{I_1}, \ldots, P_{I_m} \rangle$ be a constraint network. Initialize $\overline{P}_I \leftarrow P_I$ for all $I \in \{I_1, \ldots, I_m\}$.*

2. Do until each $\overline{P}_I$ is stable:

(a) Select K such that $K \supseteq I$ and $|K| = k$, and let $\overline{C}_K$ be the current derived constraint on K.

(b) $\overline{P}_I \leftarrow \pi_I(\overline{C}_K)$

3. Assign $P^f_{I_j} \leftarrow \overline{P}_{I_j}$ for each $j \in \{1, \ldots, m\}$ and return $A^f = \langle D, P^f_{I_1}, \ldots, P^f_{I_m} \rangle$ as the resulting network.

It should be obvious that although the algorithm is expressed in a procedural form, it is nevertheless functional in that all variables do not change content. The pseudo-procedural presentation is just a notational convenience. We note also that the join operation in the above scheme always ranges over the same set of indices I, since every constraint in the join is projected onto this set (recall also the condition $K \supset I$, ensuring that the projection is onto all of I). In this case, the join operation is just set intersection, so another way of expressing the assigment to $\overline{P}_I$ is

$$\overline{P}_I \leftarrow \bigcap_K \pi_I(C_K).$$

We show that the algorithm does not remove solutions from the constraint network, i.e. the solutions of the original network are solutions for the k-consistent one, and vice versa, by using properties of the join operator that we have noted previously.

Lemma 11 *The constraint network resulting from algorithm 1 is equivalent to the original network, i.e. A and $\overline{A}$ have the same set of solutions.*

Proof: We need to show

$$\bigsqcup_{I \in \{I_1, \ldots, I_m\}} P_I = \bigsqcup_{I \in \{I_1, \ldots, I_m\}} P^f_I$$

We show

1. $\bigsqcup_{I \in \{I_1, \ldots, I_m\}} P_I \subseteq \bigsqcup_{I \in \{I_1, \ldots, I_m\}} P^f_I$

2. if $f \in \bigsqcup_{I \in \{I_1,\ldots,I_m\}} P_I^f$ then $f \in \bigsqcup_{I \in \{I_1,\ldots,I_m\}} P_I$

ad 1: Let $\overline{P}'_I$ be the value of $\overline{P}_I$ after the assignment $\overline{P}_I \leftarrow \pi_I(\overline{C}_K)$ and $\overline{P}_I$ be the values before this assignment; then:

$$\begin{array}{lll} \overline{P}'_I & = & \pi_I(\overline{C}_K) \quad \text{by assignment} \\ & = & \pi_I(\overline{P}_I \sqcup \overline{C}_K) \quad \text{by Lemma 7} \\ & = & \overline{P}_I \sqcup \pi_I(\overline{C}_K) \quad \text{by Lemma 4} \\ & \subseteq & \overline{P}_I \end{array}$$

So at each step, $\overline{P}'_I \subseteq \overline{P}_I$, and thus $P_I^f \subseteq P_I$ by transitivity of $\subseteq$. So $\bigsqcup_{I \in \{I_1,\ldots,I_m\}} P_I \subseteq \bigsqcup_{I \in \{I_1,\ldots,I_m\}} P_I^f$ by Lemma 5.

ad 2: Suppose $f \in \bigsqcup_{I \in \{I_1,\ldots,I_m\}} P_I$. This is equivalent to $\pi_I(f) \in P_I$ for all I. We show that

$$\forall I : \pi_I(f) \in \overline{P}_I \Longrightarrow \forall I : \pi_I(f) \in \overline{P}'_I$$

At step (b), suppose $\pi_I(f) \in \overline{P}_I$ for all I, so $\pi_K(f) \in \overline{C}_K$, so $\pi_I\pi_K(f) \in \pi_I(\overline{C}_K)$, so $\pi_I(f) \in \pi_I(\overline{C}_K)$ by Lemma 8, so $\pi_I(f) \in \overline{P}'_I$. So $\pi_I(f) \in \overline{P}_I$ is an invariant and $\pi_I(f) \in P_I$ by assignment. So $\pi_I(f) \in P_I^f$, and since this is true for all I, it is equivalent to $f \in \bigsqcup_{I \in \{I_1,\ldots,I_m\}} P_I^f$.

End of Proof.

An essential property of the algorithm is that it really computes k-consistency. This can be verified in a straightforward way:

Lemma 12 *The constraint network resulting from algorithm 1 is k-consistent.*

Proof: Suppose $|J| = k-1, |K| = k, J \subset K$. We have to show $C_J^f = \pi_I(C_K^f)$. From the termination condition of the algorithm, we know for each $I \subseteq J$ that $P_I^f = \pi_I(C_K^f)$.

$$C_J^f = \bigsqcup_{I \subseteq J} P_I^f = \bigsqcup_{I \subseteq J} \pi_I(C_K^f)$$

$$= \bigsqcup_{I \subseteq J} \pi_I(\pi_J(C_K^f)) \text{ by Lemma 8} = \pi_J(C_K^f) \text{ by Lemma 9}$$

It is easy to verify that the following condition holds after execution of the algorithm.

$$\overline{P}_I = \pi_I(C_J)$$

for any I such that $|I| \leq k-1$ and $J = I \cup \{j\}$ for some j. Now

$$C_I = \bigsqcup_{I' \subseteq I} P_{I'} = \bigsqcup_{I' \subseteq I} \pi_{I'}(C_J) = \pi_I(C_J)$$

The first equality follows by definition of the derived constraint, the second by the property mentioned above, and the final by an earlier lemma. The equality of the first and last expressions is the definition of k-consistency. **End of Proof.**

7 Backtrack-Free Search

When looking for solutions of a constraint network, the question often arises whether there is a backtrack-free search, i.e. is there an order on the variables which guarantees a search without backtracking regardless of the order on the values. This question was first discussed in [Fre82]. We will reformulate the definition of backtrack-freeness in our framework and show that it is equivalent to Freuder's definition.

First, we will denote an order on the variables of a constraint network by a sequence of domains. Let $D_1, \ldots, D_n$ be the domains of the variables $x_1, \ldots, x_n$. Suppose the order $x_{i_1}, \ldots, x_{i_n}$ is given with $i_j \in 1..n$ and $i_j \neq i_k$ for $j \neq k$. Then, this order can be denoted as sequence of domains, $I_1, \ldots I_n$, such that

$$\begin{aligned} I_1 &= \{i_1\} \\ I_k &= I_{k-1} \cup \{i_k\} \quad \text{for } k \in 2..n \end{aligned}$$

A necessary and sufficient condition for backtrack-free search is that the derived constraint on each I_k is equal to the projection of the derived constraint C_{I_n} on I_k. The following lemma formalizes this condition:

Lemma 13 *An order on the variables, given by $I_1, \ldots, I_n$, is backtrack-free, if and only if:*

$$\forall k \in 1..n : C_{I_k} = \pi_{I_k}(C_{I_n})$$

Proof: Assume that there is some k such that $C_{I_k} \neq \pi_{I_k}(C_{I_n})$, then:

$$\exists f \in C_{I_k} \text{ such that } \forall g \in C_{I_n} : f \sqcup g \text{ is undefined}$$

Thus, the assignment of values to variables given by f cannot be extended to a complete value assignment, contradicting backtrack-freeness.

For the other direction, it is sufficient to observe that if $C_{I_k} = \pi_{I_k}(C_{I_n})$, then every element of C_{I_k} may be extended to an element of C_{I_n}, and this is just the condition for backtrack-freeness.

End of Proof.

References

HMTAN81 : Henkin, L., Monk, D., Tarski, A., Andréka, H., and Németi, I., *Cylindric Set Algebras*, Lecture Notes in Mathematics 883, Springer, 1981.

ChaMer76 : Chandra, A.K., and Merlin, P.M., *Optimal Implementation of Conjunctive Queries in Relational Databases*, Proceedings of the Ninth Annual Symposium on the Theory of Computing, ACM Press, 1976.

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

Fre82 : Freuder, E.C., *A Sufficient Condition for Backtrack-Free Search*, Journal of the ACM 32 (4), Jan 1982, 24-32.

HeMoTa71 : Henkin, L., Monk, D., and Tarski, A., *Cylindric Algebras, Part I*, North-Holland 1971.

HeMoTa85 : Henkin, L., Monk, D., and Tarski, A., *Cylindric Algebras, Part II*, North-Holland 1985.

ImiLip84 : Imielinski, T., and Lipski, W., Jr., *The Relational Model of Data and Cylindric Algebras*, Jour. of Computer and System Sciences 28, 1984, 80-102.

JónTar51 : Jónsson, B. and Tarski, A., *Boolean Algebras with Operators I*, American J. Mathematics (73), 1951.

JónTar52 : Jónsson, B. and Tarski, A., *Boolean Algebras with Operators II*, American J. Mathematics (74), 1952, 127-162.

LadMad88.2 : Ladkin, P.B., and Maddux, R.D., *On Binary Constraint Networks*, Kestrel Institute Technical Report KES.U.88.8.

LadMad89 : Ladkin, P.B., and Maddux, R.D., *Parallel Path-Consistency Algorithms for Constraint Satisfaction*, Technical Report TR-89-045, International Computer Science Institute, Berkeley.

Mad78 : Maddux, R.D., *Topics in Relation Algebras*, Ph. D. Thesis, University of California at Berkeley, 1978.

Ull80 : Ullman, J.D., *Principles of Database Systems*, first edition, Computer Science Press, 1980.