Conditional interchangeability and substitutability [*]

Yuanlin Zhang [†] and **Eugene C. Freuder**
Cork Constraint Computation Center
University College Cork
Cork, Republic of Ireland
{y.zhang, e.freuder}@4c.ucc.ie

Abstract

In a constraint satisfaction problem, two values of a variable are interchangeable if every solution involving one value remains a solution with the value replaced by the other one. Although interchangeability occurs in many problems, there are also problems where little interchangeability occurs. In this paper, we study *conditional interchangeability and substitutability* for problems where values are interchangeable or substitutable under certain conditions.

1 Introduction

A *binary Constraint Satisfaction Problem (CSP)* consists of a set of variables, each of which has a finite domain, and a set of binary constraints on these variables. A solution to a CSP is an assignment of values to variables such that all constraints are satisfied.

Given a constraint satisfaction problem, two values of a variable are *interchangeable* [Freuder, 1991] if every solution involving one value remains a solution when the value is replaced by the other one. Derived from this concept of interchangeability are concepts of substitutability, partial interchangeability, and functional interchangeability etc. These variations are present in many real-world problems and help to solve, abstract, and compile CSPs [Freuder and Sabin, 1997; Rainer Weigel, 1999]. However, for some problems, there is little or no interchangeability. For example, consider a CSP with only one constraint $x < y$ with $x, y \in [1..10]$. No values of x (or y) are interchangeable, but if we know that $y > 6$, the values from 1 to 6 of x are interchangeable with each other. In other words, under certain conditions, some values of a variable become interchangeable.

In this paper, we introduce conditional interchangeability (CI) and conditional substitutability (CS), and their restricted version to neighboring variables. We also present alternative ways to express the 'condition' and compare this work with the existing work.

[*]This work has received support from Science Foundation Ireland under Grant 00/PI.1/C075.

[†]Current address: Department of Computer Science, Texas Tech University, Lubbock, Texas. Email: yzhang@cs.ttu.edu

2 Conditional interchangeability and substitutability

A CSP is usually represented as a triple (V, D, C), where V is a set of variables $\{x_1, \ldots, x_n\}$, D a set of domains $\{D_1, \ldots, D_n\}$, and C a set of constraints. A constraint between x and y is denoted by c_{xy}. In this paper, the values of a variable are referred to by letters of $a, b, c, d, e, f, \ldots$. d is also used to denote the size of the maximum domain in a CSP. We assume that a constraint is given explicitly by a set of allowed tuples. For simplicity, $x.a$ refers to value a of variable x.

Definition 1 *A* condition *on a set of variables X is defined as a set of constraints on these variables X.*

The idea of conditional interchangeability is to consider the interchangeability under a solution space restricted by certain conditions, as shown by the example of $x < y$ in the first section whose solution space is reduced by the condition $y > 6$.

Definition 2 *Given a CSP (V, D, C) and a variable x, two values a, b of x are* conditionally interchangeable (CI) *under condition Con iff they are interchangeable in $(V, D, C \cup Con)$.*

In this section, a condition is assumed to consist of primitive constraints of the form $x = a$ where x is a variable and a a value, and logical conjunctions and disjunctions.

Example 0 Assume a CSP with a list of variables $< x, y, z >$ has a solution space

$$\{(a, a, a), (a, b, c), (b, b, c), (a, c, c), (b, c, c), (b, b, b)\}.$$

A tuple, e.g., (a, b, c), is an instantiation of the variables $< x, y, z >$ in that order. The values a and b of x are not interchangeable due to the existence of the solutions (a, a, a) and (b, b, b). After exposing the condition of $y = b \wedge z = c$ or $y = c \wedge z = c$, these two solutions are excluded, and thus values a, b of x are interchangeable. It can be written as

$$(y = b \wedge z = c \vee y = c \wedge z = c) \rightarrow x.a \equiv x.b$$

where $\equiv$ denotes interchangeability.

It is straightforward to verify the following observation.

Proposition 1 *Given a condition, CI is reflective, symmetric, and transitive.*

As such, CI provides a way to put the values of a variable into equivalent groups.

Substitutability is defined in [Freuder, 1991] as follows. Given a CSP and two values a and b of a variable x, a is substitutable for b if any solution involving b remains a solution after a is substituted for b. Similar to the interchangeability under conditions, an otherwise non-substitutable value could become substitutable under some condition.

Definition 3 *Given a CSP (V, D, C) and two values a and b of a variable x, a is* conditionally substitutable *for b under a condition Con iff it is substitutable for b in $(V, D, C \cup Con)$.*

Consider Example 0 again. Neither a nor b of x is substitutable for the other. After introducing the condition

$$y = b \wedge z = c \textit{ or } y = c \wedge z = c \textit{ or } y = a \wedge z = a,$$

solution (b, b, b) is excluded and thus a is substitutable for b. However, under this condition, b is not substitutable for a and consequently not interchangeable with b. The fact that b is conditionally substitutable for a is denoted by

$$(y = b \wedge z = c \vee y = c \wedge z = c \vee y = a \wedge z = a) \rightarrow a \preceq b,$$

where $\preceq$ means "substitutable for".

Conditional substitutability describes a relationship between values of a variable, and has the following property.

Proposition 2 *Given a condition Con, conditional substitutability is reflexive and transitive.*

The following stronger substitutability will be useful later.

Definition 4 *Given a CSP and a variable x, a value a of x is* completely substitutable *if it is substitutable for any other value in the domain of x.*

The conditional version of this concept is given below.

Definition 5 *Given a CSP (V, D, C) and a variable x, a value a of x is* conditionally completely substitutable *under condition Con iff it is completely substitutable in $(V, D, C \cup Con)$.*

Since the solution space of a CSP is not known a priori, it is usually not easy to identify the (conditionally) interchangeable values in the problem. In the next section, we study the conditional interchangeability of the values of a variable x by considering only the constraints between x and its neighbors.

3 Conditional Neighborhood Interchangeability

We first explain some notations on the neighborhood of a variable. Given a constraint c_{xy}, value $a \in D_x$ is *compatible* with $b \in D_y$ if $(a, b) \in c_{xy}$ and a is also called a *support* of b. Two variables are neighbors if there is a constraint between them. $N(x)$ is used to denote an ordered list of all neighboring variables of x: $< x_1, \ldots, x_l >$. Given a CSP and a variable x, the *neighborhood subproblem* on x refers to the problem of variable x, its neighbors, and the constraints between x and its neighbors. Note that the subproblem of x does not include any constraints between its neighbors. Once a CSP is restricted to a neighborhood subproblem, it is easy to recognize the conditions for the values to be interchangeable.

Definition 6 *Given a CSP, two values a and b of a variable x are* conditionally neighborhood interchangeable (CNI) *under condition Con iff under Con, a and b are interchangeable with respect to the neighborhood subproblem on x.*

The conditional interchangeability has the following relationship with CNI.

Proposition 3 *Two values of a variable are conditionally interchangeable under a condition Con if they are conditionally neighborhood interchangeable under Con.*

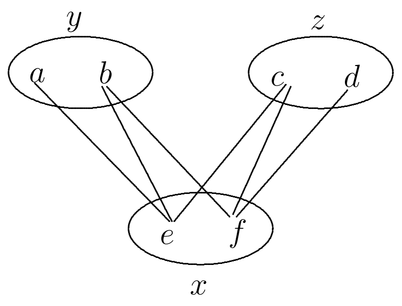

Figure 1: A CSP instance. The big circle represents a domain of a variable. The alphabets inside a circle are values. Two values are compatible if there is an edge between them. The constraint between two variables are exactly the set of all the edges connecting values of these variables.

Example 1 Consider the neighborhood subproblem, of a CSP, on x in Fig 1. values $e, f \in D_x$ are CNI under the condition $y = b \wedge z = c$. Hence,

$$y = b \wedge z = c \rightarrow e \equiv f.$$

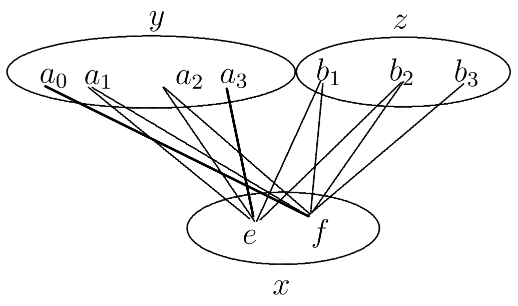

Figure 2: Values e and f are CNI under many conditions

Example 2 In Fig 2, e and f share many supports in y and z respectively. They are CNI under any one of the four instantiations of (y, z): (a_1, b_1), (a_1, b_2), (a_2, b_1), and (a_2, b_2). We could have

$$y = a_1 \wedge z = b_1 \rightarrow e \equiv f,$$
$$y = a_1 \wedge z = b_2 \rightarrow e \equiv f,$$
$$y = a_2 \wedge z = b_1 \rightarrow e \equiv f,$$
$$y = a_2 \wedge z = b_2 \rightarrow e \equiv f.$$

Here we are interested in finding all conditions under which two values are CNI. Consider two values a and b of a variable x_i. Let $< x_1, \dots, x_l >$ be the neighbors of x_i. For a neighboring variable $x_j (j \in 1..l)$, let $SS_j(\{a, b\})$ be the shared supports of a and b with respect to the constraint c_{ij}. Every tuple in $SS_1(\{a, b\}) \times \cdots \times SS_l(\{a, b\})$ is a condition for a and b to be CNI. In the example above $SS_y(\{e, f\}) = \{a_1, a_2\}$ and $SS_z(\{e, f\}) = \{b_1, b_2\}$. The number of conditions can be exponential to the number of neighboring variables. Given the fact that all the conditions for a and b to be CNI are from a Cartesian product of their shared supports, the conditions can be simplified as

$$x_1 \in SS_1 \wedge \cdots \wedge x_l \in SS_l \rightarrow a \equiv b. \quad (1)$$

Specifically, in Example 2, we have

$$y \in \{a_1, a_2\} \wedge z \in \{b_1, b_2\} \rightarrow e \equiv f. \quad (2)$$

It can be shown that the shared supports of two values provide the "weakest" condition under which they are CNI.

Proposition 4 *To represent the weakest condition for two values of x to be CNI, we need a space of size $\mathcal{O}(|N(x)|d)$ where d is the size of the maximum domain of the problem of concern.*

Now let us turn to the neighborhood substitutability under conditions.

Definition 7 *Given a CSP and two values a and b of a variable x, value a is* conditionally neighborhood substitutability (CNS) *for b under condition Con iff under Con, a is substitutable for b with respect to the neighborhood subproblem on x.*

Example 3 Consider the CSP in Fig 3. Neither e nor f of variable x is substitutable for the other. If we restrict y to be in $\{a_1, a_2, a_3\}$, i.e., $y \in \{a_1, a_2, a_3\}$, value e is substitutable for f. Since e and f share the same supports in z, it is not necessary to put any condition on z for e to be substitutable for f. The substitutability of a for b under the condition above can be expressed as

$$y \in \{a_1, a_2, a_3\} \rightarrow e \preceq f.$$

Is there a condition under which e is completely substitutable (i.e., substitutable for f and g of x)? For this example, it can be verified that

$$y \in \{a_1, a_2, a_3\} \wedge z \in \{b_1, b_2\} \rightarrow e \preceq f \wedge e \preceq g.$$

In fact, we can identify a general condition for any value a of a variable x to be completely substitutable: If each neighboring variable of x contains only values that are supports of a, a is then substitutable for any other value of x.

Proposition 5 *Given a CSP and a value a of a variable x, for any neighboring variable x_j of x, let S_j be the set of supports for a with respect to the constraint on x and x_j. a is completely substitutable under the condition*

$$x_1 \in S_1 \wedge x_2 \in S_2 \wedge \cdots \wedge x_l \in S_l.$$

Conditional neighborhood substitutability implies conditional substitutability.

Proposition 6 *Given a CSP and two values a and b of x, if a is conditionally neighborhood substitutable for b under condition Con, a is conditionally substitutable for b under condition Con.*

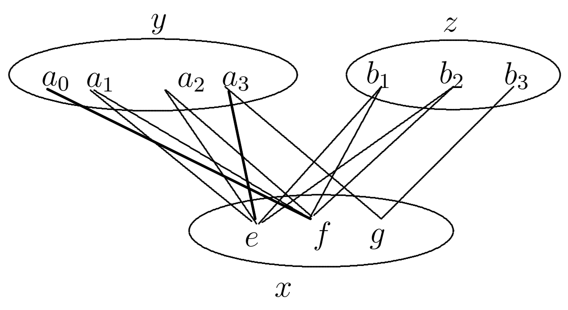

Figure 3: More complicated constraint c_{yx}

4 Extension of conditional interchangeability and substitutability

The interchangeability and substitutability of *values* of a variable can be generalized to those of *instantiations* of a set of variables. Since a set of variables can be regarded as one variable whose values are consistent instantiations of the original variables, it is not difficult to extend the conditional interchangeability and substitutability of values to those of instantiations.

Given a list of variables X, an instantiation of X is *consistent* iff it satisfies all the constraints involving only variables in X. We use $\bar{a}, \bar{b}$, ... to refer to an instantiation of X.

We are interested in the set of variables each of which is outside X and is a neighbor of some variable in X. These variables are called neighbors of X and denoted by $N(X)$.

Two consistent instantiations of X are *interchangeable* iff every solution involving one instantiation remains a solution with the instantiation replaced by the other one.

Definition 8 *Given a CSP (V, D, C) and a set of variables X, let $\bar{a}$ and $\bar{b}$ be two consistent instantiations of X. $\bar{a}$ and $\bar{b}$ are* conditionally interchangeable *under condition Con iff they are interchangeable in CSP $(V, D, C \cup Con)$.*

The conditional substitutability of a consistent instantiation is defined below.

Definition 9 *Given a CSP (V, D, C) and a set of variables X, let $\bar{a}$ and $\bar{b}$ be two consistent instantiations of X. $\bar{a}$ is* conditionally substitutable *for $\bar{b}$ under condition Con iff it is substitutable for $\bar{b}$ in CSP $(V, D, C \cup Con)$.*

Given a CSP, the neighborhood subproblem on X is the one consisting of X and $N(X)$, the constraints involving variables in X, and the constraints involving one variable in X and the other in $N(X)$. In other words, the subproblem on X includes constraints on X and constraints connecting a variable in X with a neighbor of X. Now we are able to list the neighborhood version of conditional interchangeability and substitutability.

Definition 10 *Given a CSP and two consistent instantiations $\bar{a}$ and $\bar{b}$ of a set of variables X, $\bar{a}$ and $\bar{b}$ are* conditionally neighborhood interchangeable *under condition Con iff under Con, $\bar{a}$ and $\bar{b}$ are interchangeable with respect to the neighborhood subproblem on X.*

Definition 11 *Given a CSP and two consistent instantiations $\bar{a}$ and $\bar{b}$ of a set of variables X, $\bar{a}$ is* conditionally neighborhood substitutable *for $\bar{b}$ under condition Con iff under Con, $\bar{a}$ is substitutable for $\bar{b}$ with respect to the neighborhood subproblem on X.*

In the following we discuss conditions to make instantiations interchangeable or substitutable.

For any consistent instantiation $\bar{a}$ of X, and a neighbor $y_i \in N(X)$, let $S_i(\bar{a})$ be the set of values of y_i that are compatible to $\bar{a}$, i.e., each value of $S_i(\bar{a})$ and $\bar{a}$ satisfy the constraints between y_i and any variable in X. Given two instantiations $\bar{a}$ and $\bar{b}$ of X and a neighbor y_i, their shared support set, denoted by $SS_i(\{\bar{a}, \bar{b}\})$, is the intersection of the support set of $\bar{a}$ and that of $\bar{b}$. Assuming that $N(X) = \{y_1, y_2, \ldots, y_m\}$, we are able to list the general conditions for neighborhood interchangeability

$$y_1 \in SS_1 \wedge y_2 \in SS_2 \wedge \cdots \wedge y_m \in SS_m \rightarrow \bar{a} \equiv \bar{b}$$

where SS_i, $i \in 1..m$, refers to $SS_i(\{\bar{a}, \bar{b}\})$.

Under the condition

$$y_1 \in S_1(\bar{a}) \wedge y_2 \in S_2(\bar{a}) \wedge \cdots \wedge y_m \in S_m(\bar{a}),$$

$\bar{a}$ is completely substitutable, i.e., substitutable for every other consistent instantiation of X.

5 On the application of conditional interchangeability and substitutability

In the case of checking the satisfiability of a CSP, conditional interchangeability and substitutability may be used to prune the search space. The conditional interchangeability, as an equivalence relation, partitions the values of a variable into equivalent groups. Given a group of interchangeable values (under certain condition), we can choose to keep only one value of the group in the domain of the variable while not affecting the satisfiability of the original problem. The reason is that if there is any solution including another member of the group, it remains a solution if we replace the member by the value we choose to keep.

Consider a value a of a variable x. Assume x has l neighbors, for neighbor $x_i (i \in 1..l)$, S_i is the support set of a, and SS_i is the shared support set of a and another value b of x.

By the condition in (1), if we have

$$x_1 \in SS_1 \wedge \cdots \wedge x_l \in SS_l,$$

we can keep a and prune b and all other values which are interchangeable with a under this condition.

By the condition in Proposition 5, conditional neighborhood substitutability results in

$$x_1 \in S_1 \wedge x_2 \in S_2 \wedge \cdots \wedge x_l \in S_l \rightarrow x = a. \qquad (3)$$

Concerning the pruning ability, the CNS is clearly more powerful than CNI. CNI can only remove values interchangeable with a while CNS removes all other values. Furthermore, each SS_i is a subset of S_i, implying that the CNS provides a weaker premise than CNI.

In the rest of this section, we study the relationship between the concepts here and those in the work reported in [Bowen and Likitvivatanavong, 2004; Prestwich, 2004; Chmeiss and Sais, 2003].

5.1 Domain transmutation

The work by Bowen and Likitvivatanavong embeds the idea of conditional neighborhood interchangeability without explicitly introducing condition. They introduce the concept of domain transmutation by splitting a value into two or merging two values in terms of CNI. In other words, [Bowen and Likitvivatanavong, 2004] creates virtual values for those conditions making values interchangeable.

Consider Example 1 (Fig 1). For values e and f of x, their shared supports are $SS_y = \{b\}$, $SS_z = \{c\}$. We know that

$$y \in SS_y \wedge z \in SS_z \rightarrow e \equiv f.$$

But we can not remove either e or f because they are interchangeable only when the condition holds. The method in [Bowen and Likitvivatanavong, 2004] simply introduces a new value, say g, whose support sets are exactly SS_y and SS_z. Now that some role of e has been assumed by y, we only need to figure out the *other* role e plays when the condition is not true (i.e., $y \neq b \vee y \neq c$). When $y \neq b$, e is supported by a of y and c of z, but when $z \neq c$, e is not supported by any value of z and thus all its supports in y are useless. See the picture in Fig. 4 for the supports of e. Similarly, when the condition is false, the supports of f are $\{b\}$ (for y) and $\{d\}$ (for z).

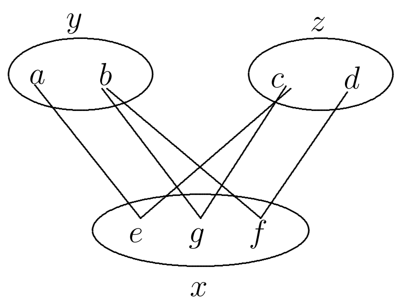

Figure 4: New value g is introduced and the constraints c_{xy} and c_{xz} are updated to reflect the conditional neighborhood interchangeability

5.2 "Interchangeability constraints"

Since the work in [Prestwich, 2004] is based on a different representation of constraints, to establish the connection, we first explain a "nogood" (disallowed) representation of a constraint.

Consider the constraint c_{yx} in Example 2 (Fig 2). The disallowed tuples by c_{yx} are:

$$\{(a_0, e), (a_3, f)\}.$$

That (a_0, e) is not allowed is expressed by $\neq$ and logical connectives

$$y \neq a_0 \vee x \neq e.$$

Similarly, disallowing (a_3, f) is expressed as

$$y \neq a_3 \vee x \neq f.$$

To represent c_{yx}, we connect the two formulae above by and:

$$(y \neq a_0 \vee x \neq e) \wedge (y \neq a_3 \vee x \neq f). \tag{4}$$

Now a constraint is represented by a formula with conjunctive normal form, instead of a set of allowed tuples. Consequently, the conjunction of all constraints of CSP is still of conjunctive normal form. The CSP in Example 2 is represented as

$$\begin{array}{l} (y \neq a_3 \vee x \neq e) \wedge (y \neq a_0 \vee x \neq f) \ \ //c_{yx} \\ \wedge \\ (z \neq b_3 \vee x \neq e) \ \ //c_{zx} \end{array}$$

We are now able to introduce the idea to prune values in [Prestwich, 2004]. Given a variable x and a term $x \neq a$, we first select all conjuncts containing $x \neq a$ from all constraints (for simplicity, we consider only binary constraints here although non binary constraints can be treated similarly)

$$\begin{array}{ccc} x \neq a & \vee & x_1 \neq a_{11} \\ & \vdots & \\ x \neq a & \vee & x_1 \neq a_{1n_1} \\ x \neq a & \vee & x_2 \neq a_{21} \\ & \vdots & \\ x \neq a & \vee & x_2 \neq a_{2n_2} \\ x \neq a & \vee & x_l \neq a_{l1} \\ & \vdots & \\ x \neq a & \vee & x_l \neq a_{ln_l} \end{array} \tag{5}$$

Note, in the formulae above, we group the conjuncts with the same variables together. For instance, the first group involves x and x_1.

By assuming an ordering "$\leq$" on the values in the domain of x, Prestwich gives the following pruning "constraint" [Prestwich, 2004] for all b such that $a \leq b$,

$$\begin{array}{c} x_1 \neq a_{11} \wedge \cdots \wedge x_1 \neq a_{1n_1} \wedge \\ x_2 \neq a_{21} \wedge \cdots \wedge x_2 \neq a_{2n_2} \wedge \cdots \wedge \\ x_l \neq a_{l1} \wedge \cdots \wedge x_l \neq a_{ln_l} \rightarrow x \neq b. \end{array} \tag{6}$$

This constraint is named by Prestwich as **interchangeability constraints (IC)**. Its premise is the conjunctions of all the conjuncts in (5) with $x \neq a$ removed. Since conjuncts in (5) include *all* those involving $x \neq a$, if the premise of (6) is true, we can simply let x be a, which makes all other $\neq$'s on x (in the whole CSP of concern) true and thus the satisfiability of the whole problem is not affected. The IC can be strengthened as follows

$$\begin{array}{c} x_1 \neq a_{11} \wedge \cdots \wedge x_1 \neq a_{1n_1} \wedge \\ x_2 \neq a_{21} \wedge \cdots \wedge x_2 \neq a_{2n_2} \wedge \cdots \wedge \\ x_l \neq a_{l1} \wedge \cdots \wedge x_l \neq a_{ln_l} \rightarrow x = a. \end{array} \tag{7}$$

In fact, with the traditional representation of constraints in mind, $x_1 \ldots x_l$ are exactly the neighbors of x, and $a_{i1} \ldots a_{il}$ are the values which are not consistent with a of x. The constraint (7) could be written as

$$\begin{array}{c} x_1 \notin (D_1 - S_1) \wedge x_2 \notin (D_2 - S_2) \wedge \cdots \wedge \\ x_l \notin (D_l - S_1) \rightarrow x = a \end{array} \tag{8}$$

where S_i is the support set of a of x with respect to x_i, as defined as before. Obviously, this pruning constraint is equivalent to (3) that is derived from CNS directly.

In summary, the conditional neighborhood substitutability facilitates stronger pruning and more intuition than IC's.

5.3 Generalized neighborhood substitutability

In this subsection, we switch back to the traditional representation of a constraint as a set of allowed tuples.

The generalized neighborhood substitutability (GNS) proposed in [Chmeiss and Sais, 2003] says that two values of a variable is GNS iff they share at least one support with respect to each neighboring variable.

There is a relationship between GNS and CNI.

Proposition 7 *Two values are GNS iff there exists a condition Con such that they are CNI under Con.*

However, when a value is CNS for another value, these two values might not be GNS. For example, consider e and g of x in Fig. 3. e is CNS for g, but they do not share any support in the domain of variable z and thus they are not GNS.

A "constraint" to prune search space is also proposed in [Chmeiss and Sais, 2003]. By assuming a total ordering on the values, the key component of that pruning constraint on a variable x can be translated, by using notations developed here, to

$$\begin{array}{c} x_1 \in S_1 \wedge x_2 \in S_2 \wedge \cdots \wedge x_l \in S_l \\ \rightarrow (x = a_1 \vee x = a_2 \vee \cdots \vee x = a_m \vee x = a) \end{array} \tag{9}$$

where a is a value of x, $a_1 \cdots a_m$ are the values smaller than a, $x_1 \ldots x_l$ are the neighbors of x, and again S_i is the support set of a with respect to x_i.

It is interesting to observe that this constraint is equivalent to the IC (6) if the same ordering on values is used in both constraints.

As we have been aware, under the premise of (9), a is completely substitutable for every other value of x, and thus the pruning constraint can be strengthened by letting $x = a$, equivalent to (3).

There is no obvious connection between the constraint (9) and GNS. But the relationship between the specific pruning constraints (3) and CNS is immediate.

6 Other related work

6.1 Partial interchangeability

Two values of a variable x are *partially interchangeable* [Freuder, 1991] with respect to a set of variables X iff any solution involving one implies a solution involving the other with possibly different values for X. Partial interchangeability is a special type of conditional interchangeability where the condition is on the assignments of X.

In this section, we present a result on a property of partial interchangeability. If x is not a neighbor of any variable in X, partial interchangeability is equivalent to interchangeability.

Proposition 8 *Given two values a and b of a variable x and a set X, if a and b are partially interchangeable with respect to X, and x is not a neighbor of X, then a and b are interchangeable.*

Proof. Consider any solution where x takes value a . Let $\bar{a}$ be the list of values for X in the solution and a' the list of values for other variables in the solution. We represent the solution by a list $(a, \bar{a}, a')$. We need to prove that $(b, \bar{a}, a')$ is also a solution.

Since a and b are partially interchangeable, there exist a list of values $\bar{b}$ for X such that $(b, \bar{b}, a')$ is a solution. The fact that x is not a neighbor of any variable in X implies that the neighbors (recalling the definition of the neighbors of a set of variables) of X have the same values in both solutions. Both $\bar{a}$ and $\bar{b}$ are consistent with the values taken by their neighbors. Hence, replacing $\bar{b}$ in the second solution with $\bar{a}$, $(b, \bar{a}, a')$ is still a solution.

Similarly, we can show that any solution containing b remains a solution by substituting a for b. Hence, they are interchangeable. □

6.2 Context dependent interchangeability

Context dependent interchangeability (CDI) [Weigel *et al.*, 1996] is equivalent to conditional interchangeability. Instead of focusing on the neighborhood of a variable, [Weigel *et al.*, 1996] resorts to a rather sophisticated decomposition method to identify CDI values under certain "conditions".

The identification of conditional *neighborhood* interchangeability is at least tractable for binary CSPs and could be a practical tool to prune the search space.

6.3 Domain partition

The idea in [Haselbock, 1993] is that although two values of a variable may not be neighborhood interchangeable, they could be (fully) interchangeable with respect to only one, rather than all, neighboring variable. Based on this observation, the domain of a variable is partitioned with respect to each constraint on it such that values in each partition are interchangeable with respect to a certain constraint. One immediate advantage of this method is that a filtering procedure (e.g., AC algorithms) could be implemented more efficiently by taking each partition as one value. A search procedure is also introduced to make use of the neighborhood interchangeability (but not CNI or CNS).

6.4 Inferred Disjunctive Constraints

The inferred disjunctive constraints [Freuder and Hubbe, 1993] make use of the complete substitutability of CNS and some other observations to decompose a CSP.

7 Conclusion

When two values of a variable are not (neighborhood) interchangeable or substitutable, there exists some "interchangeability" and "substitutability" among them under some condition. The condition is usually a restriction on the domain of each neighboring variable. we propose conditional (neighborhood) interchangeability and substitutability which could be used to prune search space. They further strengthen the pruning constraints and concepts proposed by Prestwich [Prestwich, 2004] and Chmeiss & Sais [Chmeiss and Sais, 2003]. They also offer a uniform perspective on the previous work (e.g., [Bowen and Likitvivatanavong, 2004; Freuder and Hubbe, 1993; Prestwich, 2004; Chmeiss and Sais, 2003]). Prestwich has studied IC's, based on SAT solver. We are planning experiments to study the efficiency to prune the search space by using conditional neighborhood substitutability in a CSP solver.

8 Acknowledgement

We thank Chavalit Likitvivatanavong, Barry O'Sullivan, Steve Prestwich, and Nic Wilson for useful discussions on the topics covered in this material.

References

[Bowen and Likitvivatanavong, 2004] James Bowen and Chavalit Likitvivatanavong. Domain transmutation in Constraint Satisfaction Problems. To appear in Proceedings of AAAI-04, 2004.

[Chmeiss and Sais, 2003] A. Chmeiss and L. Sais. About neighborhood substitutability in CSPs. In *Proceedings of SymCon-03*, pages 41–45, Kinsale, Ireland, 2003.

[Freuder and Hubbe, 1993] Eugene C. Freuder and Paul D. Hubbe. Using inferred disjunctive constraints to decompose constraint satisfaction problems. In *Proceedings of IJCAI-93*, pages 254–260, Chambery, France, 1993. AAAI press.

[Freuder and Sabin, 1997] Eugene C. Freuder and Daniel Sabin. Interchangeability supports abstraction and reformulation for multi-dimensional constraint satisfaction. In *Proceedings of AAAI/IAAI-97*, pages 191–196, Providence, Rhode Island, 1997. AAAI press.

[Freuder, 1991] E.C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *Proceedings of AAAI-91*, pages 227–233, Anaheim, California, 1991. AAAI press.

[Haselbock, 1993] A. Haselbock. Exploiting interchangeabilities in Constraint Satisfaction Problems. In *Proceedings of IJCAI-93*, pages 282–287, Chambery, France, 1993. IJCAI Inc.

[Prestwich, 2004] Steven Prestwich. Exploiting full interchangeability. Manuscript, 2004.

[Rainer Weigel, 1999] Boi Faltings Rainer Weigel. Compiling constraint satisfaction problems. *Artificial Intelligence*, 115(2):257–287, 1999.

[Weigel *et al.*, 1996] R. Weigel, B. V. Faltings, and B. Y. Choueiry. Context in Discrete Constraint Satisfaction Problems. In *Proceedings of ECAI-96*, pages 205–209, Budapest, Hungary, 1996. John Wiley & Sons, Ltd.