# Symmetry Definitions for Constraint Satisfaction Problems

David Cohen[1], Peter Jeavons[2], Christopher Jefferson[3], Karen E. Petrie[4], and Barbara M. Smith[4]

[1] Department of Computer Science, Royal Holloway, University of London, UK
D.Cohen@rhul.ac.uk,
[2] Computing Laboratory, University of Oxford, UK
peter.jeavons@comlab.ox.ac.uk,
[3] Department of Computer Science, University of York, UK
christopher.jefferson@cs.york.ac.uk,
[4] Cork Constraint Computation Centre, University College Cork, Ireland
{kpetrie,b.smith}@4c.ucc.ie

**Abstract.** We review the many different definitions of symmetry for constraint satisfaction problems (CSPs) that have appeared in the literature, and show that a symmetry can be defined in two fundamentally different ways: as an operation preserving the solutions of a CSP instance, or else as an operation preserving the constraints. We refer to these as *solution symmetries* and *constraint symmetries*. We define a constraint symmetry more precisely as an automorphism of a hypergraph associated with a CSP instance, the microstructure complement. We show that the solution symmetries of a CSP instance can also be obtained as the automorphisms of a related hypergraph, the *k-ary nogood hypergraph* and give examples to show that some instances have many more solution symmetries than constraint symmetries. Finally, we discuss the practical implications of these different notions of symmetry.

## 1 Introduction

The issue of *symmetry* is now widely recognised as of fundamental importance in constraint satisfaction problems (CSPs). It seems self-evident that in order to deal with symmetry we should first agree what we mean by symmetry. Surprisingly, this appears not to be true: researchers in this area have defined symmetry in fundamentally different ways, whilst often still identifying the same collection of symmetries in a given problem and dealing with them in the same way.

In this paper, we first survey the various symmetry definitions that have appeared in the literature. We show that the existing definitions reflect two distinct views of symmetry: that symmetry is a property of the solutions, i.e. that any mapping that preserves the solutions is a symmetry; or that symmetry preserves the constraints, and therefore as a consequence also preserves the solutions. We propose two new definitions of *solution symmetry* and *constraint symmetry* to capture these two distinct views, and show that they are indeed different: although any constraint symmetry is also a solution symmetry, there can be many solution symmetries that are not constraint symmetries. We discuss the relationship between the symmetry groups identified by these definitions and show that each is the automorphism group of a hypergraph, derived from either the solutions or the constraints of the CSP. We illustrate these ideas by discussing how they apply to a well-studied example problem, the $n$-queens problem. Finally, we discuss how these definitions of symmetry may be used in practice.

## 2 A Brief Survey of Symmetry Definitions

There have been many papers in recent years on symmetry in constraint satisfaction and related problems, not all of which give a clear definition of symmetry. In this section, we review the variety of definitions that have been used.

We first fix our terminology by defining a CSP instance as follows.

**Definition 1.** *A CSP instance is a triple $\langle V, D, C\rangle$ where:*

- *$V$ is a set of* variables*;*
- *$D$ is a universal* domain*, specifying the possible values for those variables;*
- *$C$ is a set of* constraints*. Each constraint $c \in C$ is a pair $c = \langle \sigma, \rho\rangle$ where $\sigma$ is a list of variables from $V$, called the constraint* scope*, and $\rho$ is a $|\sigma|$-ary relation over $D$, called the constraint* relation*.*

*An* assignment *of values to variables is a set $\{\langle v_1, a_1\rangle, \langle v_2, a_2\rangle, ..., \langle v_k, a_k\rangle\}$ where $\{v_1, v_2, ..., v_k\} \subseteq V$ and $a_i \in D$, for all $i$ such that $1 \leq i \leq k$. Note that the constraint relation of a constraint $c$ is intended to specify the assignments that are allowed by that constraint.*

*A* solution *to the CSP instance $\langle V, D, C\rangle$ is a mapping from $V$ into $D$ whose restriction to each constraint scope is a member of the corresponding constraint relation, i.e., is allowed by the constraint.*

We will call a CSP $k$-ary if the maximum arity of any of its constraints is $k$.

There are two basic types of definition for symmetry in a CSP instance: those that define symmetry as a property of the set of solutions, and those that define symmetry as a property that can be identified in the statement of the problem, without solving it. We shall refer to these informally in this section as *solution symmetry* and *problem symmetry* or *constraint symmetry*. In Section 3 we will define them formally and use these definitions to show how the two types of symmetry are related.

An example of an early definition of *solution symmetry* is given by Brown, Finkelstein & Purdom [5], who define a symmetry as a permutation of the problem variables that leaves invariant the set of solutions. Backofen and Will [2] similarly define a symmetry as a bijective function on the set of solutions of a CSP: they allow a symmetry to be specified by its effect on the individual assignments of values to variables.

A number of papers have defined *problem symmetry* in propositional calculus. Aguirre [1] and Crawford, Ginsberg, Luks & Roy [6] each define symmetry similarly: if $S$ is a set of clauses in CNF, then a permutation $\pi$ of the variables in those clauses is a symmetry of $S$ if $\pi(S) = S$. The expression $\pi(S)$ denotes the result of applying the permutation $\pi$ to the clauses in $S$. If this permutation simply re-orders the literals in individual clauses, and reorders the clauses, then it leaves $S$ effectively unchanged, and so in this case $\pi(S) = S$ and $\pi$ is a symmetry. Benhamou and Sais [4] use a slightly more general definition, in which a symmetry is a permutation defined on the set of *literals* that preserves the set of clauses. For example, given two variables $x$ and $y$, $x$ may be mapped to $\neg y$.

In CSPs, some authors have similarly defined a symmetry as a mapping that leaves the constraints unchanged, but have often restricted the allowed mappings to those that affect only the variables or only the values. Note that a constraint may be specified *extensionally* by listing its allowed tuples, or *intensionally* by giving an expression such as $x < y$ from which the allowed tuples could be determined. Permuting the variables in a constraint will in general change it: for example, the constraint $x + y = z$ is not the same as the constraint $x + z = y$. Puget [15] defines the notion of a *symmetrical constraint*, that is, a constraint which is unaffected by the order of the variables. For example, the binary inequality constraint, $\neq$, is symmetrical. He defines a symmetry of a CSP as a permutation of the variables which maps the set of constraints into an equivalent set: any constraint is either unchanged by the permutation or is an instance of a symmetrical constraint and is mapped onto a constraint on the same set of variables.

A similar idea was introduced by Roy and Pachet [17]. They define the notion of *intensional permutability*: two variables are intensionally permutable if: they have the same domain; any constraint affecting either of them affects both; for any constraint affecting these two variables, interchanging them in the expression defining the constraint does not change it. (The constraint is assumed to be defined intensionally, hence the name.) For example, in a linear constraint, any two variables with the same coefficient are intensionally permutable (with respect to that constraint, and assuming that they have the same domain).

Both Puget [15] and Roy and Pachet [17] restrict their definitions of symmetries to mappings that permute the *variables* of the problem only. Meseguer and Torras [14] define symmetries that act on both the variables and the values of a CSP. They define a symmetry on a CSP with $n$ variables as a collection of $n+1$ bijective mappings $\Theta = \{\theta, \theta_1, ..., \theta_n\}$. The mapping $\theta$ is a bijection on the set of variables $\{x_1, x_2, \ldots, x_n\}$; each $\theta_i$ is a bijection from $D(x_i)$ to $D(\theta(x_i))$ (where $D(x_i)$ is the domain $D$ restricted to the acceptable values for $x_i$ by unary constraints). These mappings will also transform each constraint. The set $\Theta$ is called a symmetry if it does not change the set of constraints $C$, as a whole.

Meseguer and Torras's definition allows both *variable* symmetries (that permute only the variables) and *value* symmetries (that permute only the values) as special cases, and hence is more general than many earlier definitions. However, it does not allow mappings in which variable-value pairs involving the same variable (say $\langle x_i, a_1\rangle$ and $\langle x_i, a_2\rangle$) can be mapped to variable-value pairs involving different variables (say $\langle x_j, a_j\rangle$ and $\langle x_k, a_k\rangle$, where $x_j \neq x_k$). For example, Meseguer and Torras consider the $n$-queens problem, in the commonly-used CSP formulation in which the variables correspond to the rows of the chessboard and the values to the columns. They show that the reflections in the horizontal and vertical axes and the rotation of the chessboard through 180° are symmetries of the corresponding CSP according to their definition, but the other four chessboard symmetries (reflection in the diagonals, rotation through 90° and 270°) are not. This example will be considered in more detail in Section 4 below.

Finally, we consider the notion of *interchangeability*, as defined by Freuder [9]. This is a form of solution symmetry: two values $a, b$ for a variable $v$ are *fully interchangeable* if every solution to the CSP containing the assignment $\langle v, a\rangle$ remains a solution when $b$ is substituted for $a$, and vice versa. As Freuder notes, in general identifying fully interchangeable values requires finding all solutions to the CSP. He therefore defines local forms of interchangeability that can be identified by inspecting the problem. Neighbourhood interchangeability, for example, is a form of constraint symmetry: two values $a, b$ for a variable $v$ are said to be neighbourhood interchangeable if for every constraint $c$ whose scope includes $v$, the set of assignments that satisfy $c$ which contain the pair $\langle v, a\rangle$ still satisfy $c$ when this is replaced by $\langle v, b\rangle$, and vice versa.

Benhamou [3] extends the ideas of interchangeability slightly and distinguishes between *semantic* and *syntactic* symmetry in CSPs, corresponding to our notions of solution symmetry and constraint symmetry, respectively. He defines two kinds of semantic symmetry: two values $a_i$ and $b_i$ for a CSP variable $v_i$ are *symmetric for satisfiability* if the following property holds: there is a solution which assigns the value $a_i$ to $v_i$ if and only if there is a solution which assigns the value $b_i$ to $v_i$. The values are *symmetric for all solutions* if: each solution containing the value $a_i$ can be mapped to a solution containing the value $b_i$. (The latter property implies the former.) Identifying semantic symmetries requires solving the CSP to find all solutions, and examining them. The notion of *syntactic symmetry* in [3] is defined as follows. Let $P = \langle V, D, C\rangle$ be a binary CSP instance, whose constraint relations are all members of some set $R$. A permutation $\pi$ of $D$ is a syntactic symmetry if $\forall r_{ij} \in R$, we have $(d_i, d_j) \in r_{ij} \implies (\pi(d_i), \pi(d_j)) \in r_{ij}$. In other words, the permutation $\pi$ does not change any constraint relation of $P$, considered as a set of tuples.

From this brief survey of existing symmetry definitions, it can be seen that they differ both in what aspect of the CSP they act on (only the values, only the variables, or variable-value pairs) and in what they preserve (the constraints or the set of solutions). It should be noted that it has become standard in the symmetry breaking methods that act during search (e.g. [2, 7, 8, 11]), as opposed to adding constraints to the CSP, to describe symmetries by their action on variable-value pairs. Hence, almost all the definitions described in this section are more restrictive than these systems allow.

Under all the definitions, symmetries map solutions to solutions and non-solutions to non-solutions; the definitions disagree over whether this is a defining property, so that any bijective mapping of the right kind that preserves the solutions must be a symmetry, or whether it is simply a consequence of leaving the constraints unchanged. In the next section we will show that this distinction is critical: the choice we make can seriously affect the symmetries that we find.

## 3 Constraint Symmetries and Solution Symmetries

In this section we will introduce two definitions of symmetries for constraint satisfaction problems that are sufficiently general to encompass all the types of symmetry allowed by the definitions given in the last section.

Note that the essential feature that allows any bijective mapping on a set of objects to be called a symmetry is that it leaves some property of those objects unchanged. It follows from this that the identity mapping will always be a symmetry, and the inverse of any symmetry will also be a symmetry. Furthermore, given two symmetries we can combine them (by composing the mappings) to obtain another symmetry, and this combination operation is associative. Hence, the set of symmetries forms a *group*.

The particular group of symmetries that we obtain depends on exactly what property it is that we choose to be preserved. Our first definition uses the property of being a solution, and is equivalent to the definition used in [2].

**Definition 2.** *For any CSP instance $P = \langle V, D, C\rangle$, a* solution symmetry *of $P$ is a permutation of the set $V \times D$ that preserves the set of solutions to $P$.*

In other words, a solution symmetry is a bijective mapping defined on the set of possible variable-value pairs of a CSP, that maps solutions to solutions. Note that this general definition allows variable and value symmetries as special cases.

To state our definition of *constraint symmetries* we first describe a mathematical structure associated with any CSP instance. For a binary CSP instance, the details of the constraints can be captured in a graph, the *microstructure* [9, 12] of the instance.

**Definition 3.** *For any binary CSP instance $P = \langle V, D, C\rangle$, the* microstructure *of $P$ is a graph with set of vertices $V \times D$ where each edge corresponds either to an assignment allowed by a specific constraint, or to an assignment allowed because there is no constraint between the associated variables.*

For our purposes, it is more convenient to deal with the complement of this graph. The *microstructure complement* has the same set of vertices as the microstructure, but with edges joining all pairs of vertices which are *disallowed* by some constraint, or else are incompatible assignments for the same variable. In other words, two vertices $\langle v_1, a_1\rangle$ and $\langle v_2, a_2\rangle$ in the microstructure complement are connected by an edge if and only if:

– the vertices $v_1$ and $v_2$ are in the scope of some constraint, but the assignment of $a_1$ to $v_1$ and $a_2$ to $v_2$ is disallowed by that constraint; *or*
– $v_1 = v_2$ and $a_1 \neq a_2$.

Recall that any set of vertices of a graph which does not contain an edge is called an *independent set*. An immediate consequence of the definition of the microstructure complement is that a solution to a CSP instance $P$ is precisely an independent set of size $|V|$ in its microstructure complement.

The definition extends naturally to the non-binary case. Here the microstructure complement is a *hypergraph* whose set of vertices is again the set of all variable-value pairs. In this case, a set of vertices $E$ is a hyperedge of the microstructure complement if it represents an assignment *disallowed* by a constraint, or else consists of a pair of incompatible assignments for the same variable. In other words, a set of vertices $\{\langle v_1, a_1\rangle, \langle v_2, a_2\rangle, \ldots, \langle v_k, a_k\rangle\}$ is a hyperedge if and only if:

– $\{v_1, v_2, \ldots, v_k\}$ is the set of variables in the scope of some constraint, but the constraint disallows the assignment $\{\langle v_1, a_1\rangle, \langle v_2, a_2\rangle, \ldots, \langle v_k, a_k\rangle\}$; *or*
– $k = 2$, $v_1 = v_2$ and $a_1 \neq a_2$.

*Example 1.* The system of linear equations $x + y + z = 0$; $w + y = 1$; $w + z = 0$ over the integers modulo 2 (that is, where $1 + 1 = 0$) can be modelled as a CSP instance $P = \langle V, D, C\rangle$, with $V = \{w, x, y, z\}$, $D = \{0, 1\}$ and $C = \{c_1, c_2, c_3\}$, where $c_1, c_2, c_3$ correspond to the three equations.

The microstructure complement of $P$ is shown in Figure 1. It has eight vertices: $\langle w, 0\rangle$, $\langle w, 1\rangle$, $\langle x, 0\rangle$, $\langle x, 1\rangle$, $\langle y, 0\rangle$, $\langle y, 1\rangle$, $\langle z, 0\rangle$, $\langle z, 1\rangle$, and twelve hyperedges. The equation $x + y + z = 0$ disallows the assignment $\{\langle x, 0\rangle, \langle y, 0\rangle, \langle z, 1\rangle\}$ and three other assignments. Hence, the microstructure complement has four ternary hyperedges arising from this constraint, including $\{\langle x, 0\rangle, \langle y, 0\rangle, \langle z, 1\rangle\}$. Each binary constraint also gives two binary hyperedges. Finally, there are four binary hyperedges (one per variable) corresponding to pairs of different values for the same variable; for example, the hyperedge $\{\langle y, 0\rangle, \langle y, 1\rangle\}$.

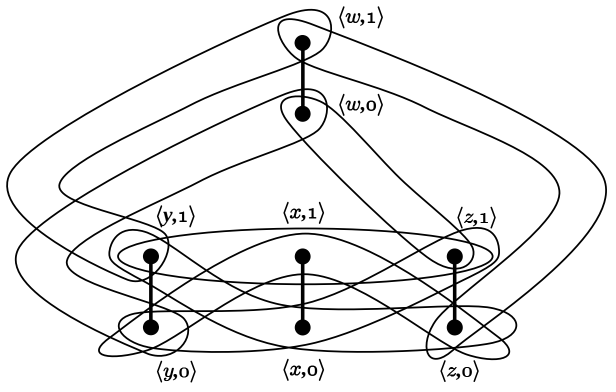

**Fig. 1.** The microstructure complement of the CSP instance $P$ defined in Example 1

We are now in a position to define a constraint symmetry. Recall that an *automorphism* of a graph or hypergraph is a bijective mapping of the vertices that preserves the edges (and hence also preserves the non-edges).

**Definition 4.** *For any CSP instance $P = \langle V, D, C \rangle$, a* constraint symmetry *is an automorphism of the microstructure complement of $P$ (or, equivalently, of the microstructure).*

The microstructure complement is related to the direct encoding of a CSP as a SAT instance [18]. The direct encoding has a variable for each variable-value pair in the original CSP; a clause for each pair of values for each variable, forbidding both values being assigned at the same time; and a clause for each tuple of variable-value pairs not allowed by a constraint (as well as other clauses ensuring that a value is chosen for every variable). A constraint symmetry as defined here is therefore equivalent to a permutation of the variables in the SAT encoding that does not change the set of clauses, and so is related to the definition of symmetry in SAT given by Crawford *et al.* [6].

*Example 2.* We consider the constraint symmetries of the CSP defined in Example 1, whose microstructure complement is shown in Figure 1. The automorphisms of this graph are the identity permutation together with the following permutations:

- $(\langle w,0\rangle\langle w,1\rangle)\ (\langle y,0\rangle\langle y,1\rangle)\ (\langle z,0\rangle\langle z,1\rangle)$;
- $(\langle w,0\rangle\langle w,1\rangle)\ (\langle y,0\rangle\langle z,0\rangle)\ (\langle y,1\rangle\langle z,1\rangle)$;
- $(\langle y,0\rangle\langle z,1\rangle)\ (\langle y,1\rangle\langle z,0\rangle)$;

(These permutations of the vertices are written in cycle form: for example, the first swaps the vertices $\langle w,0\rangle$ and $\langle w,1\rangle$ while simultaneously swapping $\langle y,0\rangle$ and $\langle y,1\rangle$ and swapping $\langle z,0\rangle$ and $\langle z,1\rangle$, but leaves $\langle x,0\rangle$ and $\langle x,1\rangle$ unchanged.) Hence, these four mappings are the constraint symmetry group of this CSP.

This example also shows that there can be more solution symmetries than constraint symmetries. The CSP has only two solutions: $\{\langle w,0\rangle, \langle x,1\rangle, \langle y,1\rangle, \langle z,0\rangle\}$ and $\{\langle w,1\rangle, \langle x,1\rangle, \langle y,0\rangle, \langle z,1\rangle\}$. The permutation $(\langle w,0\rangle\langle z,0\rangle\langle y,1\rangle)$, which maps $\langle w,0\rangle$ to $\langle z,0\rangle$, $\langle z,0\rangle$ to $\langle y,1\rangle$, $\langle y,1\rangle$ to $\langle w,0\rangle$ and leaves all other variable-value pairs unchanged, is a solution symmetry. This mapping preserves both solutions, but clearly is not a constraint symmetry.

Although Definition 2 and Definition 4 appear to be very different, we now show that there are some simple relationships between solution symmetries and constraint symmetries.

**Theorem 1.** *The group of constraint symmetries of a CSP instance $P$ is a subgroup of the group of solution symmetries of $P$.*

*Proof.* Let $P$ be a CSP instance and let $\pi$ be any automorphism of the microstructure complement of $P$. We will show that $\pi$ maps solutions to solutions, and hence is a solution symmetry of $P$.

Let $s$ be any solution of $P$, and let $W$ be the corresponding set of vertices in the microstructure complement of $P$. By the construction of the microstructure complement, $W$ is an independent set of size $|V|$. Since $\pi$ is an automorphism, we know that $\pi(W)$ is also an independent set of size $|V|$, and so is a solution. $\square$

Next we show that the group of all solution symmetries of an instance $P$ is also the automorphism group of a certain hypergraph. We first define a *nogood*.

**Definition 5.** *For any CSP instance $P$, a $k$-ary* nogood *is an assignment to $k$ variables of $P$ that cannot be extended to a solution of $P$.*

*The $k$-*nogood hypergraph *of $P$ is a hypergraph whose set of vertices is $V \times D$ and whose set of edges is the set of all $m$-ary nogoods for all $m \leq k$.*

The $k$-nogood hypergraph of a CSP instance has the same vertices as the microstructure complement. For a $k$-ary CSP (one whose constraints have maximum arity $k$), the $k$-ary nogood hypergraph contains every hyperedge of the microstructure complement, and possibly some others. The additional hyperedges represent partial assignments of up to $k$ variables that are allowed by the constraints, but do not appear in any solution because they cannot be extended to a full assignment satisfying all the constraints.

*Example 3.* Consider again the CSP instance $P$ defined in Example 1, with solutions, $\{\langle w,0\rangle, \langle x,1\rangle, \langle y,1\rangle, \langle z,0\rangle\}$ and $\{\langle w,1\rangle, \langle x,1\rangle, \langle y,0\rangle, \langle z,1\rangle\}$.

This instance has a large number of 3-ary nogoods, and the 3-nogood hypergraph of $P$ has a large number of hyperedges, in addition to those in the microstructure complement. These include the hyperedge $\{\langle x,0\rangle, \langle y,0\rangle, \langle z,0\rangle\}$, for example. This assignment is allowed by the 3-ary constraint on the variables $x, y, z$, but cannot be extended to a complete solution of $P$. Many of the additional hyperedges do not correspond to the scope of any constraint: for example, the hyperedge $\{\langle w,0\rangle, \langle x,1\rangle, \langle y,0\rangle\}$.

**Theorem 2.** *For any $k$-ary CSP instance $P$, the group of all solution symmetries of $P$ is equal to the automorphism group of the $k$-nogood hypergraph of $P$.*

*Proof.* Let $F$ be the $k$-nogood hypergraph of $P$ and let $\pi$ be any automorphism of $F$. We will show that $\pi$ preserves solutions, and hence is a solution symmetry.

Let $s$ be any solution of $P$, and let $W$ be the corresponding set of vertices in $F$. By the construction of this hypergraph, $W$ is an independent set of size $|V|$. Since $\pi$ is an automorphism of $F$, we know that $\pi(W)$ is also an independent set of size $|V|$. Hence $\pi(W)$ is not disallowed by any of the constraints of $P$, and is a solution.

Conversely, let $\pi$ be a solution symmetry of $P$. We will show that $\pi$ maps every set of $k$ or fewer vertices of $F$ which is not a hyperedge to another non-hyperedge, and hence $\pi$ is an automorphism of this hypergraph.

Let $E$ be any set of $k$ or fewer vertices in $F$ which is not a hyperedge. Since every nogood of $P$ of size $k$ or less is a hyperedge of the $k$-nogood hypergraph, it follows that $E$ can be extended to at least one solution of $P$.

Hence we may suppose that $E$ is part of some solution $s$. Now, $s$ is mapped to the solution $\pi(s)$ by the solution symmetry $\pi$. Every $k$-ary projection of this solution, including the image $\pi(E)$ of $E$, is a non-hyperedge in $F$, and so we are done. □

Theorem 2 shows that to obtain the solution symmetries of a CSP instance it is sufficient to consider the automorphisms of the hypergraph obtained by adding all the nogoods of arity $k$ or less to the microstructure complement. We will show in the next section that in some cases there are hypergraphs obtained by adding a smaller number of edges to the microstructure complement which already have all solution symmetries as automorphisms. However, the next result shows that there are cases where it is in fact necessary to add all nogoods of arity $k$ or less to the microstructure complement in order to obtain a hypergraph with all solution symmetries as automorphisms.

**Proposition 1.** *For some $k$-ary CSP instances $P$, the $k$-nogood hypergraph is the only hypergraph containing the microstructure complement of $P$ whose automorphisms are exactly the solution symmetries.*

*Proof.* Consider a CSP instance $P$, with constraints of every arity less than or equal to $k$, which has no solutions. Let $H$ be the microstructure complement of $P$.

Since $P$ has no solutions, every permutation of the vertices of $H$ is a solution symmetry. For each positive integer $m \leq k$, there is at least one $m$-tuple disallowed by a constraint, so $H$ has at least one $m$-ary hyperedge. Since every permutation of the vertices of $H$ is a solution symmetry, applying the solution symmetry group to $H$ will give all $m$-sets of vertices as hyperedges. Hence the only hypergraph containing $H$ whose automorphisms include all solution symmetries is the hypergraph with all $m$-sets of vertices as edges, for all $m \leq k$, which is equal to the $k$-nogood hypergraph. $\square$

Hence to obtain all solution symmetries of a $k$-ary CSP instance it is sometimes necessary to consider all $m$-ary nogoods, for all $m \leq k$. On the other hand, Theorem 2 shows that we do not need to consider nogoods of any size *larger* than $k$. (In fact the same proof shows that adding all nogoods of size $l$, for any $l$ larger than $k$, to the $k$-nogood hypergraph does not change its automorphism group.) In particular, this means that to obtain all the solution symmetries of a binary CSP instance we need only consider the binary and unary nogoods.

Theorem 1 and Theorem 2 help to clarify the relationship between solution symmetries and constraint symmetries. One reason that it is important to distinguish these two kinds of symmetries carefully is that, in general, there can be many more solution symmetries than constraint symmetries for a given CSP instance, as we will show in the next section.

## 4 Case Study: Symmetry in $n$-Queens

In this section we will illustrate the relationship between solution symmetries and constraint symmetries by examining the $n$-queens problem. This problem is useful for discussing symmetry because the common CSP formulation has several different types of symmetry, some of which are beyond the scope of some earlier definitions in the literature.

The standard formulation of the $n$-queens problem as a CSP has $n$ variables corresponding to the rows of the chessboard, say $r_1, r_2, ..., r_n$. The domain of values corresponds to the columns of the chessboard, say $D = \{1, 2, ..., n\}$. The constraints can be expressed as follows:

- the values of $r_1, r_2, ..., r_n$ are all different;
- for all $i, j, 1 \leq i < j \leq n$, $|r_i - r_j| \neq |i - j|$.

A chessboard has eight geometric symmetries: reflections in the horizontal and vertical axes and the two diagonals, rotations through 90°, 180° and 270°, and the identity.

Recall, however, that Meseguer & Torras [14] did not allow the full set of geometric symmetries of the chessboard as symmetries of the usual CSP formulation of $n$-queens. This is because the formulation introduces an asymmetry between rows and columns, so that some of the geometric symmetries do not leave the constraints syntactically unchanged. In particular, the rotational symmetries through 90° and 270° map assignments forbidden by some of the constraints to assignments that are mutually incompatible because they assign two values to the same variable. For example, the forbidden pair consisting of $\langle r_1, 1\rangle$ and $\langle r_2, 1\rangle$ is mapped by the rotation through 90° to the incompatible pair consisting of $\langle r_1, n\rangle$ and $\langle r_1, n-1\rangle$.

The microstructure complement restores the symmetry between rows and columns, by treating in the same way both of these reasons for a pair of assignments to be disallowed. Hence each geometric symmetry of the chessboard gives rise to a constraint symmetry of the $n$-queens problem for any $n$, according to our definition of constraint symmetry (Definition 4).

Clearly, the set of *solutions* to an instance of the $n$-queens problem is invariant under each of the eight geometric symmetries of the chessboard. Hence each of these geometric symmetries is a solution symmetry of the $n$-queens problem for any $n$, according to our definition of solution symmetry (Definition 2). However, there can be many other solution symmetries for instances of this problem, as we will now show.

The 3-queens problem has no solutions; like any other CSP with no solution, any permutation of the possible variable-value pairs is a solution symmetry. This is confirmed by Theorem 2: the binary nogood hypergraph is the complete graph with nine vertices, and any permutation of the vertices is an automorphism.

The 4-queens problem has two solutions, shown in Figure 2. (These two solutions are each mapped to the other by a reflection of the chessboard.) In this case, it is easier to consider the *complement* of the binary nogood hypergraph, in which each edge represents a pair of variable-value assignments that is allowed by the solutions. Figure 2 also shows this graph, drawn so that each vertex, representing a variable-value pair, and hence a square of the chessboard, corresponds to the position on a chessboard of that square. Each solution is represented as a 4-clique in this graph, rather than as an independent set of size 4 in the binary nogood hypergraph. The automorphisms of this graph are that: the vertices within either clique can be permuted; the vertices in one clique can be swapped with those in the other; and the eight isolated vertices (representing unary nogoods) can be permuted; and we can also compose these permutations. This gives a total of $4! \times 4! \times 2 \times 8!$ automorphisms, or $46,448,640$. Since the automorphisms of a graph are the same as the automorphisms of its complement, these $46,448,640$ automorphisms are the solution symmetries of 4-queens, by Theorem 2.

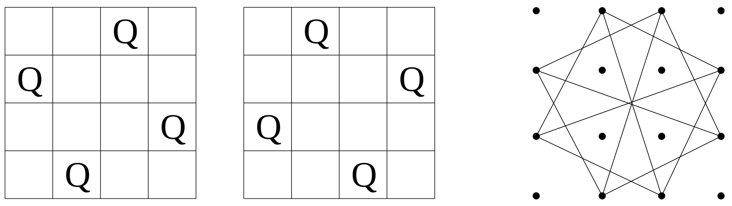

**Fig. 2.** The solutions of the 4-queens problem (left) and the complement of the binary nogood graph (right).

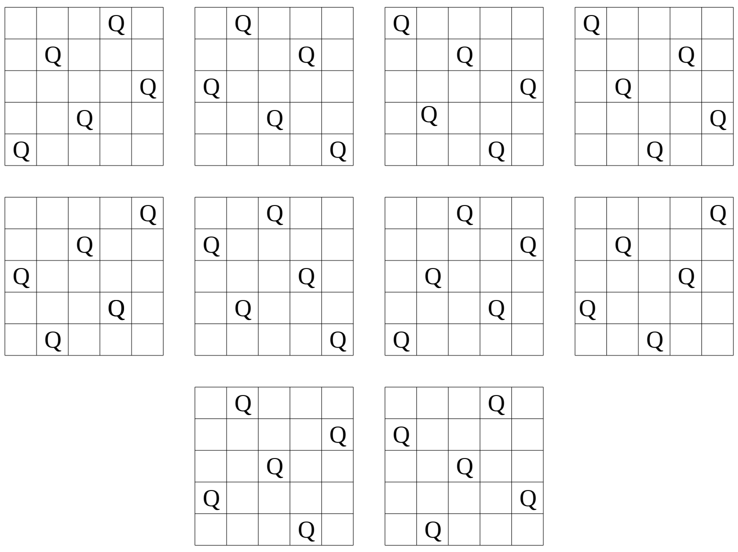

**Fig. 3.** The ten solutions of the 5-queens problem

The 5-queens problem has ten solutions, shown in Figure 3. These solutions are divided into two equivalence classes by the geometric symmetries of the chessboard; they transform any solution into another solution from the same equivalence class. (The first eight solutions shown form one class, the last two the second class.)

Every square of the chessboard has a queen on it in at least one of these ten solutions, so that there are no unary nogoods. However, there are some *pairs* of squares, where two queens can be placed consistently with the original constraints, but which are not allowed in any solution. The 40 additional binary nogoods can in fact be derived by using path consistency: for example, a pair of queens in row 1, column 1 and row 3, column 4 together attack every square in row 2, and so this pair of assignments cannot be part of any solution.

If these additional binary nogoods are added to the microstructure complement, then we obtain the 2-ary nogood graph for the 5-queens problem. By using the software tool NAUTY [13], we can find the automorphism group of this graph: it has 28,800 elements, so the 5-queens problem has a total of 28,800 solution symmetries, by Theorem 2. (We have also used NAUTY to confirm that the microstructure complement has just the eight geometric symmetries.)

In this case, unlike the 4-queens problem, it is difficult to develop an intuitive understanding of the additional solution symmetries. Some are easy to see: for instance, the rows of the board can be cyclically permuted, as can the columns. The subgroup consisting of these permutations together with the geometric symmetries and all combinations makes all 10 solutions symmetrically equivalent, i.e. there is just one equivalence class, rather than two. However, this subgroup is still much smaller than the full solution symmetry group.[5]

In the 6-queens problem, additional binary nogoods can again be derived by path consistency. For example, queens in row 1, column 2 and row 3, column 5 together attack all the squares in row 2. There are also unary nogoods, since the problem has only four solutions. Adding all these nogoods to the microstructure complement yields a very large symmetry group, as shown in Table 1.

For $n \geq 7$, path consistency does not give any new nogoods, since two queens can together attack at most six squares on any row. Even so, there are binary nogoods in addition to those in the original constraints for $n = 7, 8, 9$; beyond that, we have tested up to $n = 16$ and found no further additional binary nogoods.

Table 1 shows that for $n \geq 7$, the solution symmetries appear to be just the geometric symmetries: in spite of the additional binary nogoods for $n = 7, 8, 9$, the binary nogood graph for these instances has the same automorphism group as the microstructure complement. This demonstrates that in spite of Proposition 1, the minimal hypergraph containing the microstructure complement whose automorphisms are the solution symmetries can sometimes be smaller than the $k$-ary nogood hypergraph. Note that the number of solutions to the $n$-queens problem increases rapidly with $n$: intuitively, it becomes more difficult for a solution symmetry to preserve them all, so that eventually the solution symmetries are just the constraint symmetries.

A principal reason for identifying symmetry in CSPs is to reduce search effort by not exploring assignments that are symmetrically equivalent to assignments considered elsewhere in the search. Clearly, if the solution symmetry group is larger than the constraint symmetry group, there will potentially be a greater search reduction from using the solution symmetries, if they can somehow be

[5] In an attempt to understand the solution symmetries of 5-queens in terms of simple transformations of the chessboard, we used NAUTY to find a small number of generators of the group, including one or more of the geometric symmetries. We found that the entire group can be generated by just two permutations of the variable-value pairs, together with the rotation through $90^\circ$. However, these two permutations have no obvious geometric interpretation in terms of the chessboard.

**Table 1.** The number of additional binary nogoods derived from the sets of solutions to the $n$-queens problem, and the number of solution symmetries

| $n$ | Additional binary nogoods | Solution symmetries |
|---|---|---|
| 3 | 8 | $9! = 362{,}880$ |
| 4 | 32 | $4! \times 4! \times 2 \times 8! = 46{,}448{,}640$ |
| 5 | 40 | 28,800 |
| 6 | 280 | 3,089,428,805,320,704,000,000 |
| 7 | 72 | 8 |
| 8 | 236 | 8 |
| 9 | 40 | 8 |
| 10 | 0 | 8 |

identified in advance. In some cases, as in the 5-queens problem, establishing some level of consistency in the problem to find new nogoods (of arity $\leq k$), and adding these to the microstructure complement, will give an automorphism group that is nearer to the solution symmetry group, if not equal to it.

When finding all solutions, the aim in symmetry breaking is to find just one solution from each symmetry equivalence class; in the 5-queens problem, the solutions fall into two equivalence classes when using the constraint symmetries and only one when using the solution symmetries. Hence, if the aim is to find a set of nonisomorphic solutions, the appropriate symmetry group should be chosen in advance, since the choice can affect the number of solutions found.

This raises the question of how to identify the symmetries of a CSP, either the constraint symmetries or the solution symmetries; we discuss this next.

## 5 Identifying Symmetry in Practice

Symmetry in CSPs is usually identified, in practice, by applying human insight: the programmer sees that some transformation would transform a hypothetical solution into another hypothetical solution. The definition of constraint symmetry given earlier can be used to confirm that candidate transformations are genuine symmetries. It is not necessary to generate the entire microstructure complement for this purpose, but only to demonstrate that each candidate mapping will map edges to edges and non-edges to non-edges in this hypergraph.

Identifying symmetry in a CSP by inspection is prone to missing some of the symmetry. Using Definition 4 we can, in principle, be sure to identify all the constraint symmetries in a problem by generating the microstructure complement and finding its automorphism group. However, it will often be impracticable to generate the microstructure, especially for large CSPs with non-binary constraints. It may in that case be possible to represent the constraints more compactly while preserving the important details; for instance, Ramani and Markov [16] propose to represent constraints by parse trees and find the automorphisms of the resulting graph.

Many authors have defined symmetry in CSPs in a similar way to our definition of solution symmetry, but have effectively only identified constraint symmetries; we have shown that the solution symmetry group can be much larger than the constraint symmetry group. This suggests a novel, *incremental* approach to using symmetry during search, in which we maintain a set of currently known symmetries throughout the solution process. This set is initialised to the group of constraint symmetries. Each time a nogood of arity $k$ or less is found during preprocessing, or during the search for solutions, it is added to our current view of the $k$-nogood hypergraph, together with all of its images under currently known symmetries. Adding these edges might increase the number of automorphisms of this graph, and hence increase the set of currently known symmetries. The bigger this group of symmetries gets, the more information we get from each additional nogood.

Methods such as those proposed here may find a potentially very large group of symmetries, but with possibly only a small number of generators. For instance, as shown earlier, the solution symmetry group of 5-queens has 28,800 elements but just three generators. Symmetry-breaking methods that combine dynamic symmetry breaking during search with computational group theory, e.g. [10], can exploit such symmetry groups effectively.

## 6 Conclusion

We have reviewed definitions of symmetry in CSPs and have proposed definitions of *constraint symmetry* and *solution symmetry* to encompass two types of definition that have been used. We have shown that there can be many more solution symmetries, i.e. permutations of the variable-value pairs that preserve the solutions, than constraint symmetries, i.e. permutations that preserve the constraints. In practice, researchers have identified constraint symmetries in CSPs rather than solution symmetries, regardless of their definition of symmetry, because of the difficulty of identifying solution symmetries that are not also constraint symmetries without examining the set of solutions. However, we have shown that for a $k$-ary CSP, the solution symmetries are the automorphisms of the $k$-ary nogood hypergraph; hence, finding new nogoods of arity up to $k$ and adding them to the CSP can allow the constraint symmetry group to expand towards the solution symmetry group. Symmetry-breaking methods avoid exploring assignments that are symmetrically equivalent to assignments explored elswhere; hence, working with a larger symmetry group allows more assignments to be pruned and can further reduce the search effort to solve the problem.

## Acknowledgments

We thank the SBDS group, especially Warwick Harvey, Ian Gent and Steve Linton for helpful discussions on this topic; a review of symmetry definitions by Iain McDonald was also useful. We are grateful to Brendan McKay for his help in finding automorphism groups of graphs using NAUTY. We thank Marc van Dongen and Nic Wilson for helpful feedback. This material is based in part on works supported by the Science Foundation Ireland under Grant No. 00/PI.1/C075 and by the UK EPSRC under Grant No. GR/R29673; the authors were also supported by SymNet, the U.K. Symmetry and Search Network, funded by EPSRC.

## References

1. A. Aguirre. How to Use Symmetries in Boolean Constraint Solving. In F. Benhamou and A. Colmerauer, editors, *Constraint Logic Programming: Selected Research*, pages 287–306. MIT Press, 1992.
2. R. Backofen and S. Will. Excluding Symmetries in Constraint-Based Search. In J. Jaffar, editor, *Principles and Practice of Constraint Programming - CP'99*, LNCS 1713, pages 73–87. Springer, 1999.
3. B. Benhamou. Study of symmetry in constraint satisfaction problems. In *Proceedings of the 2nd Workshop on Principles and Practice of Constraint Programming, PPCP'94*, pages 246–254, May 1994.
4. B. Benhamou and L. Sais. Theoretical study of symmetries in propositional calculus and applications. In D. Kapur, editor, *Automated Deduction - CADE-11*, LNAI 607, pages 281–294. Springer-Verlag, 1992.
5. C. A. Brown, L. Finkelstein, and P. W. Purdom. Backtrack Searching in the Presence of Symmetry. In T. Mora, editor, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, LNCS 357, pages 99–110. Springer-Verlag, 1988.
6. J. Crawford, M. Ginsberg, E. Luks, and A. Roy. Symmetry-Breaking Predicates for Search Problems. In *Proceedings KR'96*, pages 149–159, Nov. 1996.
7. T. Fahle, S. Schamberger, and M. Sellmann. Symmetry Breaking. In T. Walsh, editor, *Principles and Practice of Constraint Programming - CP 2001*, LNCS 2239, pages 225–239. Springer, 2001.
8. F. Focacci and M. Milano. Global Cut Framework for Removing Symmetries. In T. Walsh, editor, *Principles and Practice of Constraint Programming - CP 2001*, LNCS 2239, pages 77–92. Springer, 2001.
9. E. C. Freuder. Eliminating Interchangeable Values in Constraint Satisfaction Problems. In *Proceedings AAAI'91*, volume 1, pages 227–233, 1991.
10. I. P. Gent, W. Harvey, T. Kelsey, and S. Linton. Generic SBDD using Computational Group Theory. In F. Rossi, editor, *Principles and Practice of Constraint Programming - CP 2003*, LNCS 2833, pages 333–347. Springer, 2003.
11. I. P. Gent and B. M. Smith. Symmetry Breaking During Search in Constraint Programming. In W. Horn, editor, *Proceedings ECAI'2000*, pages 599–603, 2000.
12. P. Jégou. Decomposition of Domains Based on the Micro-Structure of Finite Constraint-Satisfaction Problems. In *Proceedings AAAI'93*, pages 731–736, 1993.
13. B. McKay. Practical Graph Isomorphism. *Congressus Numerantium*, 30:45-87, 1981. (The software tool NAUTY is available for download from `http://cs.anu.edu.au/~bdm/nauty/`)
14. P. Meseguer and C. Torras. Exploiting symmetries within constraint satisfaction search. *Artificial Intelligence*, 129:133–163, 2001.
15. J.-F. Puget. On the Satisfiability of Symmetrical Constrained Satisfaction Problems. In J. Komorowski and Z. W. Ras, editors, *Proceedings of ISMIS'93*, LNAI 689, pages 350–361. Springer-Verlag, 1993.
16. A. Ramani and I. L. Markov. Automatically Exploiting Symmetries in Constraint Programming. In B. Faltings, A. Petcu, F. Fages, and F. Rossi, editors, *CSCLP 2004*, LNCS 3419, pages 98–112. Springer, 2005.
17. P. Roy and F. Pachet. Using Symmetry of Global Constraints to Speed up the Resolution of Constraint Satisfaction Problems. In *Workshop on Non Binary Constraints, ECAI-98*, Aug. 1998.
18. T. Walsh. SAT v CSP. In R. Dechter, editor, *Proceedings CP'2000*, LNCS 1894, pages 441–456. Springer, 2000.