Excluding Symmetries in Constraint-Based Search

Rolf Backofen and Sebastian Will

Institut für Informatik, LMU München
Oettingenstraße 67, D-80538 München

Abstract. We introduce a new method for excluding symmetries in constraint based search. To our knowledge, it is the first declarative method that can be applied to *arbitrary* symmetries. Our method is based on the notion of symmetric constraints, which are used in our modification of a general constraint based search algorithm. The method does not influence the search strategy. Furthermore, it can be used with either the full set of symmetries, or with an subset of all symmetries.
We proof correctness, completeness and symmetry exclusion properties of our method. We then show how to apply the method in the special case of geometric symmetries (rotations and reflections) and permutation symmetries. Furthermore, we give results from practical applications.

1 Introduction

In many search problems, one is faced with the existence of symmetries. Symmetries give rise to many different solutions found by the search procedure, which are all considered to be "similar". Often, one is not interested in also getting all the symmetric solutions to every found solution. Without exclusion of symmetries, whenever a solution is found or proven to be inconsistent with the search problem, the search algorithm still considers all symmetric solutions. Those symmetries will then give rise to an (often exponential) amplification of the search space. Hence, symmetry exclusion promises to efficiently prune the search tree.

E.g., consider constraint problems, where finite domain variables have a geometric interpretation such as in the N-queens problem or the square-tiling problem (where set of squares of given sizes must fit exactly into a fixed square). A more complex, real-world problem is the lattice protein structure prediction problem. In [2], it is shown how to find solutions for this problem using constraint programming techniques. In this case, different solutions are symmetric if they can be generated using reflections or rotations. In other problems, symmetric solutions can be generated by performing permutations on variables valuations (like in the map coloring (or graph coloring) problem).

In the following, we consider search problems that stems from constraint satisfaction problems (CSP). A common approach is to transform (problem specifically) a CSP C_1 into a CSP C_2 that excludes at least some of the symmetries of the original problem C_1.

Unfortunately, symmetry exclusion was often not straightforward to implement. Even then, it had to be redesigned for every special problem or enumeration strategy. This leaded to an inflexibility in the program structure once it was introduced. Thus, the widespread usage of symmetry exclusion was strongly hindered by its complexity. Often symmetry exclusion was not done at all or only for a small set of symmetries. In other cases, where symmetry exclusion was implemented it distracted the programmers attention from more important tasks.

In this paper, we will present a new approach for symmetry exclusion, which works by modifying the search algorithm. The technique of modifying the search algorithm to exclude symmetries was already used in the literature. In contrast to the known approaches, our approach is (to our knowledge) the first declarative method that can be applied to *arbitrary* symmetries that can be defined declaratively. Furthermore, it does not restrict the search strategy (i.e., the symmetry exclusion can be done independently of the search strategy). This is important since in many constraint problems, major part of the knowledge of the problem is encoded in the search strategy.

We have implemented our method in the concurrent constraint programming language Oz [10] using the programmable search engine described in [9]. But the method can be implemented in any system that handles implication (via entailment), and allows to modify the search procedure such that additional constraints are added in one branch of a branching step.

Related Work Previous work on symmetry exclusion by modifying the search algorithm handled only restricted forms of symmetries. In [7], the notion of interchangeable values is introduced. Two values a and b are interchangeable for a variable V iff for every solution that maps V to a, the corresponding valuation that maps V to b is also a solution.

The method in [3] handles only permutations of domain values in the case of binary constraints (thus, e.g., the all-distinct constraint has to be translated into binary inequalities, which implies that efficient propagation methods for all-distinct cannot be applied). Furthermore, it works by introducing a symmetry excluding form of constraint propagation, which is a modified form of domain propagation (thus, the method cannot make use of interval propagation).

[1] and [4] consider only propositional calculus. The symmetries there are permutations of Boolean variables. These methods do not apply to general constraint satisfaction problems.

The symmetry exclusion algorithm presented in [8] is essentially an implementation and application of our method to solve practical problems, which will be compared in greater detail in Section 3.

An example of a symmetry exclusion method that works by a (not problem specific) transformation of the constraint satisfaction problem is [5]. They introduce symmetry breaking predicates, which are true only for the smallest solution within an equivalence class (where the ordering is fixed in advance). Thus, if the search strategy encounters a non-minimal (according to the pre-fixed ordering) solution first, then it will be excluded by the symmetry breaking predicates. This implies that not every search strategy is possible given the pre-fixed ordering.

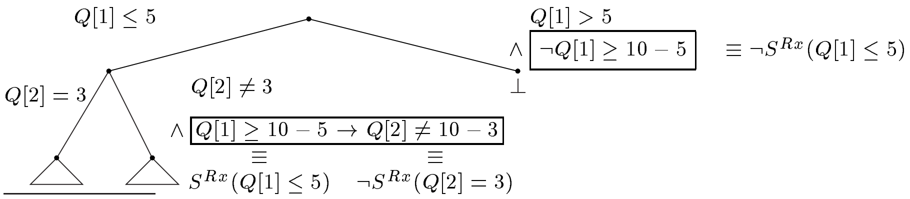
In contrast, there is no prefixed ordering in which symmetrical solutions will be excluded by our method. This implies that we respect used-defined search strategies as much as possible.[1]

Overview of the paper We start with an introductory example explaining the basic ideas of our method. After some preliminaries, we present the formal definition as wells as completeness and correctness proofs in Section 3. Additionally, we show that our method guarantees to exclude all specified symmetries. The proofs indicate that a subset of all symmetries can be enough to exclude the complete symmetry group. In Section 4, we treat geometric symmetries. We show how to define symmetric constraints and present the results of applying the problem to a complex problem. In Section 5, we consider symmetries which are generated by permuting the values of the variables. E.g., such symmetries occur in the graph coloring problem. We proof, that in this case, it is sufficient to exclude the subset of transposition in order to exclude all permutations. Furthermore, we present the results of applying the method to random instances of graph coloring.

Introductory Example We start with a simple example to explain the main concepts. Consider the N-queens problem, where we have an array $Q[1..N]$, whose elements can take values from $1..N$. $Q[i] = j$ states that there is a queen at position (i, j). Now consider the symmetry corresponding to the reflection at the axis $y = \frac{N}{2}$, which is parallel to the x-axis (in the sequel denoted by Rx). Clearly, for every solution of the N-queens, the the Rx-symmetric version is also a solution of N-queens. Hence, we want to exclude these symmetric solutions.

To use our method, we have first to introduce symmetric versions of the constraints which are inserted by the search procedure. One kind of constraint is $Q[i] = j$. If we have any distribution of queens on the board satisfying $Q[i] = j$, then the Rx-symmetric distribution will satisfy $Q[i] = N - j$. Hence, the constraint $Q[i] = N - j$ is the Rx-symmetric version of the constraint $Q[i] = j$. Similarly, we get that the Rx-symmetric constraint of $Q[i] \leq j$ is $Q[i] \geq N - j$. In the following, we write $S^{Rx}(c)$ to denote the Rx-symmetric constraint to c.

Our method works by adding additional constraints at the right branches of the search tree, which exclude the symmetric solutions that are encountered in the right branch. Consider the following search tree for the 10-queens problem, where we indicate the constraints added by symmetry exclusion via frames:



[1] Clearly, if the search strategy depends on the constraint store, then any method of symmetry exclusion that prunes the search tree must influence the way solutions are encountered (since the constraint store is changed by the symmetry exclusion).

This can be interpreted intuitively as follows. In the topmost branch, if we search for a solution in the right branch satisfying $Q[1] > 5$, then the Rx-symmetric solution will satisfy $Q[1] < 5$. Hence, the symmetric solution was already encountered earlier in the left branch (under $Q[1] \leq 5$), which is the reason that we can close the topmost right branch. For the second right branch labeled $Q[2] \neq 3$, we want again to exclude the Rx-symmetric solutions found in the right branch. Hence, we would like to add the constraint $\neg S^{Rx}(Q[2] = 3)$ to the right branch. But this would exclude too many solutions. The prerequisite of the Rx-symmetric solution to be found in the right branch is that both the solution and its Rx-symmetric version satisfies $Q[1] \leq 5$. Now the only solutions that satisfying this conditions are solutions that satisfy $Q[1] = 5$. Hence, we can add $\neg S^{Rx}(Q[2] = 3)$ *under the condition* that $Q[1] = 5$. But this is exactly the effect that we achieve by adding $(Q[1] \geq 10 - 5) \rightarrow (Q[2] \neq 10 - 3)$ at the second right branch (since the constraint store contains already $Q[1] \leq 5$).

2 Preliminaries

We fix a first-order signature Σ including the equality $\doteq$ with a set of variables $\mathcal{V}$. Constraints are literals, and constraint formulae are quantifier-free formulae over Σ. We identify $t \doteq t'$ with $t' \doteq t$. $\mathcal{C}$ denotes the set of all constraints. A set of constraints $C \subseteq \mathcal{C}$ is interpreted as the conjunction of the constraints contained in C, and we will freely mix set notation and conjunction. The set of free variables of C is denoted by $\mathcal{V}(C)$.

We fix a standard interpretation $\mathcal{A}$ with domain $\mathcal{D}^{\mathcal{A}}$, which describes our constraint theory. In the following, we assume a fixed constraint set C_{Pr} describing the problem to be solved. An *assignment* α *in* $\mathcal{A}$ is a partial function $\alpha : \mathcal{V} \rightarrow \mathcal{D}^{\mathcal{A}}$. We say that an (possible partial) assignment α *satisfies* ϕ (short $\alpha \models \phi$) if there is a total assignment $\alpha' \supseteq \alpha$ with $\mathcal{A}, \alpha' \models \phi$. We write $\phi \models \psi$ if for every α we have $\alpha \models \phi \rightarrow \psi$. A constraint set C *(syntactically) determines* a set of variables $\mathcal{X}$ to an assignment α iff for all $x \in \mathcal{X}$ exists a t such that $x \doteq t \in C$ and $\alpha(x) = \alpha(t)$.

In many problems, one is interested in only a subset of all variables, whose valuations uniquely determine the valuation of the other (auxiliary) variables. This is captured by the definition of solution variables.

Definition 1 (Solution Variables). *Let $\mathcal{X} \subseteq \mathcal{V}$ be a set of variables. $\mathcal{X}$ is a* solution variables set for C *if*

$$\forall\alpha\,[(\mathrm{dom}(\alpha) = \mathcal{X} \wedge \alpha \models C) \Rightarrow (\exists!\alpha'(\alpha \subseteq \alpha' \wedge \mathrm{dom}(\alpha') = \mathcal{V}(C) \wedge \alpha' \models C)]\,,$$

where $\exists!\alpha\phi(\alpha)$ ('exists a unique α satisfying $\phi(\alpha)$') is short for $\exists\alpha(\phi(\alpha) \wedge \forall\alpha' : \phi(\alpha') \Rightarrow \alpha = \alpha')$. For a solution variables set $\mathcal{X}$, we say that α is a $\mathcal{X}$-solution *for C if* $\mathrm{dom}(\alpha) = \mathcal{X}$ *and $\alpha \models C$. With $\|C\|^{\mathcal{X}}$ we denote the set of $\mathcal{X}$-solutions of C.*

In the following, we fix $\mathcal{X}$. Hence, we use the term 'solution' as short for '$\mathcal{X}$-solution for C', and we write $\|C\|$ as short for $\|C\|^{\mathcal{X}}$.

Definition 2 (Symmetry). *A* symmetry s for C_{Pr} *is a bijective function*

$$s : \|C_{\text{Pr}}\| \to \|C_{\text{Pr}}\|.$$

A symmetry set $\mathcal{S}$ for C_{Pr} *is a set of symmetries operating on* $\|C_{\text{Pr}}\|$. *A* symmetry group $\mathcal{S}$ *is a symmetry set for* C_{Pr} *which is a group.*

Note that for every symmetry s for C_{Pr}, also s^{-1} is a symmetry for C_{Pr}. We denote the identity function on $\|C_{\text{Pr}}\|$ with $\mathrm{id}_{C_{\text{Pr}}}$ (which is a symmetry by definition). Clearly, the set of all symmetries for C_{Pr} is a group. But many cases, we do not want to consider all symmetries, since either there are too many of them, or some of them do not have an intuitive characterization. In this case, we consider only a subset of all symmetries, which may not be a group.

3 Symmetry Excluding Search Trees

Now we have to define the notion of a search tree as we need it for the symmetry exclusion. We describe the effects of applying a constraint solver to a constraint set C by a function $P(C)$, which returns the closure of C under constraint propagation. Clearly, we have $P(C) \models\!\!| \, C$.

Usually, a search tree is a labeled tree, where the nodes are labeled with the constraints in the constraint store, and the edges are labled by constraints or negated constraints. A single node v is of the form

$$\begin{array}{ccc} & C_{\text{store}} & \\ c \swarrow & & \searrow \neg c \\ P(C_{\text{store}} \wedge c) & & P(C_{\text{store}} \wedge \neg c) \end{array} \quad .$$

The root node is labeled with $P(C_{\text{Pr}})$. Additionally, we need for the symmetry exclusion to store at a node v the positive and negative constraints that have been used along the path from the root to v to generate the node v.

Definition 3 (Search Tree). *Let t be a finite, binary, rooted and ordered tree, whose egdes are labeled by literals, and whose nodes are labeled by triples of constraint sets. The tree t is a* search tree for C_{Pr} *if the following conditions are satisfied:*

1. *The root node has the label* $(\emptyset, \emptyset, P(C_{\text{Pr}}))$,
2. *every binary node has the form*

$$\begin{array}{ccc} & (C_p, C_n, C_{\text{store}}) & \\ c \swarrow & & \searrow \neg c \\ (C_p \wedge c, C_n, C^l_{\text{store}}) & & (C_p, C_n \wedge \neg c, C^r_{\text{store}}) \end{array}$$

with $C^l_{\text{store}} \models C_{\text{store}} \wedge c$ *and* $C^r_{\text{store}} \models C_{\text{store}} \wedge \neg c$.

Intuitively, the reason for distinguishing between positive and negative constraints is just that C_n describes the previously found solutions, i.e.

$$\forall v' \prec_t v \forall \alpha \in \|C_{\text{store}}(v')\| : (\alpha \models \neg C_n).$$

Here, we denote with $\prec_t$ the partial ordering of nodes induced by t.

Note that we do not force C^l_{store} (resp. C^r_{store}) to be equivalent to $C_{\text{store}} \wedge c$ (resp. $C_{\text{store}} \wedge \neg c$). The reason is that we do have to add some additional constraints during search for excluding symmetries. Since C_{store} describes all constraints valid at some node v, we set $\|v\| = \|C_{\text{store}}\|$ for a node v in t with label $(C_p, C_n, C_{\text{store}})$.

We now state general properties of search tree in the presence of symmetries.

Definition 4 (Expanded, C_{Pr}-Complete and S-Reduced Trees). *The search tree t is* completely expanded *if every unary node $v = (C_p, C_n, C_{\text{store}})$, either $\bot \in C_{\text{store}}$, or $\|v\| = \{\alpha\}$ and C_{store} determines $\mathcal{X}$ to α. A search tree t is* C_{Pr}-complete *if for every $\alpha \in \|C_{\text{Pr}}\|$ there is a leaf v in t with $\{\alpha\} = \|v\|$. Let $\mathcal{S}$ be a symmetry set for C_{Pr}. A search tree is* C_{Pr}-complete w.r.t. $\mathcal{S}$ *if for every $\alpha \in \|C_{\text{Pr}}\|$ there is a leaf v for such that*

$$\|v\| = \{\alpha\} \ \vee \ \exists s \in \mathcal{S} \backslash \{\text{id}_{C_{\text{Pr}}}\} : \|v\| = \{s(\alpha)\}.$$

A search tree is $\mathcal{S}$-reduced *if for every leaf v with $\|v\| = \{\alpha\}$ we have*

$$\forall s \in \mathcal{S} \ \forall \textit{leafs } v' \textit{ with } v' \prec_t v : (\|v'\| = \{\alpha'\} \Rightarrow s(\alpha') \neq \alpha), \tag{1}$$

Note that (1) is equivalent to $\forall s \in \mathcal{S} \ \forall v' \neq v : (\|v'\| = \{\alpha'\} \Rightarrow s(\alpha') \neq \alpha)$, if $\mathcal{S}$ is closed under inversion.

Before we can show how to produce a $\mathcal{S}$-reduced search tree that is C_{Pr}-complete w.r.t $\mathcal{S}$, we first have to define how symmetries operate on constraints. In the following, we will assume that for every constraint c and every s, there is a constraint c' such that $s^*(\|c\|) = \|c'\|$, where we define s^* on sets of C_{Pr}-solutions by $s^*(A) = \{s(\alpha) \mid \alpha \in A\}$. For every c, there are usually different constraints c' with $s^*(\|c\|) = \|c'\|$. Hence, fix a function s_{con} on constraints such that

$$s^*(\|c\|) = \|s_{\text{con}}(c)\|.$$

In the next sections, we give examples of how to construct this function.

Proposition 1. *Let s_{con} be defined as before. Then s_{con} distributes over the Boolean operators, i.e., $s^*(\|c \wedge c'\|) = \|s_{\text{con}}(c) \wedge s_{\text{con}}(c')\|$, $s^*(\|c \vee c'\|) = \|s_{\text{con}}(c) \vee s_{\text{con}}(c')\|$ and $s^*(\|\neg c\|) = \|s_{\text{con}}(\neg c)\|$.*

Hence, we identify s_{con} with its homomorphic extension to constraint sets and arbitrary formulae. Now we can state a general mechanism for generating a search tree that is $\mathcal{S}$-reduced.

Definition 5 ($\mathcal{S}$-excluding). *Let $\mathcal{S}$ be a symmetry set. A search tree for C_{Pr} is* $\mathcal{S}$-excluding *if every binary node v has the form*

$$\begin{tikzcd}[column sep=small] & (C_p, C_n, C_{\text{store}}) \arrow[dl, dash, "c"'] \arrow[dr, dash, "\neg c"] & \\ (C_p \wedge c, C_n, C^l_{\text{store}}) & & (C_p, C_n \wedge \neg c, C^r_{\text{store}}) \end{tikzcd}$$

with $C^l_{\text{store}} = P(C_{\text{store}} \wedge c)$ *and*

$$C^r_{\text{store}} = P(C_{\text{store}} \wedge \neg c \wedge E^v(\neg c)), \text{ where } E^v(\neg c) = \bigwedge_{s \in \mathcal{S}} s_{\text{con}}(C_p) \rightarrow \neg s_{\text{con}}(c).$$

Before we prove $\mathcal{S}$-reducedness and C_{Pr}-completeness w.r.t $\mathcal{S}$ of $\mathcal{S}$-excluding search trees, we state a proposition to precise the effect of adding the implications in definition 5 to C^r_{store}. For convenience, we write C^v_p, C^v_n and C^v_{store} to access C_p, C_n and C_{store} in the label $(C_p, C_n, C_{\text{store}})$ of v. For a binary node v we refer to its left child as v_l and to its right child as v_r.

Proposition 2. *Let $\mathcal{S}$ be a symmetry set and t a $\mathcal{S}$-excluding search tree. For every symmetry $s \in \mathcal{S}$ and for every node v of t we have $C^v_{\text{store}} \models s_{\text{con}}(C^v_p) \rightarrow s_{\text{con}}(C^v_n)$.*

For notational convenience we introduce the notation $C_{path}(v) = C^v_p \wedge C^v_n \wedge C_{\mathrm{Pr}}$. A good intuition is to think of $C_{path}(v)$ describing the constraint store of v in a simple not symmetry excluding search tree.

Reducedness and completeness will be essentially corollaries to the following proposition.

Lemma 1. *Let $\mathcal{S}$ be a symmetry set. Every $\mathcal{S}$-excluding search tree t for C_{Pr} satisfies for every node v*

$$\begin{aligned} \|v\| &= \|C_{path}(v)\| - A^v_{\mathcal{S}}, \\ \textit{where} \quad A^v_{\mathcal{S}} &= \{s(\alpha) \mid s \in \mathcal{S} \wedge \exists v' \prec_t v : \alpha \in \|C_{path}(v')\|\}. \end{aligned} \tag{2}$$

Proof. We proof this by tree induction. For the root this is valid, since $A^v_{\mathcal{S}} = \emptyset$ and $\|v\| = \|C_{\mathrm{Pr}}\| = \|C_{path}(v)\|$. Assume that we have proven the claim for a binary node v:

$$\begin{tikzcd}[column sep=small] & (C_p, C_n, C_{\text{store}}) \arrow[dl, dash, "c"'] \arrow[dr, dash, "\neg c"] & \\ (C_p \wedge c, C_n, C^l_{\text{store}}) & & (C_p, C_n \wedge \neg c, C^r_{\text{store}}) \end{tikzcd}$$

For the left child v_l the claim follows immediately from the induction hypotheses, since $A^{v_l}_{\mathcal{S}} = A^v_{\mathcal{S}}$.

For v_r, we have to show $\|v_r\| = \|C_{path}(v_r)\| - A^{v_r}_{\mathcal{S}}$. Note that $A^{v_r}_{\mathcal{S}} = \{s(\alpha) \mid s \in \mathcal{S} \wedge \exists v' \prec_t v_r : (\alpha \in \|C_{path}(v')\|)\}$ subdivides into its two subsets $A^v_{\mathcal{S}}$ and

$\{s(\alpha) \mid s \in \mathcal{S} \wedge \exists v'(v' \not\prec_t v \wedge v' \prec_t v_r \wedge \alpha \in \|C_{path}(v')\|)\}$. Further according to our definitions $C_{path}(v_r)$ is equivalent to $C_{path}(v) \wedge \neg c$.

To show that $\|v_r\| \subseteq \|C_{path}(v) \wedge \neg c\| - A_{\mathcal{S}}^{v_r}$, fix a symmetry $s \in \mathcal{S}$, then we have to show for every $v' \prec_t v$ that

$$\forall \alpha \in \|C_{path}(v')\| : s(\alpha) \notin \|v_r\|.$$

The first case is $v' \prec_t v$. Let $\alpha \in \|C_{path}(v')\|$. Then $s(\alpha) \notin \|v\|$ by induction hypotheses. Since $\|v_r\| \subseteq \|v\|$, this immediately implies $s(\alpha) \notin \|v_r\|$.

The second case is $(v' \not\prec_t v)$ and $v' \prec_t v_r$, i.e., v' is a subnode of v_l. Let $\alpha \in \|v_l\|$ and assume $s(\alpha) \in \|v_r\|$. From $\alpha \in \|v_l\|$ we have $\alpha \models C_p \wedge c$ and $s(\alpha) \models s_{\text{con}}(C_p \wedge c)$. That's a contradiction, because from definition of $\mathcal{S}$-excluding search tree $s(\alpha) \models s_{\text{con}}(C_p) \rightarrow \neg s_{\text{con}}(c)$, since $(s_{\text{con}}(C_p) \rightarrow \neg s_{\text{con}}(c)) \in E^v(\neg c)$.

It remains to be shown that $\|v_r\| \supseteq \|C_{path}(v_r)\| - A_{\mathcal{S}}^{v_r}$ Let $\alpha \notin \|v_r\|$. We have to show

$$\alpha \notin \|C_{path}(v_r)\| - A_{\mathcal{S}}^{v_r} \tag{3}$$

We have the following cases:
1. $\alpha \notin \|C_{path}(v_r)\|$. Then (3) follows immediately.
2. $\alpha \in \|C_{path}(v_r)\|$ and $\alpha \notin \|v\|$. We have to show $\alpha \in A_{\mathcal{S}}^{v_r}$.

Now $\alpha \in \|C_{path}(v_r)\|$ implies $\alpha \in \|C_{path}(v)\|$ and henceforth $\alpha \in A_{\mathcal{S}}^{v} \subseteq A_{\mathcal{S}}^{v_r}$ by induction hypotheses.
3. $\alpha \in \|C_{path}(v_r)\|$ and $\alpha \in \|v\|$. We will show that

$$\begin{aligned} \alpha \in A_{\mathcal{S}}^{v_r} - A_{\mathcal{S}}^{v} &= \{s(\beta) \mid s \in \mathcal{S} \wedge \exists v' : (v' \prec_t v_r \wedge v' \not\prec_t v \wedge \beta \in \|C_{path}(v')\|\} \\ &= \{s(\beta) \mid s \in \mathcal{S} \wedge \beta \in \|C_{path}(v_l)\|\} \end{aligned} \tag{4}$$

Since $\alpha \models C_{\text{store}} \wedge \neg c$, but $\alpha \not\models C_{\text{store}}^r$, there is at least one $s \in \mathcal{S}$ with $\alpha \not\models s_{\text{con}}(C_p) \rightarrow \neg s_{\text{con}}(c)$ by the definition of an S-excluding tree. Fix one of these symmetries s. Then $\alpha \models s_{\text{con}}(C_p) \wedge s_{\text{con}}(c)$. Since $C_{\text{store}} \models s_{\text{con}}(C_p) \rightarrow s_{\text{con}}(C_n)$ by Proposition 2, we get

$$\alpha \models s_{\text{con}}(C_p) \wedge s_{\text{con}}(c) \wedge s_{\text{con}}(C_n).$$

Hence, $s^{-1}(\alpha)$, which exists by definition of symmetries, satisfies $s^{-1}(\alpha) \models C_{path}(v) \wedge c$ (recall that $C_{path}(v) = C_p \wedge C_n$). I.e., $s^{-1}(\alpha) \in \|C_{path}(v_l)\|$. Hence, with $\beta = s^{-1}(\alpha)$ we have a valuation such that $\beta \in \|C_{path}(v_l)\|$ and $\alpha = s(\beta)$, which shows that α is in the set defined by (4).

Theorem 1. *Let $\mathcal{S}$ be a symmetry set. Every $\mathcal{S}$-excluding search tree t for C_{Pr} is $\mathcal{S}$-reduced.*

Proof. Fix a symmetry $s \in \mathcal{S}$, let v be a leaf of t with $\|v\| = \{\alpha\}$. We have to show for every node v' of t with $v' \prec_t v$ that $\|v'\| = \{\alpha'\} \rightarrow s(\alpha') \neq \alpha$. Assume v' to be a node of t with $v' \prec_t v$ and $\|v'\| = \{\alpha'\}$ and $s(\alpha') = \alpha$. It follows $\alpha \in \{s(\alpha) \mid s \in \mathcal{S} \wedge \exists v' \prec_t v : \alpha \in \|C_{path}(v')\|\}$, and from this the contradiction $\alpha \notin \|v\|$ by Lemma 1.

By now we understand that an $\mathcal{S}$-excluding search tree may exclude more symmetries than actually declared by $\mathcal{S}$ directly. Hence, we have to investigate the completeness property of $\mathcal{S}$-excluding search trees.

Theorem 2. *Let $\mathcal{S}$ be a symmetry set. Every $\mathcal{S}$-excluding search tree t for C_{Pr} is C_{Pr}-complete w.r.t $\mathcal{S}'$, where $\mathcal{S}'$ is the closure of $\mathcal{S}$ under composition and inversion.*

Proof. We have to show that t is C_{Pr}-complete w.r.t. to $\mathcal{S}'$, where $\mathcal{S}'$ is the closure under composition of $\mathcal{S}$, i.e. for every $\alpha \in \|C_{\mathrm{Pr}}\|$ there is a leaf v with

$$\|v\| = \{\alpha\} \;\vee\; \exists s \in \mathcal{S}' : \|v\| = \{s(\alpha)\}. \tag{5}$$

Hence fix an $\alpha \in \|C_{\mathrm{Pr}}\|$. Then there exists a leaf v' in t with $\alpha \in \|C_{path}(v')\|$ from definition 5. There are two cases.

1. $\alpha \in \|v'\|$. This yields immediately $\|v'\| = \{\alpha\}$ and (5) is proven.
2. $\alpha \notin \|v'\|$. By Lemma 1 follows $\alpha \in A_{\mathcal{S}}^{v'}$, i.e., there is a $s_1 \in \mathcal{S}$ and a $v'' \prec_t v'$ such that $\alpha = s_1(\alpha')$, where $\alpha' \in \|C_{path}(v'')\|$. By induction we get a sequence of symmetries $s_1, \ldots, s_n \in \mathcal{S}$, where

$$\alpha = s_1 \circ \cdots \circ s_n(\alpha^{(n)}) \;\text{ and }\; \{\alpha^{(n)}\} = \|v^{(n+1)}\|$$

for a leaf $v^{(n+1)}$ of t. Since $\mathcal{S}'$ is the closure of $\mathcal{S}$, claim (5) follows.

We will given an example for symmetry excluding search trees in Secion 5, where the symmetrie group of permutation are excluded by the subset of all transpositions (as indicated by the above theorem).

Now we are able to give a more detailed comparison with [8]. The SBDS-method therein is essential our symmetry exclusion restricted to the case where the search tree branches over constraints of the form $Var = Val$. In this case, the constraint collected in C_p is of the form $Var_1 = Val_1 \wedge \ldots \wedge Var_n = Val_n$, i.e., C_p is a partial assignment A. For the restricted SBDS-method, consider the case that a symmetry S_σ is defined by a permutation σ first. Then $S_\sigma(Var = Val)$ is $Var = \sigma(Val)$. Now let C_p be $Var_1 = Val_1 \wedge \ldots Var_n = Val_n$ at some node v of the search tree, which branches over $Var = Val$. Then the constraint added by our method for excluding the symmetry S_σ is $S_\sigma(Var_1 = Val_1 \wedge \ldots \wedge Var_n = Val_n) \implies \neg S_\sigma(Var = Val)$, which is by the definition of S_σ the same as

$$Var_1 = S_\sigma(Val_1) \wedge \ldots \wedge Var_n = S_\sigma(Val_n) \rightarrow Var \neq S_\sigma(Val).$$

Now the antecedence is nothing else than a test whether S_σ leaves C_p (i.e., the partial assignment at the node v) unchanged. Thus, the entailment will be evaluated and removed directly at the right subnode of v. The restricted SBDS-method essentially replaces the entailment test by the test whether the symmetry leaves the assignment unchanged. Only those symmetries are excluded, where the test can be decided at the node the corresponding implication would be integrated by our method (even if the corresponding implication would delay this test to a later point).

Note on implementation For the symmetry exclusion we have to add implications of the form $s_{\mathrm{con}}(C_p^v) \to s_{\mathrm{con}}(c)$ during the search procedure. Hence, one has to change the search procedure for implementing the symmetry exclusion. The antecedents of these implications can be computed during the search efficiently, using the observation that the antecedences of the added implications of ancestor nodes are prefixes of the ones added in their offspring. To be more precise, let v be a node, which branches over the constraint c, and let v^l (resp. v^r) its left (resp. right) daughter. Now

$$s_{\mathrm{con}}(C_p^{v^l}) = s_{\mathrm{con}}(C_p^v \wedge c) = s_{\mathrm{con}}(C_p^v) \wedge s_{\mathrm{con}}(c).$$

For v_r, $s_{\mathrm{con}}(C_p^{v_r})$ equals $s_{\mathrm{con}}(C_p^v)$. Therefore the computation of the condition $s_{\mathrm{con}}(C_p)$ can be done incrementally. We maintain Boolean variables for this reason in our implementation. We need one variable for each symmetry s, containing the values for $s_{\mathrm{con}}(C_p)$.

4 Geometric Symmetries

We will give now a concrete example for symmetries, namely geometry symmetries. We will treat the special case where we have points in $\mathbb{Z}^d$, although our method is not restricted to this case. We will exemplify the symmetry constructions in two dimensions, and give an example for the exclusion of additional symmetries as indicated by Theorem 2.

Definition of Geometric Symmetries There are many problems where one encodes the two-dimensional position of an object i using finite domain integer variables X_i, Y_i. Examples are the N-queens and the tiling problem. A more complex and realistic example (in three dimensions) is the lattice protein structure prediction problem [2].

The symmetries for $\mathbb{Z}^2$ have exactly the same structure as for the general case $\mathbb{Z}^d$. They are defined by affine mappings $S : \mathbb{Z}^d \to \mathbb{Z}^d$ with $S(\boldsymbol{x}) = A_S\boldsymbol{x} + \boldsymbol{v}_S$ that map $\mathbb{Z}^d$ onto $\mathbb{Z}^d$. I.e., the matrix A_S is an orthogonal matrix with the property that the set of columns $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_d\}$ of A_S equals $\{\pm\boldsymbol{e} \mid \boldsymbol{e}$ is a unit-vector of $\mathbb{Z}^d\}$. E.g., for $\mathbb{Z}^2$, the matrix $\left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)$ denotes the rotation by 90°. For $\mathbb{Z}^2$, we have 8 symmetries consisting of the identity, the 4 reflections (at x- and y-axis, and the two diagonals) and the 3 rotations by 90°, 180° and 270°. For $\mathbb{Z}^3$, we have 48 symmetries including the identity.

By now, the vector $\boldsymbol{v}_S$ is not yet fixed. There are two different approaches for fixing the symmetry. We will consider $\mathbb{Z}^2$ as an example. The methods work for all other dimensions as well.

The first case is that every possible solution lies within a fixed square (in the general case, within a hypercube).[2] This is equivalent to the proposition that there are integers $x_{min}, x_{max}, y_{min}, y_{max}$ such that for all $\alpha \in \|C_{\mathrm{Pr}}\|$ we have

$$\begin{aligned} \min\{\alpha(X_i)\} &= x_{min} \qquad & \max\{\alpha(X_i)\} &= x_{max} \\ \min\{\alpha(Y_i)\} &= y_{min} \qquad & \max\{\alpha(Y_i)\} &= y_{max}. \end{aligned}$$

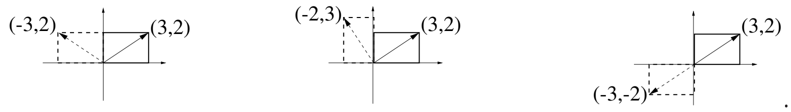
[2] The technique can be extended to the case that the hypercube is not fixed in advance.

Thus, the minimal square around the position of all objects is defined by the points (x_{min}, y_{min}) and (x_{max}, y_{max}) in every solution. We call this the *frame* of the problem C_{Pr}. In the N-queens problem, this is just the board.

Now knowing the frame of a problem C_{Pr}, we can fix the vector $\boldsymbol{v}_S$ for all symmetries. Consider as an example a problem whose frame is defined by $(0,0)$ and $(3,2)$. Furthermore, consider the three symmetries reflection at the y-axis, rotation by 90° and rotation by 180°, which we will name S_1, S_2 and S_3 in the following. The corresponding mappings are defined by $S_i(\boldsymbol{x}) = A_{S_i}(\boldsymbol{x}) + \boldsymbol{v}_{S_i}$, where the matrices A_{S_1}, A_{S_2} and A_{S_3} are defined by

$$A_{S_1} = \left(\begin{smallmatrix} -1 & 0 \\ 0 & 1 \end{smallmatrix}\right) \qquad A_{S_2} = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right) \qquad A_{S_3} = \left(\begin{smallmatrix} -1 & 0 \\ 0 & -1 \end{smallmatrix}\right).$$

The corresponding mappings of the frame are



A symmetry S is compatible with the above defined frame if the frame is mapped to itself, i.e., if $\{\boldsymbol{v} \mid (0,0) \leq \boldsymbol{v} \leq (3,2)\} = \{S(\boldsymbol{v}) \mid (0,0) \leq \boldsymbol{v} \leq (3,2)\}$. For a given matrix A_S, there exists a $\boldsymbol{v}_S$ such that $S(\boldsymbol{x}) = A_S\boldsymbol{x} + \boldsymbol{v}_S$ satisfies this condition if and only if A_S satisfies

$$A_S(3,2) = (\pm 3, \pm 2) \tag{6}$$

For the matrices A_{S_1}, A_{S_2} and A_{S_3}, we get $(-3,2)$, $(-2,3)$ and $(-3,-2)$, which excludes the symmetry characterized by A_{S_2}. We finally get

$$\boldsymbol{v}_{S_1} = (3,0) \quad \text{and} \quad \boldsymbol{v}_{S_3} = (3,2).$$

The second case is that we know a point $\boldsymbol{p} = (p_x, p_y)$ which should remain unchanged under the symmetries. In that case, we know that the symmetries are defined by $S_i(\boldsymbol{x}) = A_{S_i}(\boldsymbol{x} - \boldsymbol{p}) + \boldsymbol{p} = A_{S_i}(\boldsymbol{x}) - A_{S_i}(\boldsymbol{p}) + \boldsymbol{p}$. Hence, $\boldsymbol{v}_{S_i} = \boldsymbol{p} - A_{S_i}(\boldsymbol{p})$.

The remaining part is to define symmetric constraints. We use a specific example where we leave the point $(5,5)$ fix. Consider the two symmetries reflection at the y-axis and rotation by 90°. By what we have said above, the corresponding mappings are

$$S^{\text{Ry}}(\boldsymbol{x}) = \left(\begin{smallmatrix} -1 & 0 \\ 0 & 1 \end{smallmatrix}\right)\boldsymbol{x} + \left(\begin{smallmatrix} 10 \\ 0 \end{smallmatrix}\right) \quad \text{and} \quad S^{90^\circ}(\boldsymbol{x}) = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)\boldsymbol{x} + \left(\begin{smallmatrix} 10 \\ 0 \end{smallmatrix}\right)$$

Now suppose that we have modeled points $p_1, \ldots, p_n$ using variables $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$, and we want to define S^{Ry}_{con} and $S^{90^\circ}_{con}$ for the constraints of the form $X_i = c$ (for other kind of constraints c, the definition $S_{con}(c)$ is analogous). Now the symmetric constraints $S^{\text{Ry}}_{con}(X_i = c)$ (resp. $S^{90^\circ}_{con}(X_i = c)$) must express the constraint valid for $S^{\text{Ry}}(\alpha)$ (resp. $S^{90^\circ}(\alpha)$) for every possible α with

$\alpha \models X_i = c$. Then

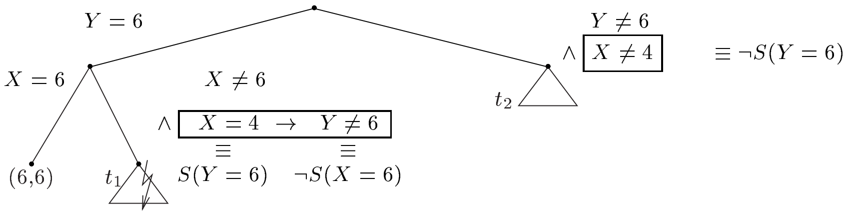
$$\alpha \models X_i = c \Leftrightarrow \boldsymbol{p_i} = (c, \alpha(Y_i)) \qquad\qquad \alpha \models X_i = c \Leftrightarrow \boldsymbol{p_i} = (c, \alpha(Y_i))$$
$$\Leftrightarrow S^{\mathrm{Ry}}(\boldsymbol{p_i}) = (10 - c, \alpha(Y_i)) \qquad\qquad \Leftrightarrow S^{90^\circ}(\boldsymbol{p_i}) = (10 - \alpha(Y_i), c)$$
$$\Leftrightarrow S^{\mathrm{Ry}}(\alpha) \models X_i = 10 - c \qquad\qquad \Leftrightarrow S^{90^\circ}(\alpha) \models Y_i = c$$

Since $X_i = c$ does not restrict the valuation of $\alpha(Y_i)$, we know that $S^{\mathrm{Ry}}_{con}(X_i = c)$ is $X_i = 10-c$, and $S^{90^\circ}_{con}(X_i = c)$ is $Y_i = c$. Analagously, we get that $S^{\mathrm{Ry}}_{con}(Y_i = c)$ is $Y_i = c$, and that $S^{90^\circ}_{con}(Y_i = c)$ is $X_i = 10 - c$. Note that $S^{\mathrm{Ry}}(c)$ has the same type as c (i.e., both have the same variable). This does not hold for c and $S^{90^\circ}(c)$.

We will now turn to an example where we exclude only one symmetry, but the exclusion method excludes compositions of this symmetry. This shows that there are cases where the symmetry exclusion method excludes more than the symmetries contained in the symmetry set $\mathcal{S}$ (as indicated by Theorem 2). We will use an artificial example to show the existence of such a case.

Example 1. Let C_{Pr} be a geometric constraint problem over the variables X, Y, where X and Y have the domain $[0..10]$ associated. Furthermore, suppose that C_{Pr} has $(6, 6)$ as a solution that we can apply the rotational symmetries with fix point $(5, 5)$. This implies, that $(4, 4)$, $(4, 6)$ and $(6, 4)$ are also solutions of C_{Pr}.

Now consider the following $\mathcal{S}$-excluding search tree for $\mathcal{S} = \{S^{90^\circ}\}$ (where the constraints added by the symmetry exclusion are in a framed box):



Then the solution $(4, 6)$ (which is $(6, 6)$ rotated by $90°$) is excluded by the symmetry exclusion in t_1. But furthermore, the solution $(4, 4)$ is excluded in t_2 by our symmetry exclusion method. This is $(6, 6)$ rotated by $180°$. Hence, although $\mathcal{S}$ contains only S^{90°, the symmetry S^{180° (which is just $S^{90^\circ} \circ S^{90^\circ}$) is also excluded by our method. Note that $(6, 4)$ (which is $(6, 6)$ rotated by $270°$) is not excluded and may be found in t_2.

Clearly, in the case of geometric symmetries, the full symmetry group is usually small enough that we can include all symmetries. But we wanted to show the effects of excluding only a subset of the full symmetry group using the above simple example. In the next section on permutation, we will give an example where the exclusion of transpositions allows to exclude all permutations.

Results We have applied the method to the lattice protein structure prediction [2], which is a hard combinatorical problem. The following table shows the number of solutions, search steps and runtimes for finding all minimal energy structures for 4 sequences, in the first case without symmetry exclusions, and with symmetry exclusion in the second part:

		without sym. excl.			with sym. excl.			ratio		
seq.	length	$\#_{noex}$	n_{noex}	t_{noex}	$\#_{ex}$	n_{ex}	t_{ex}	$\frac{\#_{noex}}{\#_{ex}}$	$\frac{n_{noex}}{n_{ex}}$	$\frac{t_{noex}}{t_{ex}}$
S1	27	4,752	45,924	15.1 min	297	2,998	69 sec	16	15.32	13.17
S2	27	408,864	2,465,728	6.2 h	25,554	155,693	26.3 min	16	15.84	14.18
S3	31	53,472	351,101	3.4 h	1,114	11,036	7.4 min	48	31.81	27.43
S4	36	56,448	732,952	3.7 h	3,528	55,086	19.9 min	16	13.31	11.24

We have added simple exclusion methods in both cases, which is the reason that we have only 16 (instead of 48) symmetries left in $S1, S2$ and S_6. As one can see from the table, we have a nearly linear speedup in number of search steps, and a close to linear speedup in runtime. In $S3$, the simple exclusion does not apply, which gives rise to 48 symmetries. Furthermore, the optimal conformations and the symmetric ones are very similar, which implies that we have less speedup (since detecting the symmetric solution is harder).

5 Permutation symmetries

In the following, we consider finite domain integer problems, where the variables have the domain $D \subseteq \mathbb{N}$ associated. In the following, we denote with $\mathrm{Perm}(D)$ the set of all permutations of D, and with $\mathrm{Trans}(D)$ the set of all transpositions of D (i.e., those permutations, which exchange just two elements of D). With $\mathcal{S}^{\mathcal{X}}_{\mathrm{Perm}(D)}$, we denote the set of all symmetries that permute the values of the variables in $\mathcal{X}$.

Definition 6. *Let $\mathcal{X}$ be the set of solution variables. The set of all value permutations of $\mathcal{X}$ is defined as the symmetry set*

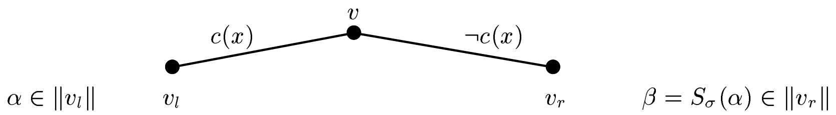
$$\mathcal{S}^{\mathcal{X}}_{\mathrm{Perm}(D)} = \{S \mid \exists \pi \in \mathrm{Perm}(D) \forall \alpha : S(\alpha)(X) = \pi(\alpha(X))\}$$

The subset of all transposition symmetries $\mathcal{S}^{\mathcal{X}}_{\mathrm{Trans}(D)}$ is defined analogously.

Now we consider search trees that branch over constraints having one single free variable (e.g., constraints of the forms $X = c$, $X < c$, $Even(X)$ and so on). The interesting part is that if a search tree branches over constraints that have one single free variable, then every $\mathcal{S}^{\mathcal{X}}_{\mathrm{Trans}(D)}$-excluding tree is even $\mathcal{S}^{\mathcal{X}}_{\mathrm{Perm}(D)}$-reduced. Thus, we need to exclude only the quadratically many transpositions to exclude the exponentially many permutations in this case.

Theorem 3. *Let C_{Pr} be a problem which has $\mathcal{S}^{\mathcal{X}}_{\mathrm{Perm}(D)}$ as a symmetry group. Let t be a $\mathcal{S}^{\mathcal{X}}_{\mathrm{Trans}(D)}$-excluding search tree such that for every branch v, the constraint c used at this branch has one free variable out of $\mathcal{X}$. Then t is $\mathcal{S}^{\mathcal{X}}_{\mathrm{Perm}(D)}$-reduced.*

Proof (sketch). Consider a permutation σ and the corresponding symmetry S_σ. Now consider any node



Since $\alpha \models c(x)$ and $S_\sigma(\alpha) \models \neg c(x)$, we know that $\sigma(\alpha(x))$ must be different from $\alpha(x)$. Now let τ be the transposition such that $\tau(\alpha(x)) = \sigma(\alpha(x))$, and let S_τ be the corresponding symmetry. Then the exclusion of all transposition would exclude the solution β if we can show that in the node considered above, we have that $\mathcal{S}_\tau{}^{-1} \circ S_\sigma(\beta) \in \|v_l\|$. But this can be shown under the assumption that the search tree uses only constraints that have one free variable. □

Graph Coloring As noted above the graph coloring is an example for a problem with value permutations as symmetries. The graph coloring problem is as follows. Given a graph $G = (V, E)$ and a set of colors. An admissible coloring c of G, is a mapping of the vertices V to the set of colors, satisfying the constraint that for all $(v_1, v_2) \in E$: $c(v_1) \neq c(v_2)$. What is the minimal set of colors, where we still can find an admissible coloring.

In this problem colorings are considered to be symmetric, if they are just permuted in colors to each other.

We compare two implementations of a solver for this problem. The first one is a naive implementation that uses a simple first fail heuristic. This implementation does not have symmetry exclusion. The second implementation with full exclusion of symmetries is just a simple extension of the first one, where we added our symmetry exclusion mechanism. Note that, due to the above theorem, it suffices to handle only the set of transpositions instead of handling the group of all permutations to get the full symmetry exclusion.

We give the results for some randomly generated problem instances of different problem sizes.

problem size			with symmetry excl.			without symmetry excl.			ratio	
verts	edges	colors	cloned	failed	time	cloned	failed	time	cloned	time
20	114	6	24	16	50	162	154	90	6.8	1.8
18	101	7	35	22	70	887	874	450	25.3	6.4
25	185	8	61	47	170	52,733	52,719	29,710	864.5	487.0
50	506	8	434	406	2,280	1,440,549	1,440,521	1,588,340	3,319.2	696.6
30	292	9	97	81	230	551,022	551,006	385,100	5,680.6	1674.3
35	339	9	168	150	370	3,428,853	3,428,835	2,569,200	20,409.8	6943.8

6 Conclusion and Future Work

We have introduced the first method that can exclude *arbitrary* symmetries in constraint-based search. In contrast to many other methods that can be found in the literature, the method does not restrict the search strategy.

The method is based on a declarative description of the symmetries in the form of symmetric constraints, which usually can be obtained easily. We have given completeness and correctness proofs as well as the proof, that the method exclude all considered symmetries. In the case that the full symmetry group is too large, our method can handle arbitrary subsets of the full symmetry group. We have shown for the case of permutation symmetries that it is sufficient to exclude the (quadratically many) transposition to exclude all (exponentially many) permutations.

The method can be implemented in any (constraint-based) search machinery that handles implications and allows to introduce additional, user-defined constraints during search. This holds for the most modern constraint programming systems. Since the method is very general, we plan to investigate in which other (logic-based) search methods our symmetry exclusion can be used. Furthermore, we intend to investigate more general conditions under which a subset of the full symmetry group allows to exclude all symmetries (as it is the case for the permutations).

References

1. Alfonso San Miguel Aguirre. How to use symmetries in boolean constraint solving. In Frdric Benhamou and Alain Colmerauer, editors, *Constraint Logic Programming, Selected Research*, chapter 16, pages 287–306. The MIT Press, 1993.
2. Rolf Backofen. Constraint techniques for solving the protein structure prediction problem. In *Proceedings of* 4^{th} *International Conference on Principle and Practive of Constraint Programming (CP'98)*, 1998.
3. Belaid Benhamou. Study of symmetry in constraint satisfaction problems. In Alan Borning, editor, *Principles and Practice of Constraint Programming, Second International Workshop, PPCP'94,*, volume 874 of *Lecture Notes in Computer Science*. Springer, 1994.
4. Belaid Benhamou and Lakhdar Sais. Tractability through symmetries in propositional calculus. *Journal of Automated Reasoning*, 12:89–102, 1994.
5. James M. Crawford, Matthew Ginsberg, Eugene Luks, and Amitabha Roy. Symmetry breaking predicates for search problems. In *Proc. of the* 5^{th} *International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, pages 149–159, 1996.
6. Thierry Boy de la Tour. Minimizing the number of clauses by renaming. In M. E. Stickel, editor, *Proc. of the* 10^{th} *Int. Conf. on Automated Deduction (CADE90)*, number 449 in LNAI, pages 558–572. Springer, 1990.
7. Eugene Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *Proc. of AAAI'91*, pages 227–233, 1991.
8. Ian P. Gent and Barbara Smith. Symmetry breaking during search in constraint programming. Report 99.02, University of Leeds, January 1999.
9. Christian Schulte, Gert Smolka, and Jörg Würtz. Encapsulated search and constraint programming in Oz. In A.H. Borning, editor, *Second Workshop on Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, vol. 874, pages 134–150, Orcas Island, Washington, USA, May 1994. Springer-Verlag.
10. Gert Smolka. The Oz programming model. In Jan van Leeuwen, editor, *Computer Science Today*, Lecture Notes in Computer Science, vol. 1000, pages 324–343. Springer-Verlag, Berlin, 1995.