A Theoretical Analysis of Reasoning By Symmetry in First-Order Logic (Extended Abstract)

James M. Crawford

AT&T Bell Laboratories
600 Mountain Ave.
Murray Hill, NJ 07974-0636
jc@research.att.com

Abstract

Many classes of reasoning problems display a large amount of symmetry. In mathematical and common-sense reasoning, such symmetries are often used to reduce the difficulty of reasoning. In this paper we show how symmetries can be used in automated reasoning both to reduce or avoid case analysis, and to reduce the scope of existential quantification. We show further that the computational complexity of the symmetry detection problem is equivalent (within a polynomial factor) to that of the graph isomorphism problem. This result is significant because the complexity of the graph isomorphism problem is a hard open problem (it is believed to lie between P and NP). Further, for graphs with bounded degree, graph isomorphism is known to be polynomial decidable. We then show how Pólya's theorem can be used to count the number of interpretations of a logical theory which are distinct under a given set of symmetries. This provides a method for predicting the effect of the symmetries on the ease of reasoning with the theory. We consider several commonly occurring types of symmetries and show how they affect the size of the space of distinct interpretations. Finally, we verify experimentally that in one well studied problems class, pigeonhole problems, the use of symmetries succeeds in reducing the complexity of determining unsatisfiability from exponential (using resolution or a tableau based method) to polynomial.

1 Introduction

Many classes of reasoning problems display a large amount of symmetry. By exploiting these symmetries one can often reduce the complexity of reasoning — in some cases by an exponential factor. Symmetries have been used in some domains for many years (*e.g.*, symmetries in the eight queens problem were taken advantage of in [Glaisher 1874]), but comparatively little attention seems to have been paid to the general problem of detecting and utilizing symmetries in first-order logic (see section 6 below).

In this paper we present three theoretical results concerning the use of symmetries in reasoning. First, we show that there is a simple relationship between symmetries of logical theories and symmetries of the models of these theories. We examine the implications of this relationship for satisfiability testing and deduction, and show how one can use symmetries to take inferential "shortcuts". Second, we show that the computational complexity of the symmetry detection problem is equivalent (within a polynomial factor) to that of the graph isomorphism problem. This result is significant because the complexity of the graph isomorphism problem is a hard open problem (it is believed to lie between P and NP). Further, for graphs with bounded degree, graph isomorphism is known to be polynomial decidable [Luks 80]. We thus define a notion of the degree of a theory and show that for theories of bounded degree, symmetries can be detected in polynomial time. Third, we show how Pólya's theorem can be used in the propositional case to count the number of interpretations of a logical theory which are distinct under a given set of symmetries. We consider several commonly occurring types of symmetries (*e.g.*, coloring problems, problems with geometric symmetries, and constraint satisfactions problems involving a collection of interchangeable objects), and show how symmetries in these problems affect the size of the space of distinct interpretations. Finally, we verify experimentally that in one well studied problems class, pigeonhole problems, the use of symmetries succeeds in reducing the complexity of determining unsatisfiability from exponential (using resolution or a tableau based method) to polynomial.

2 The Use of Symmetries in Deduction

Intuitively, a first-order theory is symmetric if one can interchange some of the constants, function symbols, and relations in the theory and end up with a logically equivalent theory. More formally, consider a syntactic sub-

stitution θ mapping constants, function symbols, and relations to constants, function symbols, and relations (respectively). We say that such a θ is a *symmetry* of a first-order theory Th iff both (1) θ is a one-to-one, and (2) $\theta(Th) \Leftrightarrow Th$.[1]

Note first that condition (1) guarantees that for any symmetry θ of a first-order theory Th, Th is satisfiable iff $\theta(Th)$ is satisfiable (θ just changes around the names of the constants, functions, and relations, so any model M of a theory Th can trivially be used to construct a model of $\theta(Th)$).

One can thus use symmetry to simplify satisfiability checking:

Lemma 2.1 For any symmetry θ of a first order theory Th, and any two well-formed formula p and q such that $\theta(p) \Rightarrow q$, and $Th \Rightarrow p \vee q$: Th satisfiable iff $Th \wedge q$ satisfiable.

The intuition behind this lemma is simply that if we are looking for a model of a symmetric theory Th, and we know $p \vee q$, then we can restrict our search by looking only for models of $Th \wedge q$ (and know that if no such model exists then there is no model of $Th \wedge p$). Since disjunctions are a primary source of intractability, such symmetries can greatly increase the efficiency of reasoning (see section 5).

Notice that we do not require that $p \vee q$ actually appears as a disjunction in the theory — only that it be implied by the theory. Consider, for example, the propositional theory $Th = \{A \vee B, C \vee B\}$. While there are no symmetries which interchange A and B, or C and B, there is the symmetry $\theta = \{A/C, C/A\}$. Notice that this symmetry interchanges the clause $A \vee B$ with the clause $C \vee B$. Together these clauses are equivalent to $(A \wedge C) \vee (A \wedge B) \vee (B \wedge C) \vee B$. By setting $p = (A \wedge C) \vee (A \wedge B) \vee B$ and $q = (A \wedge C) \vee (C \wedge B) \vee B$, we can apply lemma 2.1 to reduce the search space to $(A \wedge C) \vee (C \wedge B) \vee B$. In general, the search space due to two symmetric n-ary clauses can be reduced from n^2 to $\frac{n^2+n}{2}$.

We can similarly choose p and q so as to reduce the scope of existential quantification. Assume that θ is a symmetry of a first-order theory Th, and for some well-formed formula ϕ, $Th \Rightarrow (\exists x.\phi)$ and $\theta(\phi) = \phi$. If there is a pair of ground terms c and d such that $\theta(c) = d$ then we can set $p = \phi(x/c)$ (ϕ with x replaced by d), and $q = (\exists x.x \neq c \wedge \phi)$, and conclude from lemma 2.1 that Th satisfiable iff $Th \wedge (\exists x.x \neq c \wedge \phi)$ satisfiable. This results can be generalized by replacing c by a set of ground terms C, and θ by a set of symmetries each of which maps a member of C to d. One can then restrict the scope of the quantification to not include C.

One can also use symmetry in inference:

Lemma 2.2 For any symmetry θ of a first order theory Th, any two well-formed formula p and q such that $\theta(p) \Rightarrow q$ and $Th \Rightarrow p \vee q$, and any well-formed formula l such $l \Rightarrow \theta(l)$: $Th \Rightarrow l$ iff $Th \wedge q \Rightarrow l$.

The intuition here is that if we want to deduce a symmetric result l, then we can soundly assume either side of a symmetric disjunct $p \vee q$. This type of inference is commonly used in mathematics (usually accompanied with a statement of the form "without loss of generality assume q"). This lemma can be shown to be equivalent to the "rule of symmetry" of [Krishnamurthy 85] (see section 6).

3 The Complexity of Symmetry Detection

Consider the problem of detecting symmetries. For simplicity, we assume we are working with theories that are in clausal form. Determining whether $\theta(Th) \Rightarrow Th$ is undecidable (one can show that in the worst case it reduces to tautology checking), so we restrict our attention to cases in which $\theta(Th) = Th$ and $\theta(p) = q$.[2] We refer to such symmetries as *simple symmetries*. By the *simple symmetry detection problem* we will mean the following:

> **Given:** A clausal theory Th, and two well-formed formula p and q.
> **Find:** A simple symmetry θ of Th, p, and q.

We then have:

Theorem 3.1 The simple symmetry detection problem is equivalent to graph isomorphism under polynomial reductions.[3]

Here we briefly overview the constructions mapping symmetry detection to graph isomorphism, and vice-versa. Details will be included in [Crawford 92].

First consider reducing the graph isomorphism problem to the symmetry detection problem. We will actually reduce graph isomorphism to symmetry detection in clausal propositional theories. Consider any two graphs G_1 and G_2. Let:

$$T_1 = \{a \vee b | \langle a, b \rangle \text{ is an edge in } G_1\}$$
$$T_2 = \{a \vee b | \langle a, b \rangle \text{ is an edge in } G_2\}$$

One can then show that G_1 and G_2 are isomorphic iff there exists a one-to-one mapping θ such that $\theta(T_1) =$

[1] Throughout this paper we use $\Rightarrow$ for first-order implication.

[2] Of course θ may permute the order of the clauses in the theory, or the order of terms in the clauses. One can formally define equality of theories as syntactic equality of theories in a normal form (where normalization consists of ordering the terms in the clauses, and then ordering the clauses lexographically).

[3] The graph isomorphism problem is the following: given a pair of graphs G_1 and G_2, determine whether there exists a one-to-one mapping from the nodes of G_1 to the nodes of G_2 such that adjacent nodes are mapped to adjacent nodes.

T_2. Now let c and d be propositional variables which do not occur in T_1 or T_2, and let:

$$Th = \{t_1 \vee c | t_1 \in T_1\} \bigcup \{t_2 \vee d | t_2 \in T_2\}$$

One can show that Th has a simple symmetry mapping c to d iff there is a θ such that $\theta(T_1) = T_2$.

Now consider reducing symmetry detection to graph isomorphism. We show the mapping for propositional theories (the extension to the first-order case is straightforward). First note that we can "type" the nodes in the graphs and only allow isomorphisms which preserve type (*i.e.*, which only map nodes of a given type to other nodes with the same type) without increasing the difficulty of the isomorphism problem.[4] We use five types of nodes: nodes for positive literals, nodes for negative literals, *inverse* nodes, nodes for clauses, and *goal* nodes. We first link (the node for) each literal p to an inverse node and then link this inverse node to (the node for) $\neg p$. These links ensure that any graph isomorphism preserves negation. We then create a node for each clause and link it to the literals appearing in the clause. These links force graph isomorphisms to map clauses to clauses. Finally, recall that we are required to find a θ which maps p to q. To force this we create two copies of the graph for the theory. In the first we give p the type *goal* and in the second we give q the type *goal*. This typing forces any isomorphism between the two graphs to map p to q. One can then show that an isomorphism between the graphs exists iff the theory contains a simple symmetry mapping p to q.

It is known that the graph isomorphism problem is polynomial for graphs of fixed degree. In the construction above, the degree of the graph produced is determined by the larger of (1) the length of the longest clause in the theory, and (2) the number of occurrences of the most commonly occurring literal.[5] [6] We refer to this measure as the *degree* of a theory. We then have:

Theorem 3.2 For theories satisfying a fixed degree bound, simple symmetry detection can be done in polynomial time.

The good news here is that if we increase the size of a theory (*e.g.*, by adding propositional variables and clauses containing them) but hold its degree fixed, then the difficulty of detecting simple symmetries grows polynomially. The bad news is that when we map hard problems from other domains to satisfiability problems, it is often the case that larger problems map to theories with larger degrees. For example, in the three-color problem the obvious mapping to propositional logic gives a theory with degree equal to the degree of the graph to be colored. This gives the interesting result that the complexity of detecting symmetries in propositional versions of three-color problems increases exponentially with the connectivity of the graphs (but only polynomially with the size of the graphs).

[4] To see this note that we can "label" nodes in the graph with a type by linking them to a clique which is larger than any other clique in the graph (the graphs we construct contain no cliques of size larger than three, so this labeling can be done without affecting the degree of the graphs).

[5] Note that this is *not* the same as the number of occurrences of the most commonly occurring *term* (*e.g.*, in $p \vee q$ and $\neg p \vee q$ we have one occurrence of p and one occurrence of $\neg p$).

[6] In the first-order case the degree is the maximum of these two factors and the maximum relation or function arity.

4 Assessing the Effects of Symmetries

For a propositional theory Th with n propositional variables, there are 2^n possible interpretations (*i.e.*, mappings from the variables to the set $\{t, f\}$). However, if the theory has a symmetry θ then one can show that an interpretation M is a model of the theory iff $\theta(M)$ is a model of the theory (where $\theta(M)$ maps p to $M(\theta(p))$). One can thus define an equivalence relation on models by: $M_1 \sim M_2$ iff there is a symmetry of θ of the theory such that $M_1 = \theta(M_2)$. Satisfiability testing then reduces to the problem of testing one model from each such equivalence class. The number of equivalence classes thus provides an estimate of the usefulness of a set of symmetries.

In this section we show how Pólya's theorem can be used to count the number of equivalence classes of models under a set of symmetries of a propositional theory. We briefly overview Pólya's theorem, and then show how it can be used to compute the number of distinct interpretations under several common types of symmetries.

4.1 Pólya's Theorem

Here we briefly review Pólya's theorem as it applies to counting interpretations of propositional theories. For details see [Williamson 85] or [Pólya & Read 87]. Consider a finite set D and a set A of permutations of the elements of D (*i.e.*, each member of A is a one-to-one function from D to D). Since D is finite we can write any element in A as a set of cycles (for example, $(xy)(z)$ represents the permutation mapping x to y, y to x, and z to z). For any $a \in A$, let $v(a, i)$ denote the number of cycles of length i in a.

Now consider the set F of mappings from D to a finite set R. We can define an equivalence relation on F by: $f_1 \sim f_2$ iff for some $a \in A$, $f_1 = f_2 \circ a$. Let F_A denote the set of equivalence classes of F induced by this relation. Pólya's theorem relates the size of F_A to the lengths of the cycles in the permutations in A:

$$|F_A| = \frac{1}{|A|} \sum_{a \in A} \prod_{i=1}^{|D|} |R|^{v(a,i)}$$

In the case of counting interpretations of logical theories we let D be the set of propositional variables, A be the set of symmetries of the theory, and R be $\{t, f\}$. F_A

is thus the set of distinct interpretations of the theory, and its cardinality is given by:

$$\frac{1}{|A|}\sum_{a\in A}\prod_{i=1}^{|D|}2^{v(a,i)} \quad (1)$$

4.2 Examples of the Application of Pólya's Theorem

In this section we show several applications of equation 1. These examples are relatively simple and are not intended to exhibit the full generality of Pólya's theorem (or even of its application to estimating the effects of symmetries in logical theories). Rather they are included to demonstrate how Pólya's theorem can be applied to symmetries in logical theories, and to give some examples of common types of problems in which the use of symmetries can significantly reduce the number of distinct interpretations of logical theories.

Consider first a simple example: let $Th = \{x \vee y\}$. In this case $D = \{x, y\}$ and $A = \{(x)(y), (xy)\}$. $v((x)(y), 1) = 2$, $v((xy), 2) = 1$, and all other $v(a, i)$ are zero. By equation 1, the number of distinct interpretations is thus $\frac{1}{2}(2^2 + 2^1) = 3$.

Now consider a theory with propositional variables $p_1, \ldots, p_n$. One extreme case is a theory with no symmetries. In this case $A = \{(p_1)(p_2)\ldots(p_n)\}$, and the only non-zero $v(a, i)$ is $v((p_1)(p_2)\ldots(p_n), 1) = n$. Equation 1 thus reduces to 2^n as expected.

A somewhat more interesting case is a theory in which any permutation of the propositional variables is a symmetry of the theory. Applications of Pólya's theorem to complete permutation groups are well understood, and equation 1 is known to reduce to the sum over all sets $\{\alpha_1, \ldots, \alpha_n\}$ such that $\alpha_1 + \alpha_2 + \ldots + n\alpha_n = n$ of:

$$\prod_{i=1}^{n}\frac{2^{\alpha_i}}{(\alpha_i! i^{\alpha_i})}$$

Fortunately, with more work this can be shown to reduce further to:

$$n + 1$$

An example of a theory with complete symmetry is the following (from [Cook 74]):

$$\begin{aligned} T_0 &= \{\} \\ T_n &= \{p_n \vee x | x \in T_{n-1}\} \cup \{\neg p_n \vee x | x \in T_{n-1}\} \end{aligned}$$

(T_n is thus the set of all possible n-ary disjunctions of the propositional variables $\{p_1, \ldots, p_n\}$). [Cook 74] shows that any analytic tableau based proof of the unsatisfiability of T_n requires at least 2^{2^n} nodes. Resolution refutations of length $2^n - 1$ are known to exist. Note, however, that the analysis above shows that only $n + 1$ distinct interpretations of T_n exist. We can thus show that T_n is unsatisfiable by showing $n + 1$ distinct interpretations (which can be easily constructed by letting M_n be the interpretation with $p_i = t$ iff $i < n$) and showing that each interpretation violates a clause in the theory (this is a particularly interesting example since if the symmetries of the theory are known *a priori*, the length of the proof of unsatisfiability is *logarithmic* in the size of the theory).

A slightly more complex case is a theory with n arbitrarily permutable propositional variables (as before), and m additional literals which are completely asymmetric. In this case a similar analysis shows that there are $2^m(n + 1)$ distinct interpretations (as expected).

A more common case is a theory containing classes of propositional variables, in which the classes can be arbitrarily interchanged. For example, in coloring problems one might use p_{ic} to represent "node i has color c". If the colors are interchangeable then any substitution which exchanges p_{ic_1} and p_{ic_2} (for all i) will be a symmetry of the theory. In general, if a theory consists of n permutable classes of literals and each class has size m, then equation 1 reduces (again using text book methods) to the sum over all sets $\{\alpha_1, \ldots, \alpha_n\}$ such that $\alpha_1 + \alpha_2 + \ldots + n\alpha_n = n$ of:

$$\prod_{i=1}^{n}\frac{(2^m)^{\alpha_i}}{\alpha_i! i^{\alpha_i}}$$

This can again be further reduced to:

$$\frac{(2^m + n - 1)!}{(2^m - 1)! n!} \quad (2)$$

which is equivalent to $2^m + n - 1$ choose n.

To get a feel for equation 2 we now consider two applications. First, consider a graph coloring problem. We can apply equation 2 if the colors are symmetric (*i.e.*, if all constraints are universally quantified over all colors).[7] In this case n is the number of colors, and m is the number of nodes in the graph. If we consider a fixed number of colors and increase the number of nodes, then equation 2 grows as:

$$\frac{(2^m + n - 1)(2^m + n - 2)\ldots(2^m)}{n!}$$

Note that the dominate term here is $\frac{2^{nm}}{n!}$. This is as one would intuitively expect since there are 2^{nm} interpretations, and $n!$ possible permutations of the colors (but also note that the actual number of distinct interpretations is slightly larger than this since some colorings do not use all available colors).

As a second example, consider a scheduling problem in which a collection of n identical devices are each described by m variables (*e.g.*, p_{ij} might mean that device i is turned on at time j). In this case we can apply equation 2 if the devices are interchangeable (*i.e.*, if all

[7] If the nodes in the graph are also symmetric then the number of distinct interpretations is further reduced.

constraints are universally quantified over all devices). If we consider m (the number of variables describing each device) to be fixed then the number of distinct interpretations grows as:

$$\frac{(n+2^m-1)(n+2^m-2)\ldots(n+1)}{2^m!}$$

Notice that 2^m can be interpreted as the number of possible descriptions of each device. Thus if we let k be 2^m then the dominate term in this equation is just $\frac{n^k}{k!}$. Thus if the symmetries are taken advantage of, the difficulty of constraint satisfaction with interchangeable devices should only grow polynomially with the number of devices (but exponentially with the number of possible descriptions of each device).

As a final example, consider a problem, such as the n-queens problem, which takes place on a chess board. There are eight symmetries of the chess board. Assume that we have one propositional variable for each square on the chess board (and some set of constraints between them which are based on the geometry of the board). One can easily compute the cycles in each symmetry of the board and plug these into equation 1 to show that there are:

$$\frac{1}{8}(2^{4k^2}+2^{2k^2+log(2)}+2^{k^2+1}+2^{3k^2+1})$$

distinct interpretations (where the chess board is of size $2k$). Note that if n is the size of the board then the dominate term is $\frac{2^{n^2}}{8}$ as expected.

5 Example: The Pigeonhole Problem

The pigeonhole problem is a well studied example for which it is known that any resolution or tableau proof of unsatisfiability must be exponentially long ([Harken 85], see also [Jeroslow & Wang 90], pp. 185-186.). The problem is the following: given that only one pigeon can fit in a hole, show that $n+1$ pigeons cannot fit in n holes. This can be written in propositional form by letting $P_{i,j}$ represent the assertion that pigeon i is in hole j. We then write "one pigeon per hole" as

$$\bigwedge_{i_1=1}^{n+1}\bigwedge_{i_2=1}^{n+1}\bigwedge_{j=1}^{n}\neg P_{i_1,j}\vee\neg P_{i_2,j}, \tag{3}$$

and "every pigeon is in some hole" as

$$\bigwedge_{i=1}^{n+1}\bigvee_{j=1}^{n}P_{i,j} \tag{4}$$

Past approaches have used extended resolution (resolution plus the ability to add new propositional variables as shorthands for disjunctions) to derive polynomial length proofs of unsatisfiability. [Buss 87] also shows that there exist polynomial length Frege proofs of the pigeonhole principle. Buss' argument proceeds by showing that there exists an extended Frege proof (involving a formalization of counting and addition), and then showing that the variables introduced by the extension rule only shorten this proof by a polynomial factor.

Symmetries allow a much more direct polynomial proof of the pigeonhole principle ([Krishnamurthy 85]). One reason the pigeonhole principle is intuitively obvious is that the holes are clearly interchangeable and the pigeons are clearly interchangeable. Thus a person simply considers one assignment of pigeons to holes, realizes that there are too many pigeons, and concludes that there is no possible assignment. Resolution and tableau based proofs are exponentially long because they effectively consider and reject *every possible mapping* of pigeons to holes.

Symmetry allows us to shrink this search back to a consideration of only a single assignment. Each disjunct in equation 4 shrinks by symmetry (using lemma 2.1) to a *single term*. One can then use unit resolution to generate a full model of the theory.

The graphs in the appendix show run times for the pigeon-hole problem with and without symmetry (using a tableau based algorithm). Symmetries were computed using a fast symmetry detection algorithm which, though incomplete, still finds a large and useful class of symmetries (the algorithm will be included in [Crawford 92]).

6 Related Work

[Freuder 91] discusses the elimination of interchangeable values in constraint satisfaction problems. Freuder is primarily concerned with cases in which two values can be interchanged without affecting the rest of the theory, but in the last section there is some speculation on *functional interchangeability* — a notion which is similar to symmetry (in fact Freuder states that symmetries are a likely source of functionally interchangeable values).

[Brown et al. 88] discuss an algorithm for backtracking search in the presence of symmetry. They assume that the symmetries of the theory are given as input, and essentially search the space of equivalence classes of interpretations of the theory. The techniques presented in section 4 should thus be applicable to predicting the size of the space searched by their algorithm. Despite taking the symmetries of the theory as input, the algorithm presented by Brown et al. still uses a graph isomorphism routine as a subroutine. The analysis in section 3 thus suggests that taking the symmetries of a theory as input may not make a significant computational difference.

[Krishnamurthy 85] discusses the idea of using symmetries to reduce the lengths of resolution proofs. He uses the "rule of symmetry" which asserts that if θ is symmetry of a theory Th, and one can show that p follows from Th, then $\theta(p)$ also follows from Th (since the proof could be repeated with each step s replaced by $\theta(s)$). One can show that lemma 2.2 follows from the rule of

symmetry. Similarly, one can derive the rule of symmetry from lemma 2.2 (thus giving a model theoretic proof of the rule of symmetry). [Krishnamurthy 85] shows how the rule of symmetry can be used to produce polynomial length proofs of some propositions (such as the pigeonhole principle) which require exponentially long proofs using resolution alone. Krishnamurthy's work does not, however, address the problem of detecting symmetries or of using them in search problems.

7 Discussion

There are several important limitations of the results presented here which should be kept in mind. First, the cases in which the use of symmetries results in an exponential speed-up are comparatively rare. More commonly symmetries provide "constant factor" improvements in run times (though in larger problems, *e.g.*, coloring problems in ten colors, these constant factors can be measured in millions). Second, the graph isomorphism problem is not known to be tractable (nevertheless in applications such as matching electrical circuits, graph isomorphism problems are regularly solved on relatively large graphs apparently without exponential explosions in run times). Third, techniques are known for solving constraint satisfaction problems without searching the complete space of possible variable values. Thus results in section 4 must be regarded as estimates of the actual effects of symmetries on search. Fourth, and finally, we have not shown how to efficiently search the space of equivalence classes of interpretations. The algorithm presented in [Brown et al. 88] is a step in this direction, but has two drawbacks — first, it requires solving graph isomorphism problems during the search, and second, it is not immediately clear how to integrate it with known constraint satisfaction techniques.

Despite these limitations, there are several possible applications of these results. First, Pólya's theorem can be used to identify problem classes, such as scheduling identical devices, in which the space of distinct interpretations grows polynomially. In such cases one can then expect to be able to find special purpose polynomial time constraint satisfaction algorithms. Similarly, in problems such as the n-queens problem, one can expect to find techniques which realize the constant factor speed-ups predicted by Pólya's theorem. Second, problems without symmetry may be approximable by symmetric problems with smaller search spaces. One can then use the solution to the symmetric problem to guide the search for the solution for the original problem. Third, the mapping given here between symmetry detection and graph isomorphism does not appreciable increase the size of the problem. Thus one can utilize known graph isomorphism algorithms to take advantage of symmetries within general purpose constraint satisfaction algorithms. This can be done either by precomputing the symmetries of the problem (as in [Brown et al. 88]) or by checking for symmetries "on the fly" (in some cases partial solutions to constraint satisfaction problems may actually display symmetries not present in the original problem, so the second approach may be more powerful than the first). Finally, reasoning by symmetry can be applied to theorem proving. [Krishnamurthy 85] has shown that symmetries can significantly reduce the lengths of certain hand proofs. The results presented here suggest that graph isomorphism techniques might be applicable to utilizing such symmetries in automatic deduction.

Acknowledgments

I would like to thank Bart Selman, David Etherington, Larry Auton, and Lewis Stiller for useful discussions of this material. I would also like to thank David McAllester, Marc Vilain, and Haym Hirsh for some very helpful comments on an earlier draft of this paper. Finally, I would like to thank Rob Schapire and Mike Kerns for their help in solving some of the recurrence relations required to derive the results in section 4.

References

[Brown et al. 88] Brown, C.A., Finkelstein, L., and Purdom, P.W., Jr. (1988). Backtrack searching in the presence of symmetry. In T. Mora (ed.), *Applied algebra, algebraic algorithms and error-correcting codes, 6th International Conference*. Springer-Verlag, pp. 99-110.

[Buss 87] Buss, S.R. (1987). Polynomial size proofs of the propositional pigeonhole principle. *The Journal of Symbolic Logic*, vol. 52, number 4, pp. 916-927.

[Cook 74] Cook, S., and Reckhow, R. (1974). On the lengths of proofs in the propositional calculus. *STOC-74*, pp. 135-148.

[Crawford 92] Crawford, J.M. (1992). A Theoretical Analysis of Reasoning By Symmetry in First-Order Logic. In preparation.

[Freuder 91] Freuder, E.G. (1991). Eliminating interchangeable values in constraint satisfaction problems. *AAAI-91*, pp. 227-233.

[Glaisher 1874] Glaisher, J.W.L. (1874). On the problem of the eight queens. *Philosophical Magazine*, series 4, vol. 48, pp. 457-467.

[Harken 85] Harken, A. (1985). The intractability of resolution. *Theoretical Computer Science*, volume 39, pp. 297-308.

[Jeroslow & Wang 90] Jeroslow, R.G. and Wang, J. (1990). Solving propositional satisfiability problems. *Annals of Mathematics and Artificial Intelligence*, volume 1, pp. 167-187.

[Krishnamurthy 85] Krishnamurthy, B. (1985) Short proofs for tricky formulas. *Acta Informatica*, volume 22, pp. 253-275.

[Luks 80] Luks, E.M. (1980). Isomorphism of bounded valence can be tested in polynomial time. *Proceedings 21st Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, Los Angeles, pp. 42-49.

[Pólya & Read 87] Pólya, G. and Read, R.C. (1987). *Combinatorial Enumeration of Groups, Graphs, and Chemical Compunds*. Springer-Verlag.

[Williamson 85] Williamson S.G. (1985). *Combinatorics for Computer Science*. Computer Science Press.

Appendix

Figure 1: Run times for pigeonhole problem without symmetry (time in milliseconds).

Figure 2: Run times for pigeonhole problem with symmetry (time in milliseconds).