Math.Comput.Sci. (2017) 11:297–303
DOI 10.1007/s11786-017-0316-2

Mathematics in Computer Science

What Does "Without Loss of Generality" Mean, and How Do We Detect It

James H. Davenport

Received: 8 December 2016 / Revised: 12 March 2017 / Accepted: 15 March 2017 / Published online: 25 April 2017

Abstract When one goes from a geometrical statement to an algebraic statement, the immediate translation is to replace every point by a pair of coordinates, if in the plane (or more as required). A statement with N points is then a statement with 2N (or 3N or more) variables, and the complexity of tools like cylindrical algebraic decomposition is doubly exponential in the number of variables. Hence one says "without loss of generality, A is at (0,0)" and so on. How might one automate this, in particular the detection that a "without loss of generality" argument is possible, or turn it into a procedure (and possibly even a formal proof)?

Keywords Symmetry · Generality · Cylindrical Algebraic Decomposition

Mathematics Subject Classification 68W30

1 Introduction

> Symmetry is at once a familiar concept (we recognize it when we see it!) and a profoundly deep mathematical subject. At its most basic, a symmetry is some transformation of an object that leaves the object (or some aspect of the object) unchanged. [11]

That quotation comes from a major survey of symmetry in purely Boolean satisfiability problems, but our setting is "Satisfiability Modulo Theories" over the real numbers, and the desire to enhance this with techniques from Computer Algebra, notably Cylindrical Algebraic Decomposition: see [1,2]. Hence we need to worry about symmetry in the underlying theory as well as in the Boolean formulation, and ask what alignment there is between symmetry in the underlying theory and in the Boolean satisfiability problem that encodes the problem. Clearly if neither have any symmetry there is nothing to discuss, and if both possess *the same* symmetry, the reasoning in [11] applies. If there is symmetry in the Boolean problem, but that is not reflected in the underlying theory, then the theory in [11] is *not* applicable. For example, if $P(a, b, c)$ is symmetric in a and b, we might think that we have to consider three cases: $P(T, T, c)$, $P(T, F, c)$ and $P(F, F, c)$, since $P(F, T, c)$ is equivalent to $P(T, F, c)$. But if P is $c \Rightarrow (a \vee b) \wedge \neg(a \wedge b)$ with c being $x = 1$, a being $x < 0$ and b being $x^2 < 4$, then truth of c gives us

J. H. Davenport (✉)
Department of Computer Science, University of Bath, Bath BA2 7AY, UK
e-mail: J.H.Davenport@bath.ac.uk

the $a = F, b = T$ case we would have discarded. This paper is concerned with the remaining case, where there is symmetry in the underlying theory that does not manifest itself directly in the Boolean formulation.

Many proofs, particularly of the more computational kind, in mathematics contain a line of the form "without loss of generality, we may assume …" (often abbreviated w.l.o.g). This is discussed in [6], who claims, we believe correctly, that this means one of two, rather different, types of argument:

Type A: non-degeneracy for example "w.l.o.g. $\alpha \neq 0$", really means[1] "$\alpha = 0$ is a special case, which you can easily see for yourself, so I am not going to bother with it here";
Type B: exploitation of symmetry as in [6]'s opening example of Schur's inequality $\forall a, b, c \in \mathbf{R}, k \in \mathbf{N}$,

$$0 \leq a^k(a-b)(a-c) + b^k(b-a)(b-c) + c^k(c-a)(c-b), \tag{1}$$

where a typical proof might begin: "Without loss of generality, let $a \leq b \leq c$".

This paper is essentially concerned with the second case, though, as we shall see, it is not possible to ignore the first, and indeed a given statement might combine both in practice.

This is a very powerful form of human reasoning. Harrison [6] asks, and substantially answers, the question of how, assuming the symmetry is stated, it can be incorporated into formal proof: here we ask an equivalent question for computation, notably in the context of Symbolic Computation and Satisfiability Checking [2]. The challenge turns out to be the *detection*, rather than the use, of the symmetry. Harrison [6] does not discuss detection: we believe that our study of detection is equally applicable to the proof context.

2 Exploitation of Symmetry—Discrete

Harrison [6] explains the example above as follows.

> If asked to spell this out in more detail, we might say something like: Since $\leq$ is a total order, the three numbers must be ordered somehow, i.e. we must have (at least) one of $a \leq b \leq c$, $a \leq c \leq b$, $b \leq a \leq c$, $b \leq c \leq a$, $c \leq a \leq b$ or $c \leq b \leq a$. But the theorem is completely symmetric between a, b and c, so each of these cases is just a version of the other with a change of variables, and we may as well just consider one of them.

He then offers two possible formalisms:

- The phrase may be an informal shorthand saying 'we should really do 6 very similar proofs here, but if we do one, all the others are exactly analogous and can be left to the reader'.
- The phrase may be asserting that 'by a general logical principle, the apparently more general case and the special WLOG case are in fact equivalent (or at least the special case implies the general one)'.

He then argues that the second interpretation is closer to the informal mathematics, and shows how to implement this as a HOL-Light theorem, more precisely

$$\begin{aligned} \vdash\ & (\forall xyz.Pxyz \Rightarrow Pyxz \wedge Pxzy) \wedge \\ & (\forall xyz.x \leq y \wedge y \leq z \Rightarrow Pxyz) \\ \Rightarrow\ & (\forall xyz.Pxyz) \end{aligned} \tag{2}$$

Note 1 *There's a subtlety here: in fact the statement is invariant under S_3, the symmetric group on $\{x, y, z\}$, but the two permutations listed, $xyz \to yxz$ and $xyz \to xzy$ (in cycle notation (x, y) and (y, z)), generate S_3.*

[1] However, it may also mean **Type C:** "$\alpha = 0$ renders the result meaningless, so we shall not consider it further". These are extremely common in geometry: see Observation 1 and [7].

Table 1 Cells satisfying $a \le b \le c$

c < 0	b < c	<u>All</u>
	b = c	a < c; a = c
c = 0	b < 0	a < b; a = b
	b = 0	a < 0; a = 0
c > 0	b < 0	<u>All</u>
	b = 0	<u>a < c</u>
	0 < b < c	<u>All</u>
	b = c	a < 0; a = 0; 0 < a < c; a = c

2.1 Does this Help SC^2?

Unfortunately (1) is not polynomial: we need to specialise k. If we feed $(1)|_{k=2}$ into Cylindrical Algebraic Decomposition (either the [5] or the [3] implementations), we get 31 cells (as we do for any even k: odd k give us 59 cells, but the conclusions are similar): the major split is on how c compares with 0: $c < 0$ then splits on how b compares with c and 0 (five possibilities, with $b = c$ splitting a into five possibilities, and $b = 0$ splitting on how a compares with c); $c > 0$ similarly, and $c = 0$ having a three-way split on b, each having a three-way split on a. Of these, the 14 listed in Table 1 satisfy $a \le b \le c$, either totally, or, where underlined, only partially. Not only is this ratio of $14/31 \approx 45\%$ disappointing compared with the naïve (not allowing for equality) 1/6 one might expect: if we split the underlined cells to get precisely the cells with $a \le b \le c$, the ratio would be $18/39 \approx 46\%$.

2.2 How Might We Detect It?

The most obvious generalisation of Note 1 is the following well-known result.

Proposition 1 *The permutations* $(1, 2, \ldots, n)$ *and* $(1, 2)$ *generate* S_n *as a group acting on* $\{1, 2, \ldots, n\}$.

Corollary 1 *Hence, if a statement* $P(x_1, x_2, \ldots, x_n)$ *is given, and* $P(x_1, x_2, \ldots, x_n)$ *is mathematically equivalent to* $P(x_2, x_1, \ldots, x_n)$ *and to* $P(x_2, \ldots, x_n, x_1)$*, it is sufficient to prove*

$$x_1 \le x_2 \wedge x_2 \le x_3 \wedge \cdots x_{n-1} \le x_n \Rightarrow P(x_1, x_2, \ldots, x_n) \tag{3}$$

Note 2 *We have said "mathematically equivalent to", rather than just "equal to", as we needed the laws of algebra (at least commutativity of addition and multiplication) to show that* (1) *was actually invariant.*

3 Exploitation of Symmetry—Continuous

> One of the most important ways in which such invariances are used in proofs is to make a convenient choice of coordinate system. [6]

If a problem is intrinsically geometric, then the precise coordinate system is irrelevant to the truth of the statement. It is this sort of symmetry that we will look for in this section.

Let us consider the following example

Theorem 1 (Simson's Theorem, [9,12]) *Let* D *be on the circumcircle of the triangle* ABC*, let* P*,* Q *and* R *be the points of* AB*,* AC *and* BC *where the line to* D *is perpendicular. Then* P*,* Q *and* R *are collinear.*

Let us consider just the first statement[2] "Let D be on the circumcircle of the triangle ABC".

[2] The rest of the theorem introduces no more free variables.

Observation 1 *Implicit in this is the statement that ABC* has *a circumcircle, i.e. that it is non-degenerate. To get as far as (4), we need to state this: see [4] for the details.*

One possible coordinatisation[3] of this is (4).

$$
\begin{aligned}
x_D{}^2 + y_D{}^2 = {} & \frac{\begin{array}{c} x_D\left(x_A{}^2 y_B - x_A{}^2 y_C - x_B{}^2 y_A + x_B{}^2 y_C + x_C{}^2 y_A - x_C{}^2 y_B + y_A{}^2 y_B \\ - y_A{}^2 y_C - y_A\, y_B{}^2 + y_A\, y_C{}^2 + y_B{}^2 y_C - y_B\, y_C{}^2\right) \end{array}}{x_A\, y_B - x_A\, y_C - x_B\, y_A + x_B\, y_C + x_C\, y_A - x_C\, y_B} \\
& + \frac{\begin{array}{c} y_D\left(-x_A\left(x_B{}^2 + y_B{}^2\right) + x_A\left(x_C{}^2 + y_C{}^2\right)\right. \\ \left. + x_A{}^2\,(x_B - x_C) + y_A{}^2\,(x_B - x_C) - x_B\left(x_C{}^2 + y_C{}^2\right) + x_C\left(x_B{}^2 + y_B{}^2\right)\right) \end{array}}{x_A\, y_B - x_A\, y_C - x_B\, y_A + x_B\, y_C + x_C\, y_A - x_C\, y_B} \\
& + \frac{1}{4}\,\frac{\left(x_A{}^2 y_B - x_A{}^2 y_C - x_B{}^2 y_A + x_B{}^2 y_C + x_C{}^2 y_A - x_C{}^2 y_B + y_A{}^2 y_B - y_A{}^2 y_C - y_A\, y_B{}^2 + y_A\, y_C{}^2 + y_B{}^2 y_C - y_B\, y_C{}^2\right)^2}{\left(x_A\, y_B - x_A\, y_C - x_B\, y_A + x_B\, y_C + x_C\, y_A - x_C\, y_B\right)^2} \\
& + \frac{1}{4}\,\frac{\left(-x_A\left(x_B{}^2 + y_B{}^2\right) + x_A\left(x_C{}^2 + y_C{}^2\right) + x_A{}^2\,(x_B - x_C) + y_A{}^2\,(x_B - x_C) - x_B\left(x_C{}^2 + y_C{}^2\right) + x_C\left(x_B{}^2 + y_B{}^2\right)\right)^2}{\left(x_A\, y_B - x_A\, y_C - x_B\, y_A + x_B\, y_C + x_C\, y_A - x_C\, y_B\right)^2} \\
& - \left(x_A - \frac{1}{2}\,\frac{\begin{array}{c} x_A{}^2 y_B - x_A{}^2 y_C - x_B{}^2 y_A + x_B{}^2 y_C + x_C{}^2 y_A - x_C{}^2 y_B \\ + y_A{}^2 y_B - y_A{}^2 y_C - y_A\, y_B{}^2 + y_A\, y_C{}^2 + y_B{}^2 y_C - y_B\, y_C{}^2 \end{array}}{x_A\, y_B - x_A\, y_C - x_B\, y_A + x_B\, y_C + x_C\, y_A - x_C\, y_B}\right)^2 \\
& - \left(y_A + \frac{1}{2}\,\frac{\begin{array}{c} -x_A\left(x_B{}^2 + y_B{}^2\right) + x_A\left(x_C{}^2 + y_C{}^2\right) + x_A{}^2\,(x_B - x_C) \\ + y_A{}^2\,(x_B - x_C) - x_B\left(x_C{}^2 + y_C{}^2\right) + x_C\left(x_B{}^2 + y_B{}^2\right) \end{array}}{x_A\, y_B - x_A\, y_C - x_B\, y_A + x_B\, y_C + x_C\, y_A - x_C\, y_B}\right)^2
\end{aligned}
\tag{4}
$$

It is relatively easy (for a computer algebra system) to verify that (4) is invariant if we replace all variables z by $z + c$. Hence it is legitimate to choose $y_A = 0$, which gives us (5).

$$
\begin{aligned}
x_D{}^2 + y_D{}^2 = {} & \frac{x_D\left(x_A{}^2 y_B - x_A{}^2 y_C + x_B{}^2 y_C - x_C{}^2 y_B + y_B{}^2 y_C - y_B\, y_C{}^2\right)}{x_A\, y_B - x_A\, y_C + x_B\, y_C - x_C\, y_B} \\
& + y_D\,\frac{\begin{array}{c} y_D - x_A\left(x_B{}^2 + y_B{}^2\right) + x_A\left(x_C{}^2 + y_C{}^2\right) + x_A{}^2\,(x_B - x_C) \\ - x_B\left(x_C{}^2 + y_C{}^2\right) + x_C\left(x_B{}^2 + y_B{}^2\right) \end{array}}{x_A\, y_B - x_A\, y_C + x_B\, y_C - x_C\, y_B} \\
& + \frac{1}{4}\,\frac{\left(x_A{}^2 y_B - x_A{}^2 y_C + x_B{}^2 y_C - x_C{}^2 y_B + y_B{}^2 y_C - y_B\, y_C{}^2\right)^2}{\left(x_A\, y_B - x_A\, y_C + x_B\, y_C - x_C\, y_B\right)^2} \\
& - \left(x_A - \frac{1}{2}\,\frac{x_A{}^2 y_B - x_A{}^2 y_C + x_B{}^2 y_C - x_C{}^2 y_B + y_B{}^2 y_C - y_B\, y_C{}^2}{x_A\, y_B - x_A\, y_C + x_B\, y_C - x_C\, y_B}\right)^2
\end{aligned}
\tag{5}
$$

Again, it is relatively easy (for a computer algebra system) to verify that (5) is invariant if we replace all variables $z \in \{x_A, x_B, x_C, x_D\}$ by $z + c'$. Hence it is legitimate to choose $x_A = 0$, which gives us (6).

$$
x_D{}^2 + y_D{}^2 = \frac{x_D\left(x_B{}^2 y_C - x_C{}^2 y_B + y_B{}^2 y_C - y_B\, y_C{}^2\right)}{x_B\, y_C - x_C\, y_B} + \frac{y_D\left(-x_B\left(x_C{}^2 + y_C{}^2\right) + x_C\left(x_B{}^2 + y_B{}^2\right)\right)}{x_B\, y_C - x_C\, y_B} \tag{6}
$$

[3] Obtained with Maple's `geometry[circumcircle]` command, *after* making the non-degeneracy assumption $x_A\, y_B - x_A\, y_C - x_B\, y_A + x_B\, y_C + x_C\, y_A - x_C\, y_B \neq 0$.

In fact, both [9,12] coordinatise with $A = (x_A, 0)$ and $B = (-x_A, 0)$, taking (implicit) advantage of the fact that the problem is invariant under translation (so we can place the midpoint of AB at $(0, 0)$) and rotation (so we can place A and B on the x-axis). Then (4) becomes the simpler (7).

$$x_D{}^2 + y_D{}^2 = \frac{y_D\left(-x_A{}^2 + x_C{}^2 + y_C{}^2\right)}{y_C} + x_A{}^2 \tag{7}$$

One further step, which [9,12] could have done, and a computer system could certainly spot, is that the equation is homogeneous, and hence we can pick, say, $x_A = 1$. However, whilst appearing to be a type B w.l.o.g., exploiting symmetry under dilation, it is also asserting $x_A \neq 0$, thus a type A (trivial special case), or even type C (degenerate special case), w.l.o.g. as well.

3.1 Does this Help SC^2?

The non-vanishing of the denominators in (4)–(7) essentially corresponds to the non-degeneracy of the triangle ABC, so it is legitimate to consider just the numerators. The resource consumptions of Cylindrical Algebraic Decomposition, computing a complete CAD of $\mathbf{R}^n$ on these are shown in Table 2.

Let us consider first the [3] timings. These show, somewhat to the author's initial surprise, that Cylindrical Algebraic Decomposition is, at least in this example, unaffected in terms of cell count by the translation w.l.o.g.s, though rotation [(7) rather than (6)] and scaling (the substitution lines) help, at least in terms of cell count.

We solved (6) with variable ordering $x \succ y \succ x_C \succ y_B \succ y_C$ (i.e. y_C is the first variable to be eliminated). The different scalings were applied to (6) after $(6)|_{x_B=1}$ showed quite large denominators, e.g. cell (1,1,1,2,1) has $-\frac{3710363}{2097152} < y_B < -\frac{7420725}{4194304}$, and hence the author hoped that rescaling would reduce this problem. The effect is in fact negligible: in $(6)|_{x_B=16}$ the same cell has $-\frac{303093}{131072} < y_C < -\frac{2424743}{1048576}$, and in $(6)|_{x_B=256}$ we have $-\frac{27504107}{1048576} < y_C < -\frac{13752053}{524288}$. As can be seen, the overall effect on memory and time of changing the scaling was for the worse.

The second set of timings were produced using the software in [5], but with no special declarations, hence effectively the projection of [8]. In several cases, this warned that the projection was not well-oriented. Since the McCallum projection is a superset of the Lazard projection, and this has been recently [10] been proved unconditionally correct, we can ignore these. We observe that detecting the rotational symmetry [(7) rather than (6)] had a much greater effect here than it did for the [3] method.

Table 2 CAD of $\mathbf{R}^n$ for numerators of (4)–(7)

Chen and Moreno Maza [3]				England et al. [5] and McCallum [8]		
Equations	Cells	Time (s)	Memory MiB	Cells	Time (s)	Memory MiB
$(7)\|_{x_A=1}$	37	0.14	11	245	1.86	108
(7)	107	0.47	26	589*	3.89	303
$(6)\|_{x_B=1}$	319	3.48	256	30803*	433.20	31460
$(6)\|_{x_B=16}$	319	3.53	290			
$(6)\|_{x_B=256}$	319	4.24	318			
(6)	591	2.29	188	36531*	807.00	55000
(5)	591	2.80	235	–	>9000	
(4)	591	4.12	341			

The timings and memory usage come from Maple's `CodeTools[Usage]`, and hence both have (up to) four significant figures.
* Warning that the input is not well-oriented

The really surprising effect was the difference between (6) and (5). As far as the author could tell, the code was still projecting when interrupted after $2\frac{1}{2}$ h: at least it had produced no warnings about orientation. This needs further investigation.

3.2 How Might We Detect It?

The question of detection comes in several forms.

1D Invariance by translation by $\mathbf{R}$ can be detected, as we did in going from (4) to (5), by checking that adding c to all variables leaves the equation (or system of equations) invariant. This will fail, of course, if there are variables other than the results of coordinatisation.

2D Having detected invariance by translation by $\mathbf{R}$, we can look for invariance by translation by $\mathbf{R}^2$ as we did in going from (5) to (6), by checking that adding c to a proper subset of the variables leaves the equation (or system of equations) invariant. Of course, the author "cheated" and translated all the x variables based on variable name, but in practice one would have to try all subsets (but not a subset and its complement) of the set of variables.

3+D Though not present in our example, we could then go on to detect invariance by translation by $\mathbf{R}^3$, and so on. As we see in the discussion of rotation, it is important to do so.

Scaling This is a consequence of homogeneity, and can easily be detected. The problem is that this is also a type A (or even C) w.l.o.g. as well as type B one, and, having chosen x_i as our dehomogenising variable, we ought in principle to consider the two cases $x_i = 1$ and $x_i = 0$. The second case, if it does not collapse, is also homogeneous in the remaining variables, so we can recurse.

2D Rotation If we know that we have 2D translation symmetry, we might hope for 2D rotational symmetry as well. Let us call the set of variables translated by c in the search for 2D symmetry the "x" variables, and its complement the set of "y" variables, and assume that there are no more "x" variables than "y" variables, which will occur if we do a breadth-first search for such a set. If the problem comes from coordinatisation of a 2D geometrical problem, there should be as many "x" as "y" variables—of course whether these correspond to the original x and y or *vice versa* is a matter of chance, but from now on we shall drop the quotes, implicitly assuming the correspondence, not that it matters.

Then the question comes: which $y_j \in \{y_1, \ldots, y_m\}$ corresponds to which $x_i \in \{x_1, \ldots, x_n\}$? Here we know of no better answer than trying all $m!/(m-n)!$ possibilities for a complete assignment σ. We then replace all pairs $(x_i, y_{\sigma(i)})$ by $(cx_i - sy_{\sigma(i)}, cy_{\sigma(i)} + sx_i)$ to practice a rotation[4] by θ with $c = \cos\theta$, $s = \sin\theta$, and apply $c^2 + s^2 = 1$. For the correct assignment in our example, it is relatively easy to demonstrate equality (in particular the result is independent of c and s), and for incorrect examples we get results that still contain c and s.

3D Rotation We have no examples of this, but the principles are the same as above. Note that, if there really is 3D symmetry, we should identify it, and then choose triples $(x_i, y_{\sigma(i)}, z_{\tau(i)})$, as assuming we have merely 2D symmetry, and rotating the x and y but not the z, will fail.

4 Conclusion

We have only considered one example so far, but intend to study others. It is a "natural" example in that it comes from 2D geometry. It would be possible to build artificial examples that had, for example, rotational symmetry but

[4] The reader may object that this is a rotation about the origin. But we have already demonstrated 2D translational invariance, so one centre is as good as another.

no translational symmetry, but, in the spirit of [2], we have started with natural problems. From the basis of this limited analysis, we draw the following provisional conclusions.

It is possible to detect certain forms of symmetry simply from the equations (though it would clearly be better to detect them before coordinatisation if at all possible). For the method of [3], detecting translational symmetry has no effect on the cell count (and a modest effect on runtime and memory), but seems to be a pre-requisite to efficient detection of rotational symmetry, which is extremely helpful. The method of [8] seems much more susceptible to the number of variables, and hence all symmetry detection and "w.l.o.g." specialisation are helpful.

Acknowledgements The author is grateful to Julien Narboux for references, and to Matthew England and the referees and Programme Committee of the SC^{2}2016 workshop for useful comments. The questions by the referees of this journal improved the exposition. This work was performed as part of the H2020-FETOPEN-2016-2017-CSA Project SC^2 (712689). The underpinning Maple worksheets can be found at [4].

References

1. Ábrahám, E.: Building bridges between symbolic computation and satisfiability checking. In: Robertz, D. (ed.) Proceedings ISSAC 2015, pp. 1–6 (2015)
2. Ábrahám, E., Becker, B., Bigatti, A., Buchberger, B., Cimatti, C., Davenport, J.H., England, M., Fontaine, P., Forrest, S., Kroening, D., Seiler, W., Sturm, T.: SC^2: satisfiability checking meets symbolic computation (project paper). Proc. CICM **2016**, 28–43 (2016)
3. Chen, C., Moreno Maza, M.: An incremental algorithm for computing cylindrical algebraic decompositions. In: Feng, R., Lee, W.-s., Sato, Y. (eds.) Computer Mathematics, pp. 199–221. Springer, Berlin (2014)
4. Davenport, J.H.: Dataset supporting 'What Does "Without Loss of Generality" Mean (And How Do We Detect It)'. http://doi.org/10.5281/zenodo.305441 (2017)
5. England, M., Wilson, D.J., Bradford, R., Davenport, J.H.: Using the Regular Chains Library to build cylindrical algebraic decompositions by projecting and lifting. Proc. ICMS **2014**, 458–465 (2014)
6. Harrison, J.: Without loss of generality. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) Proceedings Theorem Proving in Higher Order Logics: TPHOLs 2009, pp. 43–59 (2009)
7. Kutzler, B., Stifter, S.: On the application of Buchberger's algorithm to automated theorem proving. J. Symb. Comput. **2**, 389–397 (1986)
8. McCallum, S.: An improved projection operation for cylindrical algebraic decomposition. Ph.D. thesis, University of Wisconsin-Madison Computer Science (1984)
9. Mou, C.: Software library for triangular decompositions. Talk at ICMS 2016 (2016)
10. McCallum, S., Parusinski, A., Paunescu, L.: Validity proof of Lazard's method for CAD construction. https://arxiv.org/abs/1607.00264 (2016)
11. Sakallah, K.A.: Symmetry and satisfiability. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, chapter 10, pp. 289–338. IOS Press, (2009)
12. Wang, D.: GEOTHER: A geometry theorem prover. In: Proceedings International Conference on Automated Deduction, pp. 166–170 (1996)