Problems on the Generation of Finite Models

Jian Zhang[1]

Institute of Software, Academia Sinica
P.O.Box 8718, Beijing 100080, P.R.China

Abstract. Recently, the subject of model generation has received much attention. By *model generation* we mean the automated generation of finite models of a given set of logical formulas. In this note, we present some problems on the generation of finite models. The purpose is two-fold: (1) to offer some test problems for model generation programs; and (2) to show the potential applications of such programs. Some of the problems are easy, some are hard and even open. We also give a new result in combinatory logic, which says that the fragment $\{B, N_1\}$ does not possess the strong fixed point property.

1 Introduction

Traditionally, the major task of automated deduction (or automated reasoning) systems is proving theorems. But there are also other tasks which rely on logical reasoning. One of them is the generation of finite models, i.e. finding a finite model satisfying a given set of formulas. A trivial example is, given a set of 3 or 5 axioms for group theory, produce the multiplication tables of a 6-element group. Model generation is complementary to theorem proving: instead of proving conjectures, it produces counterexamples refuting conjectures. It is also important in its own right, e.g. in some branches of mathematics, we would like to construct an object having some desired properties. Finite model generation can be regarded as constraint satisfaction in finite domains, with emphasis on logical reasoning instead of numerical computation. And model generators can be thought of as a special class of automated reasoning programs.

Since model generation is a new subject, we feel it necessary to collect some test problems for such programs. The following problems are selected, which is partly due to their connection with Automated Deduction, and partly because they have been attacked by existing programs. Three programs have been used: FINDER (Finite Domain Enumerator [8]), Sato (an implementation of the Davis-Putnam procedure for checking satisfiability in propositional logic [14]), and FALCON (a tool for Finite ALgebra CONstruction, previously known as Mod/E [15]). In some cases, we shall give the execution times (on a Sparcstation 2) of

[1]Supported in part by the Natural Science Foundation of China.

the programs. Throughout this note, we adopt the convention that a model of size n has the set {0, 1, 2, ..., n-1} as its domain.

2 The Problems

2.1 Ternary Boolean Algebra

A ternary Boolean algebra is a nonempty set with a ternary operator f and a unary operator g, satisfying the following 5 axioms:

$(T1)$ $f(f(v,w,x),y,f(v,w,z)) = f(v,w,f(x,y,z))$

$(T2)$ $f(y,x,x) = x$ $\quad$ $(T4)$ $f(x,x,y) = x$

$(T3)$ $f(x,y,g(y)) = x$ $\quad$ $(T5)$ $f(g(y),y,x) = x$

The axioms (T4) and (T5) can be shown to depend on the remaining axioms. But the first three are independent of each other. With assistance from a theorem prover, Winker [9] established the independence of (T2) by finding a model of (T1), (T3), (T4) and (T5) which falsifies (T2). It is easy for a model generator to find the models automatically. One such model, which is different from Winker's, was given in [15].

2.2 Combinatory Logic

Combinatory logic can be defined as an equational system satisfying the combinators S and K with $a(a(a(S,x),y),z) = a(a(x,z),a(y,z))$ and $a(a(K,x),y) = x$. There are also other combinators, such as

$a(a(a(B,x),y),z) = a(x,a(y,z))$ $\quad$ $a(a(L,x),y) = a(x,a(y,y))$

$a(a(a(N_1,x),y),z) = a(a(a(x,y),y),z)$ $\quad$ $a(M,x) = a(x,x)$

In a fragment of combinatory logic (defined by a set of combinators), the strong fixed point property holds if there exists a combinator y such that for all combinators x, $a(y,x) = a(x,a(y,x))$. Using a theorem prover, Wos and McCune [13] have obtained many results on the existence (or nonexistence) of fixed point combinators in certain fragments of combinatory logic.

To prove the nonexistence of fixed point combinators, one may find a counterexample, i.e. a model in which the fixed point property does not hold. Here we give a 5-element model of {B, N_1} which falsifies the fixed point property, answering an open problem proposed by Wos [13].

a	0	1	2	3	4
0	0	0	2	3	4
1	0	0	2	3	4
2	3	3	3	3	3
3	3	3	3	3	3
4	4	4	4	4	4

(In the model, $B = 0$, $N_1 = 1$.) FALCON-2 finds the model in about 1 second. We have also searched for finite counterexamples for the following fragments: {B, L}; {B, M}; and {B, S}. We fail to find the models when the size is restricted to be less than 6. It seems not easy to generate larger models.

2.3 Single Axioms for Group Theory

A *single axiom* for a theory is a formula from which the entire theory can be derived. With a theorem prover, one may prove that a formula is a single axiom by deriving from it the known axioms of the theory. To search for single axioms for a given theory, one usually generates a large number of candidate formulas. Many of the candidates are not single axioms. To show this, one may use a model generator to construct a model of the candidate which falsifies some known axiom (or any known theorem). In fact, FINDER has been used McCune [6] in the search for single axioms in group theory.

In [6] (and some other papers), several single axioms for groups are given. The question is: are they the simplest, or, is it possible that some of their proper instances are still single axioms? For an example, let us see the single axiom

$$(G1) \quad f(x, i(f(y, f(f(f(z, i(z)), i(f(u, y))), x)))) = u$$

One of its instances is (taking $x = u$)

$$(G2) \quad f(u, i(f(y, f(f(f(z, i(z)), i(f(u, y))), u)))) = u$$

There exists a 2-element model satisfying (G2) but falsifying (G1): $f(0,0) = f(0,1) = f(1,1) = 0$, $f(1,0) = 1$, $i(0) = i(1) = 0$. So (G2) is not a single axiom.

2.4 Equivalential Calculus

Equivalential Calculus (EC) has the following 3 axioms:

$$(E1) \quad P(e(x, x))$$
$$(E2) \quad P(e(e(x, y), e(y, x)))$$
$$(E3) \quad P(e(e(x, y), e(e(y, z), e(x, z))))$$

and the inference rule *condensed detachment* (*CD*): $P(e(x, y))$ & $P(x) \rightarrow P(y)$. It was shown [12] that (E1) is dependent on (E2) and (E3). But is it possible that one of the remaining axiom is dependent on the other? The answer is no. In fact, there is a 2-element model of (E3) and (CD) which falsifies (E2), and there is a 3-element model of (E2) and (CD) which falsifies (E3).

EC also has some short single axioms. Similarly, model generators can play some role in detecting those formulas which are not single axioms. For two testing problems, we list the following two formulas which are known not to be single axioms of EC, because (E1) is not derivable from either of them [7] [11].

$$(XBB) \quad P(e(x, e(e(e(x, e(y, z)), y), z)))$$
$$(XAK) \quad P(e(x, e(e(e(e(y, z), x), z), y)))$$

There is a 4-element model of (XBB) which does not satisify (E1).

2.5 Quasigroups and Latin Squares

A *quasigroup* is a set on which a binary operation $\circ$ is defined, such that, for every pair (a, b) of elements, each of the equations $a \circ x = b$ and $y \circ a = b$ has a unique solution. A quasigroup is *idempotent* iff $x \circ x = x$ holds for every element x. Mathematicians are interested in the existence of quasigroups having some additional property. Model generators can assist in such studies. In fact, various programs have solved some previously open questions [2]. To show

the performance of model generators on the quasigroup problems, we note that SATO takes about 2 minutes to conclude that there is no idempotent quasigroup of order 12 satisfying the identity $((y \circ x) \circ y) \circ y = x$. In contrast to the existence problems, one may consider some enumeration problems, e.g. enumerating the number of Latin squares of order 8 [4]. (A Latin square of order n corresponds to the multiplication table of an n-element quasigroup.)

2.6 The High School Identities

The following set (HSI) of identities are true in the set **N** of positive integers, which one learns in high school:

$$\begin{array}{llll} x+y = y+x & & x+(y+z) = (x+y)+z \\ x*y = y*x & & x*(y*z) = (x*y)*z \\ x*1 = x & & x*(y+z) = (x*y)+(x*z) \\ 1^x = 1 & & x^1 = x & & (x^y)^z = x^{y*z} \\ x^{y+z} = x^y * x^z & & & & (x*y)^z = x^z * y^z \end{array}$$

Tarski's High School Problem is: whether the above eleven identities serve as a basis for all the identities of **N**. It was answered in the negative in 1980 by Wilkie, who showed that the following identity holds in **N**, but cannot be derived from HSI: $(P^x + Q^x)^y * (R^y + S^y)^x = (P^y + Q^y)^x * (R^x + S^x)^y$ where $P = 1+x$, $Q = 1+x+x*x$, $R = 1+x*x*x$, *and* $S = 1+x*x+x*x*x*x$. In fact, there are finite models of HSI which do not satisfy Wilkie's identity. Such models are called *G-algebras*. An open question is: *what's the size of the smallest G-algebra*? It is known that this size is at least 7 and at most 15 [1]. The question should be a hard one for current model generation programs. The size of the model is not small, and there are 3 function symbols.

3 Concluding Remarks

We have described some problems of finite model generation. Some of them are related to previous research on automated theorem proving, while others are new. Some instances of the problems are easy, others are hard and even open. We note that model generation programs can be used to obtain independence results, to show that a formula is *not* a theorem, and to determine the existence of certain finite objects in mathematics. Combined with theorem provers, model generators may offer us a better understanding of a theory.

Due to the limit of space, many other problems are not included, such as the n-queens problem, the jobs puzzle and the zebra puzzle. We also note that, models are not necessarily finite. Among problems on the generation of infinite models, we list the following open one: does there exist an infinite Robbins algebra which is not Boolean? It is known that every finite Robbins algebra is Boolean [10]. In addition to the three programs mentioned, there are other programs which can generate finite models, such as SATCHMO [5], MGTP [3], DDPP [14] and various Constraint Logic Programming systems. We hope that more model generation programs will be developed, and more problems solved.

References

[1] Burris, S., and Lee, S., "Tarski's high school identities," *The Amer. Math. Monthly* 100 (1993) 231–236.

[2] Fujita, M. et al, "Automatic generation of some results in finite algebra," *Proc. 13th IJCAI* (1993) 52–57.

[3] Hasegawa, R. et al, "MGTP: A parallel theorem prover based on lazy model generation," *Proc. 11th CADE, LNAI* 607 (1992) 776–780.

[4] Kolesova, G. et al, "On the number of 8×8 Latin squares," *J. Combinatorial Theory* A54 (1990) 143–148.

[5] Manthey, R., and Bry, F., "SATCHMO: A theorem prover implemented in Prolog," *Proc. 9th CADE, LNCS* 310 (1988) 415–434.

[6] McCune, W., "Single axioms for groups and Abelian groups with various operations," *J. Automated Reasoning* 10 (1993) 1–13.

[7] Peterson, J. G., "The possible shortest single axioms for EC-tautologies," Report No. 105, Dept. of Mathematics, Univ. of Auckland (1977).

[8] Slaney, J., "FINDER: Finite domain enumerator. Version 3.0 notes and guide," Australian National University, (1993).

[9] Winker, S., "Generation and verification of finite models and counterexamples using an automated theorem prover answering two open questions," *J. ACM* 29 (1982) 273–284.

[10] Winker, S., "Robbins algebra: Conditions that make a near-Boolean algebra Boolean," *J. Automated Reasoning* 6 (1990) 465–489.

[11] Wos, L. et al, "A new use of an automated reasoning assistant: Open questions in equivalential calculus and the study of infinite domains," *Artificial Intelligence* 22 (1984) 303–356.

[12] Wos, L. "Meeting the challenge of fifty years of logic," *J. Automated Reasoning* 6 (1990) 213–232.

[13] Wos, L., "The kernel strategy and its use for the study of combinatory logic," *J. Automated Reasoning* 10 (1993) 287–343.

[14] Zhang, H., and Stickel, M., "Implementing the Davis-Putnam method by tries," submitted to AAAI-94.

[15] Zhang, J., "Search for models of equational theories," *Proc. 3rd Int'l Conf. for Young Computer Scientists (ICYCS-93)*, Beijing (1993) 2.60–63.