Vol.10 No.3 J. of Comput. Sci. & Technol. May 1995

Automatic Construction of Finite Algebras

Zhang Jian (张 健)

Institute of Software, The Chinese Academy of Sciences, Beijing 100080

Received December 31, 1993; revised July 21, 1994.

Abstract

This paper deals with *model generation* for equational theories, i.e., automatically generating (finite) models of a given set of (logical) equations. Our method of finite model generation and a tool for automatic construction of finite algebras is described. Some examples are given to show the applications of our program. We argue that, the combination of model generators and theorem provers enables us to get a better understanding of logical theories. A brief comparison between our tool and other similar tools is also presented.

Keywords: Finite model generation, equational theory, backtracking search.

1 Introduction

Research on automated theorem proving has been continued for more than 30 years. So far, the chief method for proving theorems relies on showing the *unsatisfiability* of a set of logical formulas. But, how can we show the *satisfiability* of a given set of formulas? How can we show that some formula is *not* a theorem of some given theory? One approach to doing this.is to produce a model of a set of formulas. In this paper, we study the automatic generation of *finite* models of a given set of *equational* axioms. Equational logic is very important in mathematics, and it is also closely related to computer science.

The paper is organized as follows. In Section 2, we describe some concepts and notations which will be used later. Then, in Sections 3 and 4, we present an algorithm for generating finite algebras and an implementation of this algorithm. Section 4 also contains several examples. Finally, we discuss the significance and difficulty of model generation, and make some comparison between our program and other similar programs.

2 Preliminaries

We assume the reader knows the basic concepts of mathematical logic. In the following, we shall introduce some additional concepts and notations, with emphasis on equational logic. An *equation* takes the form of $l = r$, where l and r are terms. In every equation, all variables are assumed to be universally quantified. An equation is called *ground*, if it does not contain any variables. A set of equations can serve as axioms of an equational theory. (Throughout this paper, we assume that each of the equational theories can be axiomatized by a finite number of equations.) For example, the following five axioms axiomatize group theory:

$$\begin{array}{ll}
(G1) & f(e,x) = x \\
(G2) & f(x,e) = x \\
(G3) & f(i(x),x) = e \\
(G4) & f(x,i(x)) = e \\
(G5) & f(f(x,y),z) = f(x,f(y,z))
\end{array}$$

Here f and i stand for the multiplication and inverse operations, respectively, and e represents the identity element of the group.

A finite *model* of an equtional theory is a finite algebra in which all the equational axioms hold. Without loss of generality, we assume that a model of size n has the following elements: 0, 1, 2, ..., $n-1$. The multiplication table of a unary (binary, resp.) function will be represented by a 1-dimensional (2-dimensional, resp.) array. The elements of the arrays are function values. For example, a 3-element model of the group theory (i.e., a group of size 3) is represented in the following way:

f	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

i	0	1	2
	0	2	1

$e: 0$

That means, the group has 3 elements: 0, 1, and 2. The identity element is 0. And,

$$\begin{array}{lll}
i(0) = 0, & i(1) = 2, & i(2) = 1; \\
f(0,0) = 0, & f(0,1) = 1, & f(0,2) = 2, \\
f(1,0) = 1, & f(1,1) = 2, & f(1,2) = 0, \\
f(2,0) = 2, & f(2,1) = 0, & f(2,2) = 1.
\end{array}$$

As the example shows, a finite model can be represented by a set of simple ground equations, which we call *assignments*. Each assignment takes the form of $f(e_1, e_2, \ldots, e_m) = v$, where f is a function symbol whose arity is m, and $e_1, e_2, \ldots, e_m$ and v are elements of the model (i.e., integers between 0 and $n-1$). In the group theory example, the ground equations $f(1,1) = 2$, $i(2) = 1$, and many others, are assignments.

3 Finding Finite Models of Equational Theories

A finite model of an equational theory is usually represented by the multiplication tables of the functions in this theory, or equivalently, by a finite number of assignments, as we have shown in the previous section. Since the number of possible assignments is also finite, we can, in principle, generate an algebra by *exhaustive search*.

Now we present the basic algorithm for generating finite algebras. We assume that the size of the algebra is n. An element of the multiplication tables is called a *cell*. Each cell can have a value which is an integer between 0 and $n-1$, or an undefined value. Initially, the values of all cells are undefined. Our goal is to assign a suitable value to each cell. The algorithm is roughly as follows:

```
while there are cells whose values are undefined
{
   choose a cell whose value is undefined;
   assign an integer between 0 and n-1 to this cell;
            /* now we get a new assignment */
   derive additional assignments from this new assignment;
   if contradiction occurs, change the value of the cell
      or backtracks;
   otherwise, continue the above process.
}
```

This procedure can terminate with success (i.e., all the cells are assigned suitable values, and a model is found) or with failure (i.e., there is no model). A slight modification of the algorithm can make it find *all* (or a fixed number of) the models of the given theory.

In typical applications, the number of possible assignments is quite large. So, many heuristics and techniques should be adopted when implementing the above algorithm and when solving problems with the implemented program. Some of our experiences have been discribed in [1]. A more significant heuristic is related to the *isomorphism* concept in algebra. We know that two different multiplication tables may represent the same algebra. For the group theory example, the identity element may be chosen as 0 or as 1 or as 2. If we enumerate all the 3 possibilities, there will be much redundancy, and the efficiency of the search algorithm will be affected. To avoid this bad behavior, we can choose — without loss of generality — the smallest element (i.e., 0) as the identity.

The isomorphism is caused by the many ways of mapping the elements of one finite model to different integers. And we adopt a technique to deal with this problem, which we call the *least number heuristic*. Now we explain its basic idea. For simplicity, suppose we have a theory in which there is only one binary function symbol f. In an arbitrary model of size n, there are n different elements. Initially, no element is given an integer. During the search, we map the elements to integers one by one. First take an arbitrary element and assign the integer 0 to it. Now consider the value of $f(0,0)$. It can be 0 or another value which is different from 0. Without loss of generality, we assume that $f(0,0)$ can only take two values: 0 or 1. Then we consider the values of $f(0,1)$, $f(1,0)$, and $f(1,1)$. In general, at some stage of the search, suppose we have used m integers ($m < n$), which are the smallest nonnegative integers. Then, when deciding the possible values of a cell whose value is not yet known, we may consider only $m+1$ cases. The first m cases correspond to the values 0, 1, ..., $m-1$. In the last case, the cell takes a new value, which can be assumed to be m (the smallest one).

4 A Tool for Constructing Finite Algebras

In this section, we describe a tool for constructing finite algebras, which we call FALCON (previously called Mod/E [1]). It can be used to find one model (or all models) of a given set of equational axioms. The input specification of this program consists of the following four parts:

> (*size*)
> { *function_name*(*arity*) } . . .
> [*equation*] . . .
> ⟨*property*⟩

Among these four parts, the first is a positive integer which is greater than one, the second and third are the same as in equational logic. The fourth part is optional. It is a first-order logical formula which does not contain predicates other than equality. And it allows the user to specify some additional properties that the model should satisfy. In the *property* formula, the symbols '!', '&', '|', 'A' and 'E' represent *not, and, or, for all* and *exists*, respectively. In the following, we shall give some examples to illustrate the use of our program.

Example 1. Boolean algebra. There are three function symbols: n, s and p, which stand for 'negation', 'sum', 'product', respectively. Suppose we want to find a Boolean algebra of order eight. The specification and the model are as follows.

```
INPUT                                 OUTPUT

    (8)                                   n | 0 1 2 3 4 5 6 7
                                          ---+----------------
    { n(1) }                                | 1 0 3 2 5 4 7 6
    { s(2) }
    { p(2) }                              s | 0 1 2 3 4 5 6 7
                                          ---+----------------
    [ s(x,y) = s(y,x) ]                   0 | 0 1 2 3 4 5 6 7
    [ p(x,y) = p(y,x) ]                   1 | 1 1 1 1 1 1 1 1
    [ s(x,0) = x ]                        2 | 2 1 2 1 1 2 2 1
    [ s(0,x) = x ]                        3 | 3 1 1 3 4 7 4 7
    [ p(x,1) = x ]                        4 | 4 1 1 4 4 1 4 1
    [ p(1,x) = x ]                        5 | 5 1 2 7 1 5 2 7
    [ s(x,n(x)) = 1 ]                     6 | 6 1 2 4 4 2 6 1
    [ s(n(x),x) = 1 ]                     7 | 7 1 1 7 1 7 1 7
    [ p(x,n(x)) = 0 ]
    [ p(n(x),x) = 0 ]                     p | 0 1 2 3 4 5 6 7
    [ s(p(x,y),p(x,z)) =                  ---+----------------
               p(x,s(y,z)) ]              0 | 0 0 0 0 0 0 0 0
    [ p(s(x,y),s(x,z)) =                  1 | 0 1 2 3 4 5 6 7
               s(x,p(y,z)) ]              2 | 0 2 2 0 6 5 6 5
                                          3 | 0 3 0 3 3 0 0 3
                                          4 | 0 4 6 3 4 0 6 3
                                          5 | 0 5 5 0 0 5 0 5
                                          6 | 0 6 6 0 6 0 6 0
                                          7 | 0 7 5 3 3 5 0 7
```

The time taken to produce the 8-element Boolean algebra is about 3.5 seconds on a Sparcstation 2. If we change the size from 8 to 7, FALCON terminates in about 4 seconds, without printing any model. It tells us that there is no Boolean algebra of order 7.

Example 2. Group theory. Given a set of 3 or 5 (equational) axioms for groups, FALCON can find a finite group. Here we shall not give the multiplication tables of ordinary groups. Instead, we show that FALCON can produce counterexamples, and can be used in the search for single axioms of group theory. A *single axiom* for a theory is a formula from which the entire theory can be derived. To show that an equation (*eq*) is a single axiom for group theory, one may use a theorem prover to prove that each of the known axioms (G1-5) is derivable from (*eq*). (See, for example, [2].) In case that (*eq*) is not a single axiom, a model generator can be used to produce a counterexample, i.e., a model of (*eq*) in which some of the axioms (G1-5) do not hold. Neumann[3] claimed that the following equation is a single axiom for group theory:

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

where i means inverse and f means multiplication. FALCON can generate all the 2-element models of (GN), which are listed below:

i	0	1
	0	1

f	0	1
0	0	1
1	1	0

i	0	1
	1	0

f	0	1
0	0	1
1	1	0

i	0	1
	0	1

f	0	1
0	1	0
1	0	1

i	0	1
	1	0

f	0	1
0	1	0
1	0	1

It is well-known that each group has an identity element e such that $i(e) = e$. Yet neither of the last 2 models possesses this property, since in each of them, $i(0)=1$ and $i(1)=0$. So, the equation (GN) is not a single axiom for group theory[1].

Example 3. 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 B, I, L, M, N_1 and Q, which are defined by the following equations:

$$\begin{aligned}
a(a(a(B,x),y),z) &= a(x,a(y,z)) \\
a(I,x) &= x \\
a(a(L,x),y) &= a(x,a(y,y)) \\
a(M,x) &= a(x,x) \\
a(a(a(N_1,x),y),z) &= a(a(a(x,y),y),z) \\
a(a(a(Q,x),y),z) &= a(y,a(x,z))
\end{aligned}$$

For a given set of combinators, the (*strong*) *fixed point property* holds iff there exists a combinator y such that for all combinators x, $a(y,x) = a(x,a(y,x))$. Wos and McCune[5,6] have studied the existence problem of fixed point combinators and obtained many results. However, they still left some questions open[6,7]. Some of these questions are listed below:

[1]We remark that FINDER[4] was the first program to find such a counterexample.

1. Is there a finite model of {B, L} in which the fixed point property does not hold?
2. Is there a finite model of {Q, L} in which the fixed point property does not hold?
3. Does {B, N1} satisfy the fixed point property?
4. Does {B, M} satisfy the fixed point property?

(Although McCune and Wos[5] have established that neither {B, L} nor {Q, L} has the fixed point property, the corresponding model might be infinite. So the first 2 questions are still meaningful.) Among the four questions, the second and third can be answered with FALCON.

```
INPUT
            (4)
            { a(2) }
            [ a(a(a(0,x),y),z) = a(y,a(x,z)) ]   ## Q is 0
            [ a(1,x),y) = a(x,a(y,y)) ]          ## L is 1
            < Ay.Ex.!(a(y,x)=a(x,a(y,x))) >

OUTPUT
             a | 0 1 2 3
            ---+---------
             0 | 0 0 2 2
             1 | 0 0 2 2
             2 | 1 1 3 3
             3 | 1 1 3 3

INPUT

    (5)
    { a(2) }
    [ a(a(a(0,x),y),z) = a(x,a(y,z)) ]         ## B is 0
    [ a(a(a(1,x),y),z) = a(a(a(x,y),y),z) ]    ## N1 is 1
    < Ay.Ex.!(a(y,x)=a(x,a(y,x))) >

OUTPUT

     a | 0 1 2 3 4
    ---+-----------
     0 | 0 0 2 3 4
     1 | 0 0 2 3 4
     2 | 2 2 2 2 2
     3 | 2 2 2 2 2
     4 | 4 4 4 4 4
```

For the third question, if B and N_1 are allowed to be equal, then we may choose $B = N_1 = 0$. In this case, there is a 4-element model. The first and the fourth questions are still open.

5 Discussion

It is in general undecidable to determine if there is a nontrivial finite model of a given (equational) theory. (A model is nontrivial if it contains more than one element.) When the size of the model is fixed, in principle, one can find the models by exhaustive search. But the complexity of this approach is very high. So automated model generation is very difficult. Yet it is also very important. The existence of a model ensures the consistency of a theory. And, model generators can do something which cannot be done with theorem provers, as shown by the examples in the previous section. In some sense, the combination of a (finite) model generator and a refutation-based theorem prover can *simulate* a decision procedure for a subset of first-order predicate logic.

Despite the difficulty of model generation, we can still do something in this direction, as shown by us in this paper, and by other researchers[4,8,9]. Compared with other similar programs like FINDER[4] and MGTP[10], our program FALCON has built-in isomorphism rejecting mechanism, i.e., the least number heuristic. It works dynamically and is quite powerful. In fact, FALCON is much faster than FINDER on some equational problems like the generation of finite Boolean algebras. Our method for rejecting isomorphism is also very general. It does not depend on the particular problems. In addition to the aforementioned tools, the decision procedure for propositional logic may also be used to produce finite models of some first-order theories. Hantao Zhang's program SATO[11] and Mark Stickel's program DDPP are two such examples. While they exhibit good performance on some problems (e.g., quasigroup identities[9]), it seems difficult for them to deal with axioms which contain deeply nested terms.

Acknowledgement The author would like to thank Prof. C.S. Tang for his support and encouragement.

References

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

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

[3] Neumann B H. Yet another single law for groups. *Illinois J. of Math.*, 1986, 30(2): 295-300.

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

[5] McCune W, Wos L. The absence and the presence of fixed point combinators. *Theoretical Computer Science*, 1991, 87(2): 221-228.

[6] Wos L. The kernel strategy and its use for the study of combinatory logic. *J. Automated Reasoning*, 1993, 10(3): 287-343.

[7] Wos L. Automated reasoning and Bledsoe's dream for the field. In *Automated Reasoning: Essays in Honor of Woody Bledsoe*, Boyer R S (ed), Kluwer Academic Publishers, 1991.

[8] Fujita M, Slaney J, Bennett F. Automatic generation of some results in finite algebra. In *Proc. 13th Int'l Joint Conf. on Artif. Intel.(IJCAI-93)*, 1993, pp.52-57.

[9] Slaney J, Fujita M, Stickel M. Automated reasoning and exhaustive search: Quasigroup existence problems. *Computers and Mathematics with Applications*, to appear.

[10] Hasegawa R, Koshimura M, Fujita H. MGTP: A parallel theorem prover based on lazy model generation. In *Proc. 11th Int'l Conf. on Automated Deduction (CADE-11)*, *LNAI* 607, Springer, 1992, 776-780.

[11] Zhang H. Sato: A decision procedure for propositional logic. *AAR Newsletter*, No. 22, March 1993.

Zhang Jian received his B.S. and M.E. degrees in computer science from University of Science and Technology of China, in 1988 and 1991, respectively. He completed and defensed his Ph.D. thesis in the spring of 1994. Now he is a researcher at the Institute of Software, The Chinese Academy of Science, Beijing. His research interests include automated reasoning, formal specifications, equational logic and temporal logic. Most recently, he has been working on the automatic generation of finite models of given logical theories.

He is a member of the Association for Automated Reasoning.