System Description: MCS: Model-Based Conjecture Searching[⋆]

Jian Zhang

Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Beijing 100080, CHINA
zj@ox.ios.ac.cn

Abstract. An enumerative approach to conjecture formulation is presented. Its basic steps include the generation of terms and formulas, and testing the formulas in a set of models. The main procedure and some experimental results are described.

1 Introduction

Automated theorem provers are used to prove conjectured theorems. But where do the conjectures (theorems) come from? In most cases, they are provided by human users. However, a user might miss some interesting conjectures. It is also well known that the addition of a few key lemmas may make automated theorem proving much easier. The same is true for finite model searching [3]. We think that it can be beneficial to generate some simple formulas systematically and then test them for theoremhood.

About 10 years ago, Wos [4] listed "theorem finding" as a basic research problem in automated reasoning. What he had in mind is adopting some criteria that will help us select interesting theorems from those deduced by a theorem prover. The subject of this short paper is to find some simple conjectures which are *likely* to be theorems. They are tested in a set of small finite models of the underlying theory. If they pass the test, they become interesting conjectures.

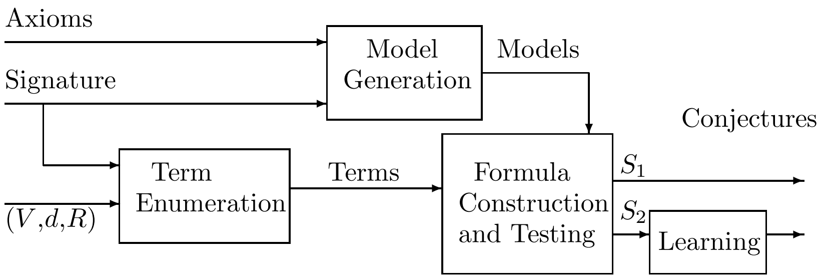
2 The Approach

Our approach is essentially generate-and-test. For a given theory which consists of a signature and a set of axioms, we need to do the following:

1. Generate a set of finite models.
2. Generate a set of closed well-formed formulas.
3. Test the truth of each formula in the models. Depending on the results, we divide the set of formulas into 3 subsets: S_0 (containing formulas which are false in every model), S_1 (containing formulas which are true in every model) and S_2 (containing the other formulas).

⋆ Partially supported by the Natural Science Foundation of China (No. 69703014).

4. Formulas in S_0 are discarded. Each formula in S_1 is a conjecture. As for formulas in S_2, we may discover possible relationships between them using machine learning techniques.
5. (Optional) Try to prove the conjectures using a theorem prover or to refute them using a model generator.



The overall process can be described by the above diagram. In the process, we can use existing provers like OTTER to prove conjectures, and use existing finite model generators (such as FINDER, SEM or MACE) to find the models and counterexamples. In addition, we need tools for enumerating terms, generating formulas and so on. In what follows, we shall give some details about them.

Obviously, for nontrivial theories, there can be too many formulas (even restricted to some length). We choose to generate some simple terms first, and then construct formulas of certain forms. To produce terms, we implemented a procedure called *STerms()* [5]. It has 3 parameters: V (a set of variables), d (a positive integer) and R (a set of rewrite rules).

The terms generated by the procedure will contain no variables other than those listed in V, and they are different from each other up to variable renaming[1]. The number of function symbol occurrences in each term will not exceed d. In addition, if R is not empty, the terms will be irreducible with respect to R. Currently our tool is not able to perform AC rewriting. When a theory has AC functions, we use McCune's equational prover EQP to reduce the number of generated terms. (Any other AC rewriting tool could also be used.)

Given a set of terms, the formulas are constructed in the following way:

```
for each pair of terms t1 and t2 {
  find the variables appearing in the terms;
  formulate different combinations of quantifiers;
  for each combination PREN
    output the formula "PREN (t1 = t2)" ;
}
```

[1] This choice is arguable. On one hand, it prevents the generation of redundant conjectures such as $\forall x\,[f(x,x) = x]$ and $\forall y\,[f(y,y) = y]$. On the other hand, we will not be able to find the commutative law, since $f(x,y)$ and $f(y,x)$ are not generated simultaneously. However, the user can add any interesting terms manually.

As an option, one can choose to generate equations only (in which all variables are universally quantified). Otherwise, we may get too many theorems, some of which are not so interesting. For example, suppose we have the theorem $\forall x \forall y\, [\phi(x,y) = \varphi(x,y)]$, then we also have $\exists x \exists y\, [\phi(x,y) = \varphi(x,y)]$, and $\forall x \exists y\, [\phi(x,y) = \varphi(x,y)]$, and so on.

Step 3 is realized in a straightforward way. For every (closed) formula generated in Step 2, we check its truth in each model.

For the formulas which are true in some models but false in the others (i.e., those in the subset S_2), we may find some relationships between them through inductive learning. For example, which formula implies another formula. We implemented a method which is similar to the least-commitment search algorithm (Section 18.5 of [1]). Its functionality is to find a minimal set of formulas whose conjunction implies a given formula. (There may be more than one minimal sets.)

The tools are implemented in C (on SPARCstations). They communicate through files. For example, when we study group theory, we work with the files `Group1.term`, `Group1.mod` and so on.

3 Experiments

Example 1. Quasigroups. The QG5 theory [3] has the following 4 axioms:

$$\begin{array}{ll} f(x,y) = f(x,z) \rightarrow y = z \qquad & f(x,x) = x \\ f(x,z) = f(y,z) \rightarrow x = y \qquad & f(f(f(y,x),y),y) = x \end{array}$$

We first generate three models, of sizes 5, 7 and 8, respectively. Then we enumerate some terms. Suppose we choose $V = \{x, y\}$, $d = 3$, and R consists of the rule $f(x,x) \Rightarrow x$. Then there will be 23 terms, as given below.

```
x                    f(x,y)               f(x,f(x,y))
f(x,f(y,x))          f(f(x,y),x)          f(f(x,y),y)
f(x,f(x,f(x,y)))     f(x,f(x,f(y,x)))     f(x,f(y,f(x,y)))
f(x,f(y,f(y,x)))     f(x,f(f(x,y),x))     f(x,f(f(x,y),y))
f(x,f(f(y,x),x))     f(x,f(f(y,x),y))     f(f(x,y),f(y,x))
f(f(x,f(x,y)),x)     f(f(x,f(x,y)),y)     f(f(x,f(y,x)),x)
f(f(x,f(y,x)),y)     f(f(f(x,y),x),x)     f(f(f(x,y),x),y)
f(f(f(x,y),y),x)     f(f(f(x,y),y),y)
```

Only 12 pairs of them are equal in all the 3 models. All of the 12 conjectures turn out to be theorems. They can be proved by any state-of-the-art prover. The simplest of these theorems include the last axiom and the following three:

$$f(f(y,f(x,y)),y) = x \qquad f(y,f(f(x,y),y)) = x \qquad f(x,f(y,x)) = f(f(x,y),x)$$

The first two are generally believed to be helpful in solving the QG5 problem[3].

We have also experimented with other theories, like group theory, rings, boolean algebras and so on. Some results are given in Table 1. The meanings of the parameters are as follows:

Theory	Models		Terms				Conjectures			
	m	n	v	d	r	t	q	k	p	f
Group	6	5	2	2	0	23	253	16	16	0
GrpH	6	5	2	2	17	29	406	5	3	2
Ring	6	7	2	2	4	42	861	23	19	4
Ring	6	8	3	2	9	39	741	8	8	0
Bool	8	3	1	2	10	17	136	37	37	0
Bool	8	3	2	2	19	24	276	23	23	0

Table 1. Experiments with Some Abstract Algebras

m:	the maximum size of the models;
n:	the number of models;
v:	the size of V;
r:	the size of R;
t:	the number of generated terms;
q:	the number of equational formulas;
k:	the number of formulated conjectures;
p:	the number of true conjectures;
f:	the number of false conjectures.

We have, $q = t(t-1)/2$ and $k = p+f$. GrpH refers to group theory extended with commutators. For this theory, one conjecture (i.e., $h(f(x,y),x) = h(g(x),y)$) is falsified by a 12-element model; another one (i.e., $h(h(x,y),x) = h(x,h(y,x))$) is also false, but we failed to find a finite counterexample (of size up to 20).

All conjectures are equational in the first example and in Table 1. The next example is about implicational conjectures in group theory.

Example 2. Group theory. Suppose we want to know under what conditions a group is Abelian. We set $V = \{x, y\}$ and $d = 2$. Using a set of 11 groups (whose sizes are between 4 and 8), we found more than 10 equations, each of which implies commutativity. Some of the equations are given below.

$$x^2 = e; \quad x^{-1} = x; \quad x^3 = x.$$

All conjectures are proved to be theorems. But in an earlier attempt, we used 5 groups (of sizes 4, 5 and 6). And one conjecture is wrong:

$$\forall x\,(x^3 = x^{-1}) \;\rightarrow\; \forall x \forall y\,(xy = yx).$$

4 Concluding Remarks

Conjecture formulation as outlined above combines term rewriting, theorem proving, model generation and machine learning. As we said in the Introduction, it helps to produce the targets of theorem proving (i.e., the theorems to be proved), and it helps to prove certain theorems or to find certain models quickly. This may also be considered as an interesting application of finite models.

In many cases, key lemmas and good theorems are very simple formulas. We believe that the model-based conjecture searching process is very useful, especially when studying a new theory or when trying to prove a new theorem. It is also feasible, since we have good model generators now. Usually we do not need too many models, and the models do not have to be very big, to distinguish between "good" (i.e., theorems) and "bad" (i.e., false conjectures).

Of course, it is inevitable that a few conjectures turn out to be false. But the number of such conjectures can be reduced, if more models are used. And some of them may lead to interesting discoveries. In the case of modes, for example, even though a conjecture like $\forall x \forall y\, [\, f(x, y) = x \,]$ is not valid, it can hold under certain conditions [6].

Useful lemmas can also be generated by a resolution style, forward chaining program. DELTA [2], for example, generates unit clauses by applying UR-resolution, and adds them to the original formula. Such tools generate conclusions (rather than conjectures) automatically, many of which are quite complicated. But most resolution-based programs do not deal with quantified formulas directly. Our approach may be considered as complementary to forward chaining.

Combinatorial explosion is a major obstacle, although some measures have been taken (such as exploiting rewrite rules). In the future, we shall study techniques for the automatic discovery and selection of more complex, more interesting conjectures.

Acknowledgment

One referee helped to determine the theoremhood of some conjectures.

References

1. Russell, S. and Norvig, P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, Englewood Cliffs, New Jersey, USA, 1995.
2. Schumann, J.M.Ph., "DELTA – A bottom-up preprocessor for top-down theorem provers," *Proc. CADE-12*, 774–777, 1994.
3. Slaney, J. *et al.* "Automated reasoning and exhaustive search: Quasigroup existence problems," *Computers and Math. with Applications* 29(2): 115–132, 1995.
4. Wos, L., *Automated Reasoning: 33 Basic Research Problems,* Prentice Hall, Englewood Cliffs, New Jersey, USA, 1988.
5. Zhang, J., *The Generation and Applications of Finite Models*, PhD thesis, Institute of Software, Chinese Academy of Sciences, Beijing, China, 1994.
6. Zhang, J., "Constructing finite algebras with FALCON," *J. Automated Reasoning* 17(1): 1–22, 1996.