Computer Search for Counterexamples to Wilkie's Identity*

Jian Zhang

Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, P.R. China
zj@ios.ac.cn

Abstract. Tarski raised the High School Problem, i.e., whether a set of 11 identities (denoted by HSI) serves as a basis for all the identities which hold for the natural numbers. It was answered by Wilkie in the negative, who gave an identity which holds for the natural numbers but cannot be derived from HSI. This paper describes some computer searching efforts which try to find a small model of HSI rejecting Wilkie's identity. The experimental results show that such a model has at least 11 elements. Some experiences are reported, and some issues are discussed.

1 Introduction

One learns in high school that the following identities are true in the set **N** of positive integers:

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

The above set of identities is denoted by HSI. Tarski's *High School Problem* is: does HSI serve as a basis for all the identities of **N**? In other words, can any identity which holds for the natural numbers be derived from HSI using equational reasoning? For more details about this problem, see [2].

In 1981, Wilkie gave the following identity, denoted by $W(x, y)$, and showed that it holds in **N**, but cannot be derived from HSI:

* Supported by the National Science Fund for Distinguished Young Scholars of China (grant No. 60125207).

R. Nieuwenhuis (Ed.): CADE 2005, LNAI 3632, pp. 441–451, 2005.

$$(P^x + Q^x)^y * (R^y + S^y)^x = (P^y + Q^y)^x * (R^x + S^x)^y$$

where $P = 1+x$, $Q = 1+x+x*x$, $R = 1+x*x*x$, and $S = 1+x*x+x*x*x*x$.

Wilkie's original proof was purely syntactic. Alternatively we can try to build a finite model of HSI in which $W(x, y)$ does not hold. In fact, Gurevič [4] constructed such a model which has 59 elements. Models like this are called *G-algebras* later in [1]. In this paper, such a model will be called a Gurevič-Burris Algebra (GBA). A GBA serves as a counterexample showing that the Wilkie identity does not follow from HSI. In the sequel, we use the constants a and b to denote a pair of elements in a GBA such that $W(a, b)$ does not hold.

A question naturally arises: what is the smallest GBA? It is proved in [1] that the size of any GBA is at least 7 and that a 15-element GBA exists. In [5], the lower bound is increased to 8, and the upper bound is decreased to 14. More recently, Burris and Yeats found a 12-element GBA [2].

All the above lower bounds are proved mathematically. Alternatively, we may also try to find a small GBA using computers. This sounds quite interesting, and also feasible, given that there have been significant advances in computer hardware and search algorithms. Over the past 10 years, we have attempted to search for a small GBA for several times. But it is a bit disappointing that no GBA was found. However, since the search algorithm is complete, the experimental results still tell us something (i.e., GBAs of certain sizes do not exist). This paper gives a brief summary of our experiments.

2 Search Programs

In the early 1990's, we developed a general-purpose search program for finding finite models, called FALCON [11,13]. Typically its input consists of a set of equations E and a positive integer n. The input may also include an inequation of the form `s != t`, where `s` and `t` are terms. The output of FALCON is an n-element model of E, if there is such a model. Usually we assume that the domain of the model is $D_n = \{0, 1, \ldots, n-1\}$. Such a model is an interpretation of the function symbols in D_n, such that every equation (inequation) holds.

SEM [14] can be regarded as a successor of FALCON. It is more efficient, and it can accept non-unit first-order clauses. To use SEM to find a 4-element GBA, we may give it the following input file:

```
4.

s(x,y) = s(y,x).
s(x,s(y,z)) = s(s(x,y),z).
p(x,1) = x.
p(x,y) = p(y,x).
p(x,p(y,z)) = p(p(x,y),z).
p(x,s(y,z)) = s(p(x,y),p(x,z)).
e(1,x) = 1.
e(x,1) = x.
```

```
e(x,s(y,z)) = p(e(x,y),e(x,z)).
e(p(x,y),z) = p(e(x,z),e(y,z)).
e(e(x,y),z) = e(x,p(y,z)).

c = p(a,a).
P = s(1,a).
Q = s(P,c).
R = s(1,p(a,c)).
S = s(s(1,c),p(c,c)).
p(e(s(e(P,a),e(Q,a)),b),e(s(e(R,b),e(S,b)),a)) !=
p(e(s(e(P,b),e(Q,b)),a),e(s(e(R,a),e(S,a)),b)).
```

Here the first line denotes the size of the model, and the rest of the file gives the formulas (in clausal form). The function symbols `s`, `p` and `e` denote summation, product and exponentiation, respectively. The variable `x`, `y` and `z` are assumed to be universally quantified.

In our recent experiments, we also used Mace4 [7], another general-purpose model searching program.

All of these programs work by backtracking search, which is exhaustive in nature. If such a program terminates without finding a model, it means that there is no model of the given size (assuming that the program is correct).

Let us briefly explain the search process. Suppose $n = 4$. Then we need to find suitable values for all the constants and these terms:

$$s(0,0),\ s(0,1),\ s(0,2),\ s(0,3),\ s(1,0),\ \ldots,\ p(0,0),\ p(0,1),\ \ldots,\ e(3,3).$$

Each of them can take a value from the domain D_4, i.e., { 0, 1, 2, 3 }. Some of the terms should get certain values, e.g., $e(1,0) = e(1,1) = 1$. But for other terms, we can only "guess" their values. For instance, $s(0,0)$ may take either of the four values. So we assign each value to $s(0,0)$ and check if there is any contradiction. Similarly for other terms and constants.

3 Useful Mathematical Results

To show that a GBA has at least seven elements, Burris and Lee [1] first establish some lemmas. They are basically properties of elements x and y in an HSI-algebra which guarantee $W(x,y)$ holds in the algebra. Some of the lemmas (and theorems) are also helpful to computer search.

Following Burris and Lee (page 154 of [1]), we use "HSI $\vdash \Sigma \to W(x,y)$" to denote that

$$\text{HSI} \ \vdash \ \forall x \forall y \, [\Sigma \to W(x,y)].$$

Here Σ is an identity. We also use "$u \mid v$" to denote that $\exists w \, (v = u * w)$.

It is shown by Lee (Lemma 8.7 of [1]) that HSI $\vdash x \mid y \ \to \ W(x,y)$. Thus in a GBA, $a \mid b$ does not hold. In other words, for any x,

$$(L1) \qquad b \neq a * x.$$

It is also shown by Lee (Lemma 8.13 of [1]) that, if Σ is one of the following conditions:

$$\begin{array}{l} P \mid Q \\ Q \mid P \\ R \mid S \\ S \mid R \end{array}$$

then we have $\quad \mathrm{HSI} \vdash \Sigma \to W(x, y)$.

Similarly, we can conclude from the first 4 conditions that, for any x,

$$\begin{array}{ll} (L2) & Q \neq P * x \\ (L3) & P \neq Q * x \\ (L4) & S \neq R * x \\ (L5) & R \neq S * x \end{array}$$

As in [1,2], we call the following elements of an HSI-algebra *integers*:

$$1, \quad 2 \;=\; 1+1, \quad 3 \;=\; 2+1, \quad \ldots$$

In [1], it is proved that a GBA must have at least 3 integers. Moreover, if $W(x, y)$ fails at a, b, then a and b should be distinct non-integer elements. Thus in a GBA, the elements $1, 2, 3, a, b$ are different from each other.

In addition, Lemma 8.20 of [1] tells us that, if Σ is one of the following conditions:

$$\begin{array}{l} 1 + x = 1 \\ 2 + x = 1 \\ x + x = 1 \\ x * x = 1 \\ 1 + x * x = 1 \\ x * x * x = 1 \\ 1 + x = x \\ 2 + x = x \\ x + x = x \\ x * x = x \\ 1 + x * x = x \\ 2 + x = 1 + x \\ x * x = 1 + x \\ x * x * x = 1 + x \\ x * x = 2 + x \\ x * x = x + x \\ 1 + x * x = x * x \end{array}$$

then we have HSI $\vdash \Sigma \to W(x, y)$. Thus in a GBA,

$$
\begin{array}{ll}
(M01) & 1 + a \neq 1 \\
(M02) & 2 + a \neq 1 \\
(M03) & a + a \neq 1 \\
(M04) & a * a \neq 1 \\
(M05) & 1 + a * a \neq 1 \\
(M06) & a * a * a \neq 1 \\
(M07) & 1 + a \neq a \\
(M08) & 2 + a \neq a \\
(M09) & a + a \neq a \\
(M10) & a * a \neq a \\
(M11) & 1 + a * a \neq a \\
(M12) & 2 + a \neq 1 + a \\
(M13) & a * a \neq 1 + a \\
(M14) & a * a * a \neq 1 + a \\
(M15) & a * a \neq 2 + a \\
(M16) & a * a \neq a + a \\
(M17) & 1 + a * a \neq a * a
\end{array}
$$

Finally, let us quote Jackson (Lemma 1 in [5]):

> Let $y = n + n_1 x + n_2 x^2 + \ldots + n_m x^m$, where $n, n_1, n_2, \ldots, n_m$ are integers. Then HSI $\vdash W(x, y)$.

Thus, to search for a GBA, we may add the following lemmas:

$$
\begin{array}{ll}
(J11) & b \neq 1 + 1 * a \\
(J12) & b \neq 1 + 2 * a \\
(J21) & b \neq 2 + 1 * a \\
\ldots &
\end{array}
$$

In mathematical terms, b can not be in the core generated by a.

4 Search for 7-Element and 8-Element GBAs

4.1 Size 7 by FALCON

In 1993 and 1994, while developing the finite algebra search program FALCON, the author chose the HSI problem as an "exercise" for the program. Since the program deals mainly with equations, we started from some partial models of HSI, and tried to extend each of them to a GBA.

For the 7-element case, there are 3 partial models initially:

(1) The integers are 0, 1, 2, with $s(0,0) = 1$, $s(0,1) = s(1,0) = 2$; and $a = 3$, $b = 4$.
(2) The integers are 0, 1, 2, 3, with $s(0,0) = 1$, $s(0,1) = s(1,0) = 2$, $s(1,1) = s(0,2) = s(2,0) = 3$; and $a = 4$, $b = 5$.

(3) The integers are 0, 1, 2, 3, 4, with $s(0,0) = 1$, $s(0,1) = s(1,0) = 2$, $s(0,2) = s(1,1) = s(2,0) = 3$, and $s(0,3) = s(1,2) = s(2,1) = s(3,0) = 4$; and $a = 5$, $b = 6$.

Several lemmas in $(M01)$–$(M17)$ are added to the input of FALCON. In addition, we established our own lemmas. When a partial model satisfies certain conditions, it will not be extended, because these conditions guarantee that Wilkie's identity holds. Some heuristics are also used in the search. For example, when choosing an unknown, we not only consider the number of possible values for it, but also consider the number of new assignments which may be generated when the unknown is assigned a value. More details are given in [11].

The search for a 7-element GBA was completed on SUN SPARCstations. It lasted for several days.

4.2 Size 8 by SEM

The search for an 8-element GBA was completed on HP workstations running UNIX, using SEM [14]. It also lasted for several days. As in the case of size 7, we started from many partial models and tried to extend each of them using SEM. The initial partial models were obtained by hand. The search was performed in 1995, during the author's visit to the University of Iowa, U.S.A.

5 Recent Experimental Results

In August 2004, we completed the search for 9-element and 10-element GBAs. No model was found. The searches were mainly performed on a desktop (Dell Optiplex GX270: Pentium 4, 2.8 GHz CPU, 2G memory) running RedHat Linux. Other similar machines were also used when searching for an 11-element GBA. We used both SEM [14] and Mace4 [7] (mace4-2004-C).

5.1 Problem Formulation

In addition to the lemmas $(M01)$–$(M17)$, we also added the lemmas $(L1)$, $(L2)$, $(L3)$, $(L4)$, $(L5)$. The basic formulation is given in the Appendix. But in individual searches, we used more formulas representing different partial models.

5.2 The Case of Size 9

This case is divided further into several subcases:

(1) There are 3 integers: 1, 2, 3.
(2) There are 4 integers: 1, 2, 3, 4.
  (a) s(4,1) = 4.
  (b) s(4,1) = 3.
  (c) ...
(3) There are 5 integers: 1, 2, 3, 4, 5.
(4) There are 6 integers: 1, 2, 3, 4, 5, 6.
(5) There are 7 integers: 1, 2, 3, 4, 5, 6, 7.

5.3 The Case of Size 10

The "search tree" is roughly like the following:

(1) There are 3 integers: 1, 2, 3.
(2) There are 4 integers: 1, 2, 3, 4.
  (a) s(4,1) = 4.
    i. s(0,0) = 2.
    ii. s(0,0) = 6.
  (b) s(4,1) = 3.
  (c) ...
(3) There are 5 integers: 1, 2, 3, 4, 5.
(4) There are 6 integers: 1, 2, 3, 4, 5, 6.
(5) There are 7 integers: 1, 2, 3, 4, 5, 6, 7.
(6) There are 8 integers: 1, 2, 3, 4, 5, 6, 7, 8.

5.4 The Case of Size 11

The search for an 11-element counterexample has not been completed yet. So far we have only eliminated some subcases. Our conclusion is that, if a GBA of size 11 exists, it should have 3 or 4 integers.

This case is much more difficult than the search for smaller models. For example, when the size is 10, the subcase of 6 integers takes SEM a little more than 10 minutes; but when the size is 11, the subcase of 6 integers takes SEM more than 150 hours.

As previously, we examine many partial models and try to extend each of them. The longest completed single run of Mace4 lasted for about 11 days. For some subcases, Mace4 did not complete the search after running for two weeks, and the process was killed.

6 Some Experiences

In addition to SEM and Mace4, we have tried other programs such as Paradox [3] and Gandalf [10]. A problem with Gandalf is that it starts the search from 1-element model to 2-element model and so on. There is no convenient way to ask it to search for 9-element model only, for example.

The GBA problem is highly "first-order". It seems that SAT-based tools are not so advantageous as on other problems. The running times of MACE2 [6] are not satisfactory. This may be due to the inefficiency of its internal SAT solver. The performance of Paradox (version 1.0-casc) is similar to that of SEM.

The problem is also highly "equational", since most of the (ground) clauses are equations/identities. We found that it is better to turn off the negative propagation rules in SEM and Mace4, as shown in Table 1. Informally speaking, such rules try to deduce from negated equations or try to generate negated equations, while ordinary (positive) propagation rules deduce equations from equations.

Table 1. Negative Propagation

size	program	neg_prop	time
7	SEM	Yes	0.24
7	SEM	No	0.18
8	SEM	Yes	7.96
8	SEM	No	5.32
7	Mace4	Yes	0.64
7	Mace4	No	0.39
8	Mace4	Yes	25.14
8	Mace4	No	11.41

Table 2. Using Lemmas

size	program	lemma	time
7	SEM	Yes	0.24
7	SEM	No	19.36
7	Mace4	Yes	0.64
7	Mace4	No	43.76

The lemmas (esp. short lemmas) are quite useful in general. Table 2 compares the performances of SEM and Mace4, when the input has or does not have the lemmas $(L1)$–$(L5)$. In both cases, the lemmas $(M01)$–$(M17)$ are included in the input file, and negative propagation is not used.

In Table 1 and Table 2, "size" denotes the size of the model. The running times ("time") are given in seconds. The data were obtained on a Dell Optiplex GX270 (Pentium 4, 2.8 GHz CPU, 2G memory, RedHat Linux).

From Table 2 we see that the five lemmas can greatly reduce the search time. We briefly described the search process at the end of Sec. 2. During the search, some terms get values, while others do not. Let us look at lemma $(L1)$ which says, for any x, $b \neq a * x$. Usually we can assume that $a = 0$ and the integers are $1, 2, 3, \ldots$ With the above lemma, if we know the value of b, we can safely exclude this value from consideration when trying to find a value for $p(0, 0)$. Similarly for $p(0, 1)$, $p(0, 2)$ and so on. Thus the number of choices is reduced, and the search time is reduced too. We note that, about ten years ago, Slaney *et al.* [9] also used some extra constraints when solving the quasigroup problems.

Although the lemmas are generally helpful, there are also some exceptions. For example, consider the subsubcase in 11-element GBA search where there are 7 integers and $7 + 1 = 7$. If we add the Jackson lemmas $(J11, \ldots, J77)$, the running time of Mace4 is 257 seconds. But without these lemmas, the running time is 144 seconds.

Electricity outage occurred once (in a weekend). Fortunately, SEM did not run for too long before it was interrupted. We think it wise to set a time limit on each run. The '`-t`' and '`-f`' options of SEM can be helpful. When the time limit is reached, SEM saves the current path of the search tree in a file called "`_UF`". From this file, we can restart the search later. We can also see the (approximately

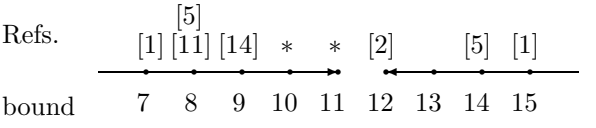
maximum) depth of the search tree. The deepest search path in our `_UF` files has 61 branches/decisions (i.e., assignments to cells).

7 Concluding Remarks

Search for a small GBA is an interesting problem for mathematicians. In this paper, we summarize some computer search results, which show that the smallest counterexample to Wilkie's identity has at least 11 elements.

Of course, this conclusion is not proved mathematically. It is possible that the programs have some bugs, or the user (myself) made some errors. Like Slaney [8], we believe that it is valuable to double check the results using other programs. This may increase their reliability.

The following picture shows some recent attempts to increase the lower bound and decrease the upper bound on the size of the smallest GBA:



In the picture, related papers and documents are given above the line. For example, it was established in [14] that the lower bound is 9 (because there is no 8-element model). The "∗" denotes this paper. When the lower bound and the upper bound meet, the problem will be solved.

In addition to its implications for mathematics, GBA search should also stimulate research in automated reasoning. It can be a challenging problem for first-order model searching programs, SAT solvers and more general constraint solvers. It was suggested as a benchmark problem more than 10 years ago [12]. Due to the advancement in hardware, improvements on algorithms and data structures, as well as useful mathematical results, we can now easily solve problem instances that were very difficult in the early 1990's. We hope that they will become even easier in the near future.

There are still some issues which deserve further investigation. First of all, are there more "efficient" specifications of the problem? Of course, this depends on which tool you are using. But it appears that certain short lemmas are always helpful. Secondly, can we prove the correctness of the search results (rather than the correctness of the search programs, which is very difficult)? Finally, we should pay more attention to the "engineering" aspects of model searching, when the problem is very difficult.

Acknowledgements

The author is very grateful to the anonymous referees for their critical comments and constructive suggestions.

References

1. S. Burris and S. Lee, Small models of the high school identities, *Int'l J. of Algebra and Computation* 2: 139–178, 1992.
2. S. Burris and K. Yeats, The saga of the high school identities, *Algebra Universalis*, to appear.
3. K. Claessen and N. Sörensson, New techniques that improve MACE-style finite model finding. In: *Model Computation – Principles, Algorithms, Applications*, CADE-19 Workshop W4, Miami, Florida, USA, 2003.
4. R. Gurevič, Equational theory of positive numbers with exponentiation. *Proc. Amer. Math. Soc.* 94: 135–141, 1985.
5. M.G. Jackson, A note on HSI-algebras and counterexamples to Wilkie's identity, *Algebra Universalis* 36: 528–535, 1996.
6. W. McCune, MACE 2.0 reference manual and guide, Technical Memorandum ANL/MCS-TM-249, Argonne National Laboratory, Argonne, IL, USA, May 2001.
7. W. McCune, *Mace4 Reference Manual and Guide*, Technical Memorandum No. 264, Argonne National Laboratory, Argonne, IL, USA, 2003.
8. J.K. Slaney, The crisis in finite mathematics: Automated reasoning as cause and cure, *Proc. 12th Int'l Conf. on Automated Deduction (CADE-12)*, 1–13, 1994.
9. J. Slaney, M. Fujita and M. Stickel, Automated reasoning and exhaustive search: Quasigroup existence problems, *Computers and Mathematics with Applications* 29(2): 115–132, 1995.
10. T. Tammet, Finite model building: improvements and comparisons. In: *Model Computation – Principles, Algorithms, Applications*, CADE-19 Workshop W4, Miami, Florida, USA, 2003.
11. J. Zhang, *The Generation and Applications of Finite Models,* PhD thesis, Institute of Software, Chinese Academy of Sciences, Beijing, 1994.
12. J. Zhang, Problems on the generation of finite models, *Proc. of the 12th Int'l Conf. on Automated Deduction* (*CADE-12*), LNAI Vol. 814, 753–757, 1994.
13. J. Zhang, Constructing finite algebras with FALCON, *J. Automated Reasoning* 17(1): 1–22, 1996.
14. J. Zhang and H. Zhang, SEM: a System for Enumerating Models. *Proc. of the 14th Int'l Joint Conf. on Artif. Intel.* (*IJCAI-95*), 298–303, 1995.

Appendix

The SEM input for a 7-element GBA search can be like the following:

```
7.

s(x,y) = s(y,x).
s(x,s(y,z)) = s(s(x,y),z).
p(x,1) = x.
p(x,y) = p(y,x).
p(x,p(y,z)) = p(p(x,y),z).
p(x,s(y,z)) = s(p(x,y),p(x,z)).
e(1,x) = 1.
e(x,1) = x.
```

```
e(x,s(y,z)) = p(e(x,y),e(x,z)).
e(p(x,y),z) = p(e(x,z),e(y,z)).
e(e(x,y),z) = e(x,p(y,z)).
c = p(0,0).
P = s(1,0).
Q = s(P,c).
R = s(1,p(0,c)).
S = s(s(1,c),p(c,c)).
p(e(s(e(P,0),e(Q,0)),b),e(s(e(R,b),e(S,b)),0)) !=
p(e(s(e(P,b),e(Q,b)),0),e(s(e(R,0),e(S,0)),b)).

2 = s(1,1).
b != 0.
b != 1.
b != 2.
b != p(0,x).
P != p(Q,x).
Q != p(P,x).
R != p(S,x).
S != p(R,x).
s(1,0) != 1.
s(2,0) != 1.
s(0,0) != 1.
     c != 1.
s(1,c) != 1.
p(c,0) != 1.
s(1,0) != 0.
s(2,0) != 0.
s(0,0) != 0.
     c != 0.
s(1,c) != 0.
s(2,0) != s(1,0).
     c != s(1,0).
p(c,0) != s(1,0).
     c != s(2,0).
     c != s(0,0).
s(1,c) != c.
```

Here we assume that a is the element 0. Given the above input, SEM and Mace4 can complete the search very quickly when the size is 7 or 8. See Table 1.