A comparison of decision procedures in Presburger arithmetic*

Predrag Janičić **Ian Green** **Alan Bundy**
University of Edinburgh

Abstract

It is part of the tradition and folklore of automated reasoning that the intractability of Cooper's decision procedure for Presburger integer arithmetic makes is too expensive for practical use. More than 25 years of work has resulted in numerous approximate procedures via rational arithmetic, all of which are incomplete and restricted to the quantifier-free fragment. In this paper we report on an experiment which strongly questions this tradition. We measured the performance of procedures due to Hodes, Cooper (and heuristic variants thereof which detect counterexamples), across a corpus of 10 000 randomly generated quantifier-free Presburger formulae. The results are startling: a variant of Cooper's procedure outperforms Hodes' procedure on both valid and invalid formulae, and is fast enough for practical use. These results contradict much perceived wisdom that decision procedures for integer arithmetic are too expensive to use in practice.

1 Introduction

A *decision procedure* for some theory is an algorithm which for every formula tells whether it is valid or not. The role of decision procedures is critical in many areas, including theorem proving. As Boyer and Moore wrote [1988, §1]

> It is generally agreed that when practical theorem provers are finally available they will contain both heuristic components and many decision procedures.

Indeed, even (generally) inefficient decision procedures could reduce the search space of heuristic components of a prover and increase its abilities: a decision procedure can both close a branch in a proof, and reject non-theorems. Decision procedures can also have a important role in other areas such as geometry and type checking, for example.

A core part of automatic theorem proving involves reasoning with integer and natural numbers. Since the whole of integer arithmetic is undecidable, we are forced to look for 'useful', decidable sub-theories; there is a trade-off between usefulness and the complexity of associated decision procedures. In this paper, we take Presburger arithmetic: it is useful and there are a number of decision procedures.

In this paper we want to compare two decision procedures (and some simple variations thereof): that due to Hodes for Presburger rational arithmetic and that due to Cooper, for Presburger integer arithmetic. It is part of the tradition and folklore of automated reasoning that Cooper's decision procedure for Presburger integer arithmetic is too expensive to be of practical use. More than 25 years of work has resulted in numerous approximate procedures via rational arithmetic, all of which are incomplete and restricted to the quantifier-free fragment. It is known that the (worst-case) time complexity of Cooper's procedure is $2^{2^{2^n}}$ in the size of the formula,[1] and moreover, this is much worse than Hodes' procedure. Boyer and Moore state [1988, §3]

> ... integer decision procedures are quite complicated compared to the many well-known decision procedures for linear inequalities over the rationals [...]. Therefore, following the tradition in program verification, we adopted a rational-based procedure...

*The first author is supported by The British Scholarship Trust. The other authors are supported in part by grants EPSRC GR/L/11724, and British Council ROM/889/95/70. Authors' address: Department of Artificial Intelligence, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, Scotland. Email: `predragj@dai.ed.ac.uk`, {`I.Green`, `A.Bundy`}`@ed.ac.uk`. Please address correspondence, requests for programs and data etc. to the second author.

[1] Shostak [1977] attributes this result to Oppen.

It is this 'tradition' of work in the rationals [Bledsoe, 1975; Shostak, 1977; 1979; Boyer and Moore, 1988] that we question in this paper.

Here we report on an experimental comparison of decision procedures on 10 000 randomly generated formulae, in order to explain this tradition to some extent. However, the results are very surprising—broadly, they show that a simple variant of Cooper's procedure *outperforms* Hodes' procedure on our sample corpus!

These results cast some doubt on the perceived wisdom in the automated reasoning community that full decision procedures for integer arithmetic are too expensive to use in practice.

Overview of paper. §2.1 defines Presburger arithmetic, procedures and notation. §3 describes the test corpus and the experiments we made on it; §4 shows the results. §6 and §7 discusses further work and draws conclusions.

2 Background

2.1 Presburger arithmetic

Presburger arithmetic is (roughly speaking) a theory built up from the constant 0, variables, binary $+$, unary s, relations $<$, $>$, $=$, $\leq$, $\geq$ and the standard connectives and quantifiers of first-order predicate calculus. The notions of term, atomic formula and formula are formally defined in the usual way (a grammar is given in figure 1 for the quantifier-free part). In *Presburger integer arithmetic* (PIA), variables range over the integers. It was Presburger who first showed that PIA is decidable [1930]. *Presburger rational arithmetic* (PRA) is defined analogously and is also decidable [Kreisel and Krivine, 1967].

The restriction of Presburger integer arithmetic to Peano numbers (i.e., to non-negative integers) we call *Presburger natural arithmetic* (PNA).[2]

We write $\models_T f$ ($\not\models_T f$) to mean f is valid (invalid) in theory T. A *decision procedure* for theory T is a total function d from formulae to the set $\{\texttt{yes}, \texttt{no}\}$, having for any f the properties of soundness $d(f) = \texttt{yes}$ implies $T \models f$, and completeness, $T \models f$ implies $d(f) = \texttt{yes}$. An *incomplete decision procedure* is sound but not complete. We say the formula is *true* by the decision procedure if it returns 'yes' otherwise it is *false*.

2.2 Related work

Some decision procedures for Presburger arithmetic are based on the idea of quantifier elimination described by Kreisel and Krivine [1967]—these are Hodes' procedure for Presburger rational arithmetic [1971] and Cooper's procedure for Presburger integer arithmetic [1972]. There is also the Sup-Inf family of procedures due to Bledsoe [1975] and latterly improved by Shostak [1977; 1979].

The rational-based procedures are an attempt to overcome the complexity of integer-based procedures. It can be easily seen that there are formulae true in rational arithmetic and false in natural arithmetic and vice versa. For instance, $\exists x. 2x = 3$ is valid over the rationals, but not over the naturals. Also, $\forall x. x \leq 1 \vee x \geq 2$ is valid over the naturals, but not over the rationals. Therefore, we cannot use a decision procedure for one of these two theories in the other, not even as an incomplete decision procedure.

However, if some universally quantified PNA formula is true by Hodes' procedure (i.e., taking it to be a formula of PRA), then it must be valid in PNA. That is, $\models_{PRA} s$ implies $\models_{PNA} s$, for the universally quantified formula s. The reverse implication does not hold, and so Hodes' procedure is not a decision procedure for PNA.

This idea of applying decision procedures for rationals to the integer case is at the heart of Bledsoe's Sup-Inf method [1975], and can be seen as the start of the tradition of incomplete decision procedures. The tradition continued with Shostak's improved Sup-Inf [1977; 1979]; he showed it could decide invalid formulae, and so was indeed a decision procedure. However, the class of formulae for which Shostak's Sup-Inf decides has not been characterized syntactically: it is not a decision procedure for universally quantified PIA, but for some "semantically characterized" fragment.

Boyer and Moore too followed this track [1988], although they reverted back to Hodes' procedure rather than using Sup-Inf. Their choice was unsurprising in some sense, since the Nqthm logic is quantifier-free, so the restriction for soundness is vacuous. Somewhat curiously, Boyer and Moore conclude in that same paper that efficiency of the DP is largely irrelevant in the wider setting of a heuristic prover: in that case why not use Cooper's procedure, and have a complete procedure to boot?[3]

Remark There is a question as to what degree *negative* results from a decision procedure can be used in a heuristic theorem prover. Some systems, such as *Clam* [Bundy *et al.*, 1990], can use this information, for example, in controlling generalization, and other non-equivalence preserving heuristics. This is undoubtedly

[2] For the same theory and some related theories there are a few different terms used. For instance, Hodes calls Presburger rational arithmetic a *theory EAR* ("the elementary theory of addition on the reals"). Boyer and Moore [1988] describe a universally quantified fragment of Presburger rational arithmetic as *linear arithmetic* (although in fact they work over the integers); that same theory sometimes goes by the name *Bledsoe real arithmetic*.

[3] We emphasize that the Nqthm decision procedure is stronger than Hodes' procedure since it also has heuristics to deal with non-Presburger formulae by calling the inductive part of the prover.

true of other theorem provers.

Ancedotal evidence abounds in the literature; we are not aware of any work quantitative experimental comparision of decision procedures, whether in an attempt to decide between procedures or otherwise.

3 Experiments

3.1 Generating Presburger formulae

We randomly generated a corpus of 10 000 formulae of Presburger arithmetic. This was done using the grammar shown in figure 1 to generate quantifier-free formulae containing free variables (taken from a set of five symbols). Each rule was chosen with a probability given in the right-hand column.

Rule	*Probability*
$\langle\text{formula}\rangle := \langle\text{atomic formula}\rangle$	0.75
$\langle\text{formula}\rangle := \neg\ \langle\text{formula}\rangle$	0.1
$\langle\text{formula}\rangle := \langle\text{formula}\rangle \vee \langle\text{formula}\rangle$	0.05
$\langle\text{formula}\rangle := \langle\text{formula}\rangle \wedge \langle\text{formula}\rangle$	0.05
$\langle\text{formula}\rangle := \langle\text{formula}\rangle \Rightarrow \langle\text{formula}\rangle$	0.05
$\langle\text{atomic formula}\rangle := \langle\text{term}\rangle = \langle\text{term}\rangle$	0.2
$\langle\text{atomic formula}\rangle := \langle\text{term}\rangle < \langle\text{term}\rangle$	0.2
$\langle\text{atomic formula}\rangle := \langle\text{term}\rangle \leq \langle\text{term}\rangle$	0.2
$\langle\text{atomic formula}\rangle := \langle\text{term}\rangle > \langle\text{term}\rangle$	0.2
$\langle\text{atomic formula}\rangle := \langle\text{term}\rangle \geq \langle\text{term}\rangle$	0.2
$\langle\text{term}\rangle := \langle\text{term}\rangle + \langle\text{term}\rangle$	0.2
$\langle\text{term}\rangle := s(\langle\text{term}\rangle)$	0.2
$\langle\text{term}\rangle := 0$	0.2
$\langle\text{term}\rangle := \langle\text{variable}\rangle$	0.4

Figure 1: Grammar of Presburger arithmetic and probabilities assigned to rules for generating a corpus

3.2 Algorithms considered

In addition to Hodes' procedure and Cooper's procedure, we also used variants of these, using a heuristic that quickly rejects invalid formulae (we will call it the QR heuristic).

The heuristic is as follows: to invalidate $\forall\vec{x}.\Phi(\vec{x})$ we show that a particular instance $\Phi(\vec{c})$ is false. That is, we instantiate all universally quantified variables in a formula $\forall\vec{x}.\Phi(\vec{x})$ with particular ground values (say 0 and 100) in all ways. In that way we get a quantifier free formula $\Phi(\vec{c})$, for which validity is quickly decided. This simple heuristic is obviously sound, but not complete. However, our experiments showed that this heuristic could be very important and very useful.[4]

[4]Obviously, this procedure could be used for all types of formulae (not just universally quantified ones): using this procedure we could transform (simplify) a formula to existentially quantified formula and try to disprove it using Cooper's procedure.

Thus we compared the following four procedures:

Hodes' procedure. For PRA. Recall from §2.2 that this procedure is incomplete even for the universally quantified fragment of PNA.

Cooper's procedure. For PNA. (Cooper presented two such procedures: we used the second, improved version [1972]. Besides, Cooper's procedure is originally defined for Presburger integer arithmetic, and in our experiments we used our version, slightly modified for PNA.)

DP-A. For a given formula try to disprove it using the QR heuristic; if it succeeds, the formula is invalid; otherwise, apply Cooper's procedure; if Cooper's procedure says `yes`, the formula is a theorem, otherwise it is not.

DP-B. For a given formula try to disprove it using the QR heuristic; if QR succeeds, the formula is invalid; otherwise, if the given formula is universally quantified, apply Hodes' procedure; if the answer is `yes`, then the formula is valid, if the answer is `no` or if a given formula is not universally quantified, then apply Cooper's procedure; if the answer is `yes`, then the formula is valid, otherwise the answer is `no`, and it is invalid.

4 Results

We ran the procedures described in §3.2 on each formula of the corpus, recording whether the formula was valid or invalid, and CPU time taken to decide, subject to a time limit of 100s. The following tables show the results with CPU time measured in milliseconds.[5]

4.1 QR heuristic contribution

Of the 10 000 formulae in the corpus, roughly 8000 were invalid, the remainder valid. DP-A using values of 0 and 100 was able to quickly reject all but 70 of these invalid formulae. (We decided to take values 0 and 100 because our experiments showed that additional values were not significantly contributing to the rejection rate, and on the other hand, using just 0, the heuristic rejected less than 5000 formulae.)

4.2 CPU time distribution

Table 1 shows number of formulae handled by procedures within a given time interval, together with mean CPU time (in ms). Each entry in the table is a pair, the first part of which is the number of formulae in that time interval, the second part is the mean time taken by the procedure. The totals column shows the total number of formulae handled by each procedure within 100 seconds, and the mean time for these formulae.

[5]Programs were written in Quintus Prolog; experiments were run on a 32Mb Sun SPARC 4.

Procedure	*CPU time (ms)* $< 10^2$	10^2–10^3	10^3–10^4	10^4–10^5	*Totals*
Hodes	9213/27	639/265	106/2546	28/30120	9986/154
Cooper	5678/48	3566/292	509/2620	120/31372	9873/650
DP-A	9361/18	552/278	51/2467	17/29012	9981/94
DP-B	9254/18	605/286	95/2298	22/30189	9982/122

Table 1: Number of formulae decided vs. CPU time

4.3 Effect of number of variables

Table 2 shows that the mean CPU time spent by the procedures increases with the number of variables. The number of formulae in the corpus containing a certain number of variables is shown, with a pair the first part of which is a percentage, the second is the mean CPU time.

The percentage is of those formulae having a particular number of variables completed within the 100 second time limit. In the 5-variable case Hodes' procedure processed 95.9%, but for Cooper's procedure this figure was 78.8%. This discrepancy seems to suggest that Cooper's procedure degrades with increasing number of variables; however, notice that DP-A performed *better* then Hodes' procedure. Thus the correct explanation is that Cooper's procedure is slower on *invalid formulae*, an effect seem more clearly in §4.5.

4.4 Effect of size of formula

The efficiency of Hodes' and Cooper's procedure is governed not just by propositional structure but by term structure too. To investigate this aspect of performance, we define the size of a formula/term as the sum of the sizes of its immediate subformulae/subterms plus one, taking the size of variables and constants as 0. Table 3 shows the results.

Cooper's procedure is most exposed here: the percentage of formulae decided within the time limit drops dramatically as the size increases. Hodes' on the other hand fares quite well. Once again though, the better performance of DP-A reveals that Cooper's procedure is struggling with invalid formulae which are more easily dealt with by the QR heuristic.

4.5 Effect of validity

Table 4 shows CPU time spent by the procedures according to validity of a formula (considering only those formulae decided by all procedures within the time limit).[6]

[6] 9868 formulae from our corpus were treated by all procedures—so, 132 formulae are 'missing' in table 4. However, our QR heuristic rejected 107 of them, and thus, at most 25 formulae could change the first two columns in the table 4 if we had taken higher time limit.

The first column pertains to formulae valid in both rational and natural arithmetic;[7] the second is for those invalid in the rationals and valid in the naturals; the third for those invalid in both theories. Notice that those in the second column would not be found to be valid formulae of natural arithmetic by Hodes' procedure. Note that the heuristic versions dramatically improve the performance of Cooper's procedure in the invalid cases.

We considered the procedure DP-B, anticipating that it would take advantage of Hodes' procedure on formulae valid both in PRA and PNA. We expected that these gains would outweigh losses in other cases and therefore we expected, in general, DP-B to be faster than DP-A. However, surprisingly, it turned out that Hodes' procedure performed worse than Cooper's procedure in this group of formulae. Consequently, DP-B failed to improve upon DP-A in any of groups of formulae according to validity. Of course, DP-B cannot exploit Hodes' procedure in the case of formulae on which Hodes' procedure returns `no`, since Hodes' procedure is incomplete; in such cases, Cooper's procedure must be called, and time spent in Hodes' procedure is wasted.

Validity	$\models_{PRA} F$ $\models_{PNA} F$	$\not\models_{PRA} F$ $\models_{PNA} F$	$\not\models_{PRA} F$ $\not\models_{PNA} F$	*Total*
# formulae	756	1214	7898	9868
Hodes	117	256	81	105
Cooper	33	204	758	634
DP-A	66	267	49	77
DP-B	148	523	58	122

Table 4: CPU times according to validity/invalidity of formula

4.6 Summary of results

On our corpus Cooper's procedure performed better than Hodes', on valid formulae, but it was much worse on invalid formulae. However, this is mitigated entirely by using the QR heuristic. Our conclusion then is that

[7] Out of 756 formulae valid in both rational and natural arithmetic 330 are ground, 311 with 1, 51 with 2, 29 with 3, 27 with 4 and 8 with 5 variables.

Procedure	*# variables/# formulae*					
	0/598	*1/3362*	*2/3603*	*3/1503*	*4/693*	*5/241*
Hodes	100/3	100/17	100/39	100/130	99.4/683	95.9/2883
Cooper	100/3	100/42	100/169	98.6/1202	92.1/3608	78.8/8306
DP-A	100/6	100/16	100/22	99.9/171	99.0/315	95.9/1432
DP-B	100/7	100/17	100/27	99.9/202	98.7/443	94.5/1978

Table 2: % completed/CPU time (ms) vs. number of variables

Procedure	*Size/# formulae*					
	1–10/8785	*11–20/1029*	*21–30/150*	*31–40/29*	*41–50/5*	*51–60/2*
Hodes	100/35	99.7/687	97.3/2772	79.3/4661	80.0/2430	100/1560
Cooper	99.8/225	93.7/3238	80.0/8655	55.2/10657	60.0/24690	50/38530
DP-A	100/24	99.3/460	94.7/1059	93.1/3156	80.0/4320	100/295
DP-B	100/31	99.2/647	94.0/1800	82.8/572	80.0/4537	100/310

Table 3: CPU time vs. size

the combination of Cooper's procedure and QR performs better than Hodes' procedure.

5 Towards integration

We made some preliminary experiments on using decision procedure in the *Clam* proof planner [Bundy *et al.*, 1990]. In this experiments we used an extension of our procedure DP-A, which could handle defined arithmetic functions (*double, half, minus, p*) and relations (*odd, even*). For that purpose, we use rewrite rules, which, for instance, using the theorem $\forall x.\forall y.double(x) = y \Leftrightarrow y = 2x$, it is sound to rewrite $F(double(x))$ into $\forall y.2x = y \Rightarrow F(y)$. Similarly, $\forall x.even(double(x))$ can be rewritten to $\forall x.\exists u.\forall v.2x = v \Rightarrow v = 2u$. Notice that these translations move outside the quantifier-free fragment of Presburger arithmetic,[8] so we are using DP-A in an area where Hodes' procedure would be unsound, and so useless.

We made experiments on 113 theorems from *Clam* corpus. The DP-A augmented version of *Clam* was slower on some proofs, and faster on others—we recorded the minimum for each theorem in the corpus. DP-A contributed to 65 proofs; it was used on 79 occasions to show a subgoal valid, and on 8 occasions to prune invalid subgoals (and so cause backtracking). *Clam* with DP-A was twice as fast as plain *Clam*.

6 Future work

It would be interesting to compare Hodes' and Cooper's procedures with Sup-Inf procedures.

The assignment of probabilities to the rules of the grammar for Presburger formulae was chosen somewhat arbitrarily—we have yet to investigate the effect these parameters have on validity, time to decide etc. A quickly computed measure of expected run time might be useful in a heuristic theorem prover. One can imagine pursuing a line of enquiry similar to that in the propositional satisfiability community.

According to results of Boyer and Moore [1988], an essential role of using an arithmetic decision procedure is to contribute to the proofs of deeper theorems in other theories (not just arithmetic). We have done some simple experiments in the *Clam* proof-planning system [Bundy *et al.*, 1990], and the results are promising. Our aim is to explore Bundy's idea of proof-plans for the flexible application of decision procedures [Bundy, 1991]. Transformation of a problem to a Presburger formula should be done by a communication module. First step in that direction is dealing with arithmetic defined functions and relations. After that, we are plan to deal with non-arithmetic functions (such as *list length*, *min*, *max* etc.) using decomposition techniques and different heuristics. The ability to move outside the quantifier-free restriction will be essential in this respect.

7 Conclusions

The effectiveness of DP-A (and hence any claim that Cooper's procedure is useful in tandem with the QR heuristic) must be offset by the fact that 80% of the corpus is invalid. Although Cooper's procedure outperforms Hodes' procedure on valid formulae, it did so to a lesser degree than DP-A outperformed Hodes' procedure on invalid formulae. Furthermore, another factor must be borne in mind. The corpus we generated did not contain many large constants, and the presence of these in formulae will often slow both Hodes' and Cooper's pro-

[8] Multiplication of a variable by a constant is Presburger: nx is treated as $x + \cdots + x$, where x appears n times.

cedure, and hence DP-A. Note that the Sup-Inf family of procedures is not affected in this way.

We need to be cautious when advocating the use of DP-A more widely—more experiments on other corpora are required, especially on the "real problems" generated during (say) inductive verification proofs. *With these caveats*, we draw the following conclusions:

- Cooper's procedure with the simple QR heuristic *outperformed* Hodes' procedures. This is a startling result. It goes against the grain of much work and commentary made on decision procedures in the past 25 years.
- When efficiency is comparable, it is highly preferable to use a decision procedure in a heuristic theorem prover rather than an incomplete decision procedure. Hodes' procedure (which is incomplete for quantifier-free PNA) fails to prove many PNA theorems; for such theorems much extra work may be incurred in trying other techniques (e.g., induction). We speculate that for most invalid PNA conjectures, even a slow decision procedure will be faster and more robust than heuristic techniques.
- Worst case analysis of complexity may be misleading: experimental evaluations can be useful.
- Even just a slightly extended decision procedure (to deal with arithmetic defined functions and relations) improved the performance of an inductive prover. The ability to move outside quantifier-free Presburger arithmetic was essential in the translation.

On the basis of these experiments, we conclude that for quantifier-free Presburger arithmetic over the natural numbers, Cooper's procedure augmented with a 'quick reject' heuristic is superior to Hodes' procedure. This is a startling result that questions much of the perceived wisdom in the automated reasoning community.

References

[Bledsoe, 1975] W. W. Bledsoe. A new method for proving certain Presburger formulas. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, Tbilisi, Georgia, U. S. S. R., 3–8 September 1975.

[Boyer and Moore, 1988] R. S. Boyer and J S. Moore. Integrating decision procedures into heuristic theorem provers: A case study of linear arithmetic. In J. E. Hayes, J. Richards, and D. Michie, editors, *Machine Intelligence 11*, pages 83–124, 1988.

[Bundy *et al.*, 1990] Alan Bundy, F. van Harmelen, C. Horn, and A. Smaill. The Oyster-Clam system. In M. E. Stickel, editor, *10th International Conference on Automated Deduction*, pages 647–648. Springer-Verlag, 1990. Lecture Notes in Artificial Intelligence No. 449. Also available from Edinburgh as DAI Research Paper 507.

[Bundy, 1991] Alan Bundy. The use of proof plans for normalization. In R. S. Boyer, editor, *Essays in Honor of Woody Bledsoe*, pages 149–166. Kluwer, 1991. Also available from Edinburgh as DAI Research Paper No. 513.

[Cooper, 1972] D. C. Cooper. Theorem proving in arithmetic without multiplication. In B. Meltzer and D. Michie, editors, *Machine Intelligence 7*, pages 91–99. Elsevier, New York, 1972.

[Hodes, 1971] Louis Hodes. Solving problems by formula manipulation in logic and linear inequalities. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence*, pages 553–559, Imperial College, London, England, 1971.

[Kreisel and Krivine, 1967] Georg Kreisel and Jean Louis Krivine. *Elements of mathematical logic: Model theory.* North Holland, Amsterdam, 1967.

[Presburger, 1930] Mojżesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Sprawozdanie z I Kongresu metematyków slowiańskich, Warszawa 1929*, pages 92–101, 395. Warsaw, 1930. Annotated English version also available [?].

[Shostak, 1977] Robert E. Shostak. On the SUP-INF method for proving Presburger formulas. *JACM*, 24(4):529–543, October 1977.

[Shostak, 1979] Robert E. Shostak. A practical decision procedure for arithmetic with function symbols. *JACM*, 26(2):351–360, April 1979.