MATHsAiD: a Mathematical Theorem Discovery Tool*

Roy L. McCasland †

Alan Bundy ‡
School of Informatics
University of Edinburgh
Edinburgh, Scotland, UK

Abstract

In the field of automated reasoning, one of the most challenging (even if, perhaps, somewhat overlooked) problems thus far has been to develop a means of discerning, from amongst all the truths that can be discovered and proved, those which are either useful or interesting enough to be worth recording. As for human reasoning, mathematicians are well known for their predilection towards designating certain discoveries as theorems, lemmas, corollaries, etc., whilst relegating all others as relatively unimportant. However, precisely how mathematicians determine which results to keep, and which to discard, is perhaps not so well known. Nevertheless, this practice is an essential part of the mathematical process, as it allows mathematicians to manage what would otherwise be an overwhelming amount of knowledge.

MATHsAiD is a system intended for use by research mathematicians, and is designed to produce high quality theorems, as recognised by mathematicians, within a given theory. The only input required is a set of axioms and definitions for each theory. In this paper we briefly describe some of the more important methods used by MATHsAiD, most of which are based primarily on the human mathematical process.

1 Introduction

In this paper we briefly describe some of the more important methods used by **MATHsAiD** (an acronym for **M**echanically **A**scertaining **T**heorems from **H**ypothese**s**, **A**x**i**oms and **D**efinitions) in attempting to produce, for a given mathematical theory, the sorts of theorems – and only those theorems – one might expect to find in a mathematics textbook on the theory. In our view, an automated mathematical theorem discovery tool should incorporate as much of the human mathematical process as is both possible and prudent. The point being that the human process enables mathematicians to discover and store that data which represents the "essence" of the given theory. This "essence" not only aids mathematicians in their understanding of the theory discovered thus far, but also aids them in discovering and proving still more of the truths contained within the theory. It is reasonable to expect that an automated reasoning system would likewise benefit from a similar ability to capture the essential truths.

A sample of the results produced by MATHsAiD, as well as an account of the philosophy behind the system, can be found in [7]. The authors are aware of other systems (see, for example, [2], [3], [4], and [6]) and procedures that have similar aims; perhaps most notable among the latter is the Knuth-Bendix procedure [5]. However, there are several significant differences in the methods employed (for instance, MATHsAiD does not need any examples provided, it does not try to directly measure "interestingness", it does not count usages of potential "lemmas", nor does it first prove a specific result, and only look for lemmas after that proof is complete).

2 The System

The code for MATHsAiD is written in two languages: most of what one might consider to be the mathematical process is carried out in Prolog; the rest is written in Java. The three main components of MATHsAiD are the automatic hypothesis generator (HG), the theorem generator (TG), and the theorem filter (TF). (It is important to note, however, that some filtering is already built-in to the theorem generating phase, making the overall process more efficient. See Figure 1.) In addition, there is a GUI that, among other things, facilitates the input of axioms/definitions (which we shall henceforth refer to merely as "axioms"), presents the resulting Theorems (i.e., the results that mathematicians would – hopefully, at least – refer

*This work is supported by EPSRC MathFIT grant GR/S31099. Part of this work was carried out while the first author was attending the Special Semester on Groebner Bases and Related Methods 2006, at RISC/RICAM.

†Email: rmccasla@inf.ed.ac.uk

‡Email: A.Bundy@ed.ac.uk

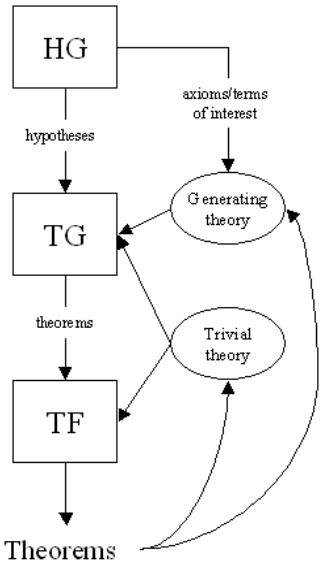


Figure 1. Component Diagram

to as either theorems, lemmas, corollaries, etc.; as opposed to the logicians' typical usage of the word "theorems") to the user, and allows the user to manually suggest various hypotheses and "terms of interest" for further exploration. (By "terms of interest" we mean any term that the user might want to find one or more Theorems pertaining to; for instance, given the hypotheses that A and B are sets, the user could designate the term $A \cap B$ as a "term of interest", in hopes of finding the Theorem that $A \cap B = B \cap A$). It also presents the user with a list of the theories already existing in MATHsAiD, and offers an opportunity for the user to begin a new theory. Once the user chooses a theory, he (or she, as the reader prefers) may at any time add new axioms to the database. He may also select any collection of axioms from within this theory, in order to initiate the Theorem discovery process.

Once the selection of axioms is made, the HG builds a sequence of inputs, each of which is fed, in turn, to the TG. From each input, the TG derives whatever conclusions it can, subject to numerous constraints, and sends its results to the TF. The TF then determines which, if any, of these conclusions should be recorded as Theorems. Once the TF has made its determination, the TG takes the next input from the HG, and the process continues, until the end of the sequence.

In storing each Theorem, and for that matter, each axiom, as a Prolog clause, we adopt the mathematicians' habit of partitioning each into a (possibly empty) set of hypotheses and a conclusion. This is, of course, well-suited to the standard Prolog "if, then" format.

2.1 Automatic Hypothesis Generator

The main purpose of the HG is to build a sequence of sets of hypotheses (each hypothesis typically involves one or more fixed, but arbitrary objects), and couple each set of hypotheses with certain judiciously chosen axioms or terms of interest. (For a sample of these sequences, see http://dream.inf.ed.ac.uk/projects/mathsaid/MATHsAiDResults.html). This allows MATHsAiD to focus, at any given time, on various local aspects of the theory under development, and thus, new Theorems may more easily be built up in layers, rather than be discovered all at one time. This not only makes it easier to discover new Theorems, but at any given time, the previously discovered Theorems can (and perhaps, as we argue later, should) be used to help determine which newly-discovered results should be stored, and which should be discarded.

Since the HG is meant to work across a broad spectrum of theories, it aims to facilitate the TG in discovering the more "routine" Theorems, such as commutativity and associativity for operations, and properties such as reflexivity, symmetry, and transitivity for relations, etc. In addition, for each axiom or Theorem for which 'converse' makes sense, one would like to know whether the converse is indeed true; thus the HG also identifies each such converse, to be provided to the TG towards the end of the sequence. As for any non-routine Theorems the user might wish to find (assuming they have not been found automatically), these are left to the user to provide the necessary hypotheses and terms of interest. We hope in future to expand MATHsAiD in this latter regard, either by coupling it with an interactive automated theorem prover, or by providing more extensive interactive capabilities within the system itself.

It is important to note that, only in the 'converse' case does the HG provide a conjecture for the TG to attempt to prove – in all other cases, the TG is simply given one or more axioms or terms of interest upon which to focus, plus the accompanying hypotheses, and then left to discover whatever Theorems it can. For instance, in order for MATHsAiD to discover that intersection of sets is commutative, the HG need only specify the hypotheses that A and B (say) are sets, and designate the definition of intersection as the axiom to consider (alternatively, one could provide the term of interest $A \cap B$). For another example, consider the situation $a < b$, where a and b are natural numbers. One might wish to determine, for any number c, whether the terms $a + c$ and $b + c$ compare similarly. In this case, the HG specifies the hypotheses that $a, b, c \in \mathbb{N}$ and $a < b$, and designates $a + c$ and $b + c$ as the terms of interest.

We remark that, regarding the sequence of inputs constructed by the HG, the ordering is rather important. The

HG provides, as a rule, the simpler hypotheses to the TG, before feeding it the more complicated hypotheses. For example, in our study of set theory, MATHsAiD discovers, early on, that for any set A, then $A = \varnothing$, iff $\forall x, x \notin A$. If this Theorem were not discovered and stored, then subsequent investigations would result in "Theorems" such as, $\forall x, x \notin A \cap \varnothing$ and $\forall x, x \notin A - A$ (where A is a set). While both of these statements are true, we would much prefer the simpler (corresponding) Theorems that $A \cap \varnothing = \varnothing$ and $A - A = \varnothing$.

More importantly, each recorded Theorem is available to be used in the filtering phase (see section 2.3). Thus, the absence of certain Theorems, in the early stages, can rather adversely affect the numbers of recorded "Theorems" in latter stages. For example, once the system records the Theorem $A \cap A = A$ (for a set A), then it will not allow any subsequent statement, involving the term $X \cap X$ (for any set X), to be recorded as a Theorem (preferring instead statements where $X \cap X$ has been replaced with X). However, if this initial result is not discovered in a timely manner, then the corresponding restriction is not enforced, resulting in numerous (bogus) "Theorems".

2.2 Theorem Generator

The theorem generating phase begins after the HG finishes building a finite sequence of sets of hypotheses $\{H_i\}_{i=1}^{n}$ coupled with their corresponding terms or axioms of interest. The latter are used primarily to restrict the search space, as we shall soon see. We remind the reader that, for each set of hypotheses H_k $(1 \leq k \leq n)$, we conduct an entire generating and filtering process, before moving on to the hypotheses H_{k+1}. This means that for each k, the hypotheses within H_k are asserted, in the Prolog sense, and thereafter, a forward-chaining process is used to produce whatever conclusions can be derived – using the theorem prover built in to the TG – from these hypotheses. However, the TG only asserts those conclusions that satisfy certain constraints, which we shall soon discuss. When no more new conclusions can be found, subject to these constraints, the (final) filtering stage begins, and provided that any conclusions are deemed by the TF to be worth recording, then each such conclusion is paired with H_k, and the resulting Theorems are added to the database. Any such Theorems are therefore available in all subsequent generating and filtering processes.

The most important constraint, for a conclusion to be asserted by the TG, is that it should not be 'trivial'. A full discussion as to the nature of 'trivial' is well beyond the scope of this paper, and indeed, we do not attempt to give a precise definition here. We point out, however, that trivialness is a dynamic, rather than a static property, in the sense that, for any given time, what is trivial depends a great deal on what is known at that time. And just as human mathematicians do not, as a rule, bother writing down everything that they can prove, regardless how trivial it might be, so too does MATHsAiD refrain from asserting everything that it can prove. In this sense, then, a great deal of filtering is being done throughout the theorem generation process.

As one example, consider the fact that intersection of sets is a commutative operation. In set theory, this is non-trivial, and is indeed regarded as a Theorem. However, in group theory, this property is considered trivial. This is evidenced by the fact that, while most (undergraduate) texts on group theory include the Theorem that, for subgroups A and B of a group G, then $A \cap B$ is likewise a subgroup of G, nevertheless, one would be hard-pressed to find any textbook that has a Theorem stating that both $A \cap B$ *and* $B \cap A$ are subgroups.

Therefore, for our theorem generating phase, it is useful to think in terms of having two theories, each of which varies with k – one to derive whatever conclusions can be proven from H_k, and the other to determine the trivialities among them. Hence, for each k $(1 \leq k \leq n)$, we let $\mathcal{G}_k$ denote the 'generating' theory, and we choose another theory $\mathcal{T}_k$, which represents the 'trivial' theory. As noted earlier, there is considerable flexibility in the choice of each theory, for each k. Nevertheless, we recommend that certain principles be followed, as outlined here, which are applied in MATHsAiD. The first principle is that, for all k, both $\mathcal{G}_k$ and $\mathcal{T}_k$ contain 'mere logic' (as viewed by many mathematicians). In the case of $\mathcal{G}_k$, all previously discovered Theorems, and at least some of the 'mathematics axioms' are represented as well; this is where the aforementioned axioms and terms of interest come into play. As for $\mathcal{T}_k$, it is more than just a theory of pure logic; it is sufficient to eliminate all instances of 'simple' conclusions of the 'mathematics axioms' (i.e., all c_σ such that $\exists h.(h \rightarrow c) \in \mathcal{A}$, where $\mathcal{A}$ is the set of non-logical axioms). Moreover, while $\mathcal{T}_k$ clearly ought not contain all possible combinations of axioms (else, every conclusion would be trivial), it does contain simple combinations of certain (e.g., commutativity and associativity) axioms.

Somewhat similarly, $\mathcal{T}_k$ contains at least some of the conclusions of previously catalogued Theorems; however, there are definite differences, when it comes to trivialities, between axioms and Theorems. A mathematician would never consider a conclusion of an axiom to be a Theorem (it's an axiom!); but many corollaries are, in fact, specific instances of Theorems. It is not yet entirely clear to us precisely what ought to distinguish a trivial from a non-trivial use of a Theorem. For the time being, a conclusion of a Theorem is considered trivial, if the verification of the Theorem's hypotheses does not require the use of any other Theorem. While this is not entirely satisfactory, from either a mathematical or a logical point of view, it nevertheless

seems thus far to work reasonably well.

We remark that in MATHsAiD, while $\mathcal{G}_k$ and $\mathcal{T}_k$ vary with the circumstances, the (relatively simple) rules which determine each theory do not vary. In both cases, the variance is due primarily to the "knowledge" at hand – the set of axioms provided by the user, and the Theorems that have been discovered and stored. In the case of $\mathcal{G}_k$, its contents are also influenced by the axioms/terms of interest included in the generated set of hypotheses H_k. The simplicity of this approach, and the evolving nature of the theories $\mathcal{G}_k$ and $\mathcal{T}_k$, bodes well for MATHsAiD's potential in working with many mathematical domains, including scaling up to domains that are more involved than those we have thus far studied.

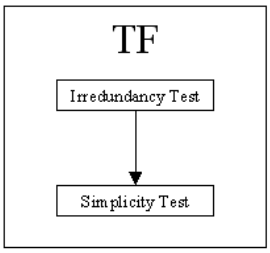
Once the theories $\mathcal{G}_k$ and $\mathcal{T}_k$ have been determined, the TG generates – i.e., proves – a preliminary set of theorems (conclusions),

$$C_k = \{c : H_k \vdash_{\mathcal{G}_k} c \text{ and } H_k \nvdash_{\mathcal{T}_k} c\} \ .$$

In order to ensure that C_k is finite, it suffices to impose an upper bound on the number of functors (in the Prolog sense) allowed for each $c \in C_k$. Each such $c \in C_k$ is given a unique identifying number, and, coupled with its 'proof list', is asserted by the TG. By 'proof list', we mean a list (proof script) that records all the steps involved in the proof of the given statement. This proof list becomes important in the final theorem-filtering stage.

We recognise that some conclusions might fail one or more of the constraints, and therefore not be theorems worth recording, yet they could conceivably still be useful in advancing the knowledge at the moment. In these cases, such conclusions are marked as "valid", and are made available to the TG for the duration of that session. These "valid" conclusions do not, however, reach the TF.

It is perhaps worth pointing out that all equalities and logical equivalences are stored, for the duration of the k session, in equivalence classes; each class is named by its 'simplest' member (as defined in section 2.3). Each time a new equality or equivalence is to be asserted, the relevant classes are tested for overlaps; if any are found, then the appropriate classes are merged accordingly, and renamed, if necessary. When the TG has finished generating C_k, then this set is modified in the following manner: for each equivalence class, the 'simplest' member is paired with each other member of that class, joined of course by the appropriate equivalence relation. Our experience shows that any Theorem involving an equivalence relation is likely to be of this form (i.e., one term will be the 'simplest' member of the relevant equivalence class). And besides, this technique gives us an effective way of dealing with the symmetry and transitivity issues, and ensures that we do not end up stating the obvious conclusion that an object is equivalent to itself.



2.3 Theorem Filter

As stated earlier, a considerable amount of theorem filtering has already taken place, before any conclusions reach the TF. Nevertheless, not all conclusions that make it this far should be regarded as Theorems. Thus, the conclusions sent to the TF are run through a (fairly small) number of tests (see Figure 2), not all of which will be discussed here. Any conclusion that fails a test is immediately discarded. At the completion of the testing phase, each conclusion that remains is coupled with the hypotheses, and recorded as a Theorem.

We should point out that – given the way MATHsAiD deals with conclusions involving equivalence relations, by first converting the information into equivalence classes, and then converting back into (albeit different) conclusions – some trivialities will almost inevitably have been introduced. However, this is easily remedied, by removing from C_k all (new equivalence-type) conclusions c such that $H_k \vdash_{\mathcal{T}_k} c$. We are convinced that the advantages gained in handling equivalences this way, far outweigh this temporary disadvantage.

2.3.1 Irredundancy Test

The first test, following the conversion of equivalence classes, is to determine whether all the hypotheses in H_k are actually required for a given conclusion c. If not, then c is discarded – on the grounds that if it were a Theorem, it should have been found earlier (i.e., for some j such that $j < k$). This of course presumes that the HG has done a proper job of building the sequence of sets of hypotheses. In the notation of the preceding section, we remove from C_k all c such that

$$A \vdash_{\mathcal{G}_k} c \text{ for some } A \subsetneq H_k \ .$$

The set of conclusions that remain at this point will be denoted by C'_k.

2.3.2 Simplicity Test

It is at this point that our interpretation of 'simplest' plays its most important role (having previously been used to deter-

mine the name of each equivalence class). Each conclusion in C'_k is measured, according to the following variant (denoted $m(t)$) on the standard size measure on expressions. Namely,

$$m(v) ::= 0; \text{ where } v \text{ is a variable}$$
$$m(f(s_1, \ldots, s_n)) ::= 1 + \sum_{i=1}^{n} m(s_i); \text{ where } f \text{ is a function or a non- '=' predicate}$$
$$m(a = b) ::= \frac{1}{2} + m(a) + m(b)$$
$$m(Qx.t) ::= 1 + m(t); \text{ where } Q \text{ is either } \exists \text{ or } \forall .$$

Regarding 'simplicity', we only compare members that are related by one of their respective proof lists being embedded contiguously in the other. The motivation behind this is as follows: a proof list represents, in some sense, a 'line of reasoning'. In our experience in conducting mathematics research, such a line of reasoning is pursued, until one reaches a satisfactory conclusion – usually, the simplest conclusion found during that particular pursuit. (One might imagine a mathematician working with an expression/conclusion that, while somewhat interesting in its own right, nevertheless should be further examined/manipulated, in order to determine whether a better/more interesting representation can be found). While this simplest conclusion, and therefore the Theorem of choice, more often than not corresponds to the embedding (longer) proof list, there are occasions where the embedded (shorter) proof list provides the better Theorem. The latter case represents a situation where, once a non-trivial conclusion has been reached, any attempts to continue pursuing that line of thought, succeed only in further complicating matters.

To be more precise, for a conclusion $c \in C'_k$, let $pl(c)$ denote the proof list of c. Then for $s, t \in C'_k$, we say that s and t are *comparable*, denoted $comp(s, t)$, if either $pl(s)$ is contiguously embedded in $pl(t)$ (denoted by $pl(s) \hookrightarrow pl(t)$), or the other way round. In other words,

$$comp(s, t) \iff pl(s) \hookrightarrow pl(t) \text{ or } pl(t) \hookrightarrow pl(s) .$$

We are now ready to define the lexicographic order $\ll$ on the set C'_k, that is used in the final determination of Theorem-hood.

Definition 1 *Using the above notation, for $s, t \in C'_k$, then*

$$s \ll t ::= \begin{cases} \top & comp(s, t) \textit{ and } m(s) < m(t) \\ \top & m(s) = m(t) \textit{ and } pl(t) \hookrightarrow pl(s) \\ \bot & \textit{otherwise} . \end{cases}$$

Finally, the Theorems are simply the minimal elements of C'_k, relative to $\ll$.

3 Conclusions and Future Work

The results (see http://dream.inf.ed.ac.uk/projects/mathsaid/MATHsAiDResults.html, in addition to [7]) produced by MATHsAiD thus far have been excellent, in that a high percentage – typically at least 80% – of its Theorems can be found in at least some textbooks, and likewise, a similar percentage of textbook Theorems – amongst those that can be proven with the axioms provided thus far to MATHsAiD – are indeed among its Theorems. Admittedly, the system has as yet only been applied to a few domains (e.g., set theory, the positive integers and group theory), and only to a rather rudimentary level in each. It is, of course, impossible to know whether we can achieve the same quality of results in all areas of mathematics, until we try them. That said, it is not at all clear that the criteria for Theorem-hood should change, from one mathematical domain to the next. Moreover, since the methods employed in this system are rather simple, and indeed, quite comparable to the human mathematical process, the prospects for this system seem, at the very least, encouraging.

As for future work, we plan to automate the discovery of theorems which are typically proved by mathematical induction; for example, theorems involving integer-exponents of group elements. We also intend to develop sufficient underlying theories, as to enable the study of module-theory, and perhaps Zariski Spaces. Another possible topic of study is the theory of Groebner Bases (see, for example, [1]). In the more distant future, assuming that our studies of these algebraic structures (our areas of expertise) are successful, we plan to move into the analytic branch of mathematics. In order to handle a broad range of domains, however, the automatic hypothesis generator (HG) needs to first be enhanced. In addition, we intend to combine MATHsAiD with one or more interactive theorem-proving systems, anticipating that the ability to discover high-quality Theorems will considerably improve the performance of these systems.

References

[1] B. Buchberger, Introduction to Groebner Bases, In: Groebner Bases and Applications (B. Buchberger, F. Winkler, eds.), Cambridge University Press, 1998, pp.3-31.

[2] S. Colton, *Automated Theory Formation in Pure Mathematics*, Distinguished Dissertations Series, Springer-Verlag, London, 2002.

[3] J. Denzinger and S. Schulz, Recording and Analysing Knowledge-Based Distributed Deduction Processes, *Journal of Symbolic Computation*, Vol. 21, 523–541, 1996.

[4] Y Gao, Automated Generation of Interesting Theorems, Masters Thesis, University of Miami, 2004.

[5] D. Knuth and P. Bendix, Simple word problems in universal algebras, In J. Leech, editor, *Computational Problems in Abstract Algebras,* Pergamon Press, Oxford, U.K., 263–297, 1970.

[6] D. Lenat, *AM: An Artificial Intelligence approach to discovery in mathematics*. Unpublished PhD thesis, Stanford University, 1976.

[7] R. L. McCasland, A. Bundy, and P. F. Smith, Ascertaining Mathematical Theorems, Electronic Notes in Theoretical Computer Science, Vol. 151(1), 21-38, 2006.