1 Background

1.1 Motivation

Given a logical theory, such as the theory of lists shown in Figure 1, we may want to find theorems which describe its behaviour. This could be for mathematical curiosity, or due to the theory's importance in some domain. In particular, for theories which capture the semantics of some software library, we may want to verify that certain (un)desirable properties do (not) hold. We might also want to optimise programs using this library, rewriting expressions into a form which requires less time, memory, network usage, etc. To avoid altering a program's result, such rewrites should come with theorems proving their correctness, such as the following theorem for our theory of lists:

8f:8xs:8ys:map(f; append(xs; ys)) = append(map(f; xs); map(f; ys)) (1) [**]

This justifies a rewrite rule for splitting a single map call into multiple independent calls dealing with different sections of a list. Such rules are useful optimisations since each call can be evaluated in parallel, leading to the /reduce" programming paradigm. All of these example use cases require the ability to discover theorems about some particular theory. This is a hard problem in general, but presents opportunities for automation due to the precise, symbolic nature of the domain.

1.2 Automating Construction and Exploration of Theories

The task of discovering theorems in an arbitrary theory can be described as the interplay of several processes: (i) formulating new definitions and concepts; (ii) finding patterns suitable for posing as conjectures; and (iii) proving that those conjectures are theorems.

The latter (iii) is studied extensively in the field of Automated Theorem Proving (ATP). Far less attention has been paid to automating the first two tasks, which both Automated Theory Formation (ATF) and Automated Theory Exploration (ATE) address. We discuss the relationship between ATF and ATE in historical context in §2.

A major challenge for both ATF and ATE is choosing how to narrow down the set of generated conjectures to those deemed "interesting", since this is an imprecise term with many different interpretations. For example, all existing approaches agree that simple tautologies are "uninteresting", but differ when it comes to more complex statements. Colton *et al.* give a survey of systems for theory formation and exploration, and their notions of “interestingness” for concepts and conjectures, identifying six notions in total.

are identified in these systems, their use by each is summarised, along with (our interpretation of) their use in three more recent ATE systems. These notions, as applied to ATE, are:

Empirical plausibility, which checks whether a property holds across some specific examples. This is especially useful for avoiding false conjectures, without resorting to a full proof search.

Novelty is whether a conjecture, or one isomorphic or more general, has already been seen.

Surprisingness of a conjecture is whether or not it is “obvious”, for example if it is an instance of a tautology.

Applicability depends on the number of models in which a conjecture holds. The system of Bagai et al conjectures the non-existence of objects, and hence favours statements with zero applicability. Other systems treat ap- plicability as a positive aspect: the more applicable the statement, the more interesting it is.

Comprehensibility is the complexity of a statement. Simpler statements are considered more interesting, and many of the search algorithms explore simpler/smaller statements before complex/larger ones, to more efficiently find those which are interesting.

Utility is the relevance or usefulness of a conjecture to the user’s particular task. For example, if we want to find optimising rewrite rules such as equation 1, then utility would include whether or not a conjecture justifies a rewrite rule, the difference in resource usage of the expressions involved, and how common those expressions are in real usage.

We summarise the criteria in Table 1 and state which were used in key ATF/ATE systems, where Colton identified usage in AM, GT, Graffiti and Bagai et al, and his own ATF system HR, and we identify usage in three more recent ATE systems QuickSpec, IsaCoSy and IsaScheme. These criteria, especially empirical plausibility and comprehensibility, are not only used for analysing results after the fact, but instead may form a core part of an algorithm’s design decisions. For example, the designers of IsaCoSy consider simple substitutions of existing conjectures (i.e. those with low novelty) to be uninteresting, and hence their system includes a constraint solving component which avoids generating such statements entirely.

With such ambiguous and varied goals, approaches and assessment criteria it is difficult to compare ATE systems in a quantitative way, and hence to form some measure of "progress" for the field.

2 Theory Exploration versus Theory Formation

In this section we put the Mathematical Theory Exploration approach into context both historically, and with respect to related approaches such as in Automated Theory Formation systems.

2.1 A Short History of Automated Mathematics

> ..."in his subsequent design for an Analytical Engine Mr. Babbage has shown that material machinery is capable, in theory at least, of rivalling the labours of the most practised mathematicians in all branches of their science." [14, p. 498]

The first automated mathematical system, the mechanical calculator (known as the Pascaline), was an adding machine that could perform additions and subtractions directly and multiplication and divisions by repetitions, and was conceived by Pascal in 1642 while reorganising tax revenues [10]. Subsequent early systems include Müller's universal calculating machine in 1784, which he built for the purpose of calculating and printing numerical tables: he invented this when he had to check and recalculate some tables relating to the volumes of trees [17, p. 65]. Thirty seven years later, Babbage invented his famous difference engine: an automatic, mechanical calculator designed to tabulate polynomial functions. This was inspired by a flawed table of logarithms and the idea that machines would be quicker and more reliable [1]. It is interesting to note background and motivation: while Pascal and Babbage were mathematicians, with an interest in engineering, Müller was an engineer with an interest in mathematical knowledge. All three systems were conceived as an aid to mathematicians, as well as scientists, accountants and surveyors.

Differences in background and motivation continue to be relevant today. While the majority of work in automating mathematics has been in symbolic manipulation and theorem proving, we are concerned here with other aspects of mathematics, including the construction of concepts, axioms, conjectures, examples and theorems. This is varyingly known as "Automated Theory Formation" [16; 8], "Automated Mathematical Theory Exploration", "Mathematical Theory Exploration" [5] (also sometimes prefaced with "Computer-Aided" or "Algorithm-Supported"), "Automated Mathematical Discovery" [11; 9; 7], "Concept Formation in Discovery Systems" [12], and "Automated Theorem Discovery" [18]. Such a plethora of terminology can be unhelpful and can mask similarities between the different fields. In particular, the twin strands of Automated Theory Formation and Automated Mathematical Theory Exploration seem to be developing somewhat independently without a clear differentiating methodology. Below we discuss commonalities and differences between the two schools of thought.

2.2 Automated Theory Formation

Automated theory formation, the focus of this paper, derives its terminology from psychology in which the term "concept formation" is used (see, for example, [2]) to describe the search for features which differentiate exemplars from non-exemplars of various categories. Lenat used this term in his 1977 paper: *Automated Theory Formation in Mathematics* [16] and we have extended it to describe a family of techniques used to construct and evaluate concepts, conjectures, examples and proofs (embodied in the HR, HRL and GC systems and derivatives, as described above).

When Lenat built the AM system [16], there were systems which could define new concepts for investigation, such as those described in [22], and systems which could discover relationships among known concepts, such as Meta-Dendral [3]. No system could perform both of these tasks: Lenat saw this as the next step. AM was designed to both construct new concepts and conjecture relationships between them; fully automating the cycle of discovery in mathematics. Lenat describes this as follows:

> "What we are describing is a computer program which defines new concepts, investigates them, notices regularities in the data about them, and conjectures relationships between them. This new information is used by the program to evaluate the newly-defined concepts, concentrate upon the most interesting ones, and iterate the entire process." [16, p. 834]

AM was a rule-based system which used a frame-like scheme to represent its knowledge, enlarged its knowledge base via a collection of heuristic rules, and controlled the firing of these rules via an agenda mechanism. Lenat chose elementary arithmetic as the development domain because he could use personal introspection for the heuristics for constructing and evaluating concepts. Given the age of this discipline, Lenat thought it unlikely that AM would make significant discoveries, although he did cite its "ultimate achievements" as the concepts and conjectures it discovered (or could have discovered). He suggested various criteria by which his system could be evaluated, many of which focused on an exploration of the techniques. For instance, he considered generality (running AM in new domains) and how finely-tuned various aspects of the program are (the agenda, the interaction of the heuristics, etc). Lenat saw his system and future developments in this field as having implications for mathematics itself (finding results of significance), for automating mathematics research (developing AI techniques), and for designing "scientist assistant" programs (aids for mathematicians). This shows a broad spread of motivation. Despite the seeming success of the AM system, it is one of the most criticised pieces of AI research. In their case study in methodology, Ritchie and Hanna analysed Lenat's written work on AM and found that there was a large discrepancy between his theoretical claims and the implemented program [20]. For instance, Lenat made claims about how AM invented natural numbers from sets, whereas it used one heuristic which was specifically written in order to make this connection (and not used in any other context). Another problem was that the processes were sometimes under-explained. For an argument of why many of the claims made by Lenat about AM were false, see chapter 13 of [8].

2.3 Mathematical Theory Exploration

The phrase Mathematical Theory Exploration (MTE) is a recent term which seems to have originated with Buchberger and colleagues (see, for example, [6]). His motivation is to support mathematicians during their exploration of mathematical theories. This support is intended to be for the straightforward reasoning, which he argues, covers most mathematical thought, rather than the ingenious points, which he leaves for human mathematicians. Buchberger's long term goal is to provide routine tools for the exploration activity of working mathematicians, to support the invention and structured build-up of mathematical knowledge. The Theorema project aims at prototyping features of a system for such a purpose. These features include Integration of the Functionality of Current Mathematical Systems (retention of the full power of current numerics and computer algebra systems, as well as enabling the user to add their own algorithms to the system); Attractive Syntax (input and output is readable and presented attractively, and can be personalised by the user); and Structured Mathematical Knowledge Bases (tools are provided for building and using large mathematical knowledge libraries). Buchberger has evaluated the potential of this strategy by illustrating the automated synthesis of his own Gröbner bases algorithm [4].

Recent systems developed in this area include MATHsAiD [18], IsaScheme [19] and IsaCosy [15]. The goal of the MATHsAiD (Mechanically Ascertaining Theorems from Hypotheses, Axioms and Definitions) project is to build a tool which takes in a set of axioms, concept definitions and a logic and applies its inference rules to reason from the axioms to theorems. The motivation is to produce a tool which will help mathematicians to explore the consequences of a set of axioms or a particular concept definition. One of the main challenges of the project has been to automatically evaluate interestingness: to distinguish important theorems from results which, although they follow from a set of axioms, are of little mathematical interest.

Montaño-Rivas *et al.* have implemented a scheme-based approach to MTE in their IsaScheme system [19]. Schemes are higher-order formulae which can be used to generate new concepts and conjectures; variables within the scheme are instantiated automatically and this drives the invention process. For instance, in the theory of natural numbers, given the concepts of successor (suc), addition ($+$) and zero (0), IsaScheme can use the following scheme to invent the concept of multiplication:

$$def - scheme(g,h,i,j) \equiv \exists f.\forall xy. \begin{cases} f(g,y) = h(y)\ and \\ f(i(x),y) = j(y, f(x,y)) \end{cases}$$

where the existentially quantified variable f stands for the new function to be defined in terms of the variables g, h, i and j. Within the theory of natural numbers, IsaScheme instantiates this scheme with $\sigma_1 = \{g \mapsto 0, h \mapsto (\lambda x.0), i \mapsto suc, j \mapsto +\}$. In this example, $f \mapsto *$ (multiplication), since:

0*y = 0
suc(x) * y =y + (x*y). The new multiplication function f can itself be used to instantiate variables in the same scheme, resulting in the invention of the exponentiation concept (these examples are taken from [19]).

The IsaCosy system (Isabelle Conjecture Synthesis) is a program for inductive theory formation, which synthesises conjectures from the available constants and free variables. Only terms which do not contain subterms that can be rewritten by a set of rewrite rules can be synthesised. Conjectures are tested by sending them to a counterexample checker and, if no counterexamples are found, then sent to IsaPlanner which attempts to prove them.

2.4 Commonalities and Differences between MTE and ATF

We believe that ATF and MTE share many goals and would benefit from a closer alignment: both fields are highly specialised and by joining forces (while being explicit about differences) the techniques being developed would have greater impact. This closer alignment is already beginning to take place. The main commonalities between MTE and ATF are that both are interested in the same aspects of mathematical thinking, aiming to automatically construct and evaluate elements of a mathematical theory, including concepts, conjectures, theorems, axioms and examples. Both strands contrast themselves with Automated Theorem Proving. The main historical differences between these two approaches seem to be that in MTE:

- the main proponents define themselves as mathematicians (they hold a PhD and have experience in research mathematics);
- the primary motivation is to support mathematicians;
- systems tend to be user-interactive;
- systems tend to be specific to mathematics,

while in ATF:

- the main proponents define themselves as AI researchers (they often have a mathematics background up to Masters level, but their PhD and research experience is in AI);
- the primary motivation is to extend AI techniques; a secondary motivation is to produce interesting new mathematics;
- systems tend to be fully automated;
- systems can often be applied to non-mathematical domains.

The different backgrounds of the people who named the fields "automated theory formation" (Lenat) and "mathematical theory exploration" (Buchberger) perhaps reflects different metaphysical perspectives on invention (in which case new mathematical ideas are being *formed*, created or produced) and discovery (in which case abstract mathematical objects exist independently of us and are *explored* or investigated) in mathematics.[1] As can be seen above, some of these historical

[1]There is a vast literature on the Platonic and the constructivist view of mathematics: we shall not venture down this path here (interested readers are referred to [13; 21]). Instead, for now, we make the pragmatic assumption that while there may well be different cognitive processes involved in invention than those involved in discovery, the fields of ATF and MTE are not currently concerned with this level of detail and so the philosophical distinction is not relevant for our purposes.

differences are disappearing, and current systems such as MATHsAiD [18], IsaScheme [19] and IsaCosy [15] are bridging the methodological gap between ATF and MTE. This is particularly true in terms of fully-automatic/user-interactive and the domain-specific/general aspects.

References

[1] B. V. Bowden (Ed). *Faster Than Thought: A Symposium on Digital Computing Machines*. Pitman Publishing, London, UK, 1953.

[2] J. Bruner, J. J. Goodnow, and G. A. Austin. *A study of thinking*. Science Editions, New York, 1967.

[3] B. Buchanan. Applications of artificial intelligence to scientific reasoning. In *Second USA-Japan Computer Conference*, pages 189–194, Tokyo, 1975. AFIPS and IPS I.

[4] B. Buchberger. Towards the automated synthesis of a gröbner bases algorithm. *RACSAM (Review of the Royal Spanish Academy of Science)*, 98(1):6575, 2004.

[5] B. Buchberger. Mathematical theory exploration. *Invited talk at IJCAR. www.easychair.org/FLoC-06/buchberger_ijcar_floc06.pdf*, 2006.

[6] B. Buchberger, A. Craciun, T. Jebelean, L. Kovacs, T. Kutsia, K. Nakagawa, F. Piroi, N. Popov, J. Robu, M. Rosenkranz, and W. Windsteiger. Theorema: Towards computer-aided mathematical theory exploration. *Journal of Applied Logic*, pages 470–504, 2006.

[7] J. Charnley and S. Colton. Applications of a global workspace framework to mathematical discovery. In *Proceedings of the Conferences on Intelligent Computer Mathematics workshop on Empirically Successful Automated Reasoning for Mathematics*, 2008.

[8] S. Colton. *Automated Theory Formation in Pure Mathematics*. Springer-Verlag, 2002.

[9] S. Colton, A. Bundy, and T. Walsh. On the notion of interestingness in automated mathematical discovery. *International Journal of Human Computer Studies*, 53(3):351–375, 2000.

[10] M. d'Ocagne. Le calcul simplifié. *Annales du Conservatoire national des arts et métier. 2e série*, 5,, 1893.

[11] S. L. Epstein and N. S. Sridharan. Knowledge representation for mathematical discovery: Three experiments in graph theory. *Journal of Applied Intelligence*, 1(1):7–33, 1991.

[12] K. Haase. Discovery systems. Technical Report 898, MIT, 1986.

[13] R. Hersh. *What is mathematics, really?* Oxford University Press, UK, 1997.

[14] W. S. Jevons. On the mechanical performance of logical inference. *Philosophical Transactions of the Royal Society of London*, 160:497–518, 1870.

[15] M. Johansson, L. Dixon, and A. Bundy. Conjecture synthesis for inductive theories. *Journal of Automated Reasoning*, Forthcoming, 2010.

[16] D. B. Lenat. Automated theory formation in mathematics. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 833–842, Cambridge, MA, 1977. Morgan Kaufmann.

[17] C. G. Lindgren, M. (translated by McKay. *Glory and Failure: The Difference Engines of Johann Müller, Charles Babbage, and Georg and Edvard Sheut*. The MIT Press, USA, 1990.

[18] R. McCasland and A. Bundy. MATHsAiD: a mathematical theorem discovery tool. In *Proceedings of SYNASC*, 2006.

[19] Omar Montano-Rivas, Roy McCasland, Lucas Dixon, and Alan Bundy. Scheme-based theorem discovery and concept invention. *Expert Systems with Applications*, (0):–, 2011.

[20] G. Ritchie and F. Hanna. AM: a case study in methodology. In D. Partridge and Y. Wilks, editors, *The foundations of AI: a sourcebook*, pages 247–265. CUP, Cambridge, 1990.

[21] S. Shapiro. *Thinking about Mathematics: The philosophy of mathematics*. OUP, Oxford, 2000.

[22] P. Winston. Learning structural descriptions from examples. Technical Report TR-231, MIT, 1970.