Machine Learning Case Splits for Theorem Proving

Simon Colton*, Ferdinand Hoermann*, Geoff Sutcliffe** and Alison Pease***

*Department of Computing, Imperial College London, UK. {sgc,fmh02}@doc.ic.ac.uk
** Department of Computer Science, University of Miami, USA. geoff@cs.miami.edu
***School of Informatics, University of Edinburgh, UK. A.Pease@sms.ed.ac.uk

1 Overview

We believe that in order to build more powerful AI systems, it will be necessary to combine techniques from machine learning, automated deduction, constraint solving, computer algebra, planning and other domains. In particular, as argued by philosophers such as Lakatos [5], mathematical discovery processes rely on a plethora of reasoning techniques, including deduction, induction, abduction, symbolic manipulation, etc. To demonstrate that the whole can be more than the sum of the parts when combining systems, we have performed a number of case studies in automated mathematics. These demonstrate that, with respect to stand-alone systems, combined reasoning systems can: (a) be more flexible in their application [4] (b) undertake new tasks [2] and (c) perform standard tasks better [3].

We present here the initial stages of a case study which we hope will further demonstrate that combined reasoning systems have the potential to out-perform single technique systems at standard tasks, in this case proving theorems. In particular, we show that descriptive machine learning can be used to suggest ways to split theorems into cases. This can be done in such a way that the combined time to prove all the cases is less than the time to prove the unaltered theorem. We use the HR descriptive learning system [1] to provide the case splits and Otter [6] as the theorem prover, with the TM system [4] providing the integration of these systems. While Otter is not the fastest prover, it has certain advantages, not least the fact that it is probably the most used prover in the mathematical community. Hence, optimizing the runtime in the application of Otter is useful, and doing it in a cognitively plausible way such as case splitting may be of interest to the mathematicians who use Otter. Indeed, it may be that the case splits provided by the combined system are more interesting than the fact that the theorem provided was true.

2 Case Splitting Procedure

We build on the work presented in [4]. In that application, we used TM to provide alternative proved theorems to a given non-theorem. For instance, given the non-theorem that all groups are Abelian, TM replied by saying that this was false, but that self-inverse groups are Abelian. In the current application, we supply TM with a theorem of the form $Ax \models S$, where Ax is a set of axioms which define a domain, and S is a statement which is hypothesised to follow from the axioms. TM generates a set of additional axioms to be added as extensions to the original set. We denote these extensions $E_1, \ldots, E_m$, and they have the property that:

- Otter has proved that $\forall i \ (Ax \wedge E_i \models S)$
- $E_1 \vee \ldots \vee E_n$ follows from the definitions of the extensions.

Given this, we can infer that $Ax \wedge (E_1 \vee \ldots \vee E_m) \models S$, hence $Ax \wedge \text{True} \models S$, hence $Ax \models S$. So, if TM can generate such a set of axiom extensions, it will have effectively proved the theorem.

To produce the axiom extensions, TM first uses HR to generate a theory from the axiom set Ax. Details of how HR works are given in [1], and for our purposes, we only need to know that the theory contains, amongst other things, a set of Boolean concepts $C_1, \ldots, C_n$ which are true of at least one model of the domain. To generate the extensions, TM takes C_1 and attempts to prove both: $Ax \wedge C_1 \models S$ and $Ax \wedge \neg C_1 \models S$. If both are proved, then the overall theorem is proved. Supposing, however, that the latter conjecture cannot be proved, then TM further specialises the theorem into $Ax \wedge \neg C_1 \wedge C_2 \models S$ and $Ax \wedge \neg C_1 \wedge \neg C_2 \models S$. This continues in the obvious fashion, so that each E_i is a conjunction of a subset of $\{C_1, \ldots, C_n, \neg C_1, \ldots, \neg C_n\}$, and the disjunction of all the extensions covers all models of the domain. TM maintains an agenda of specialised theorems to prove and is able to tell when the set of proved ones constitutes a proof of the entire theorem.

3 Preliminary Results

This study is very much a work in progress, and while our preliminary results are certainly not conclusive, they are encouraging. We have concentrated on a particular theorem, namely GRP119-1.p from the TPTP library [8], which is paraphrased as follows. Given the following single axiom:

$$\forall \ x, y, z \ ((y * ((y * y) * (x * z)) * (z * (z * z))) = x)$$

and the fact that the identity element is such that: $id * id = id$, then all elements in the algebra are of order 4, i.e., $\forall \ a \ (a * (a * (a * a)) = id)$. This theorem originally came from [9], and takes Otter 74 CPU seconds to prove on a 2.4Ghz intel processor.

Noting that the single axiom defines order four *groups*, we used HR to generate a theory of groups, for approximately 30 minutes. This produced 20 non-negated binary concepts to be used in case splits. After some experimenting, and using only statistics from Otter run-times, we hand-ordered the concepts (note that such hand-crafting is commonplace in, for example, constraint solving). We then ran TM in case split mode, allowing it 6 seconds for each case split. TM could not prove the original theorem, but went on to produce 4 case splits, 3 of which were proved, as in Figure 1. The proofs of the three case splits constituted a proof of the original theorem. More importantly, TM ran for only 10 seconds. This time includes the calls to Otter, but not, of course, the 30 minutes of HR time. However, we note that HR only has to be run once in group theory and the results can be applied to any theorem.

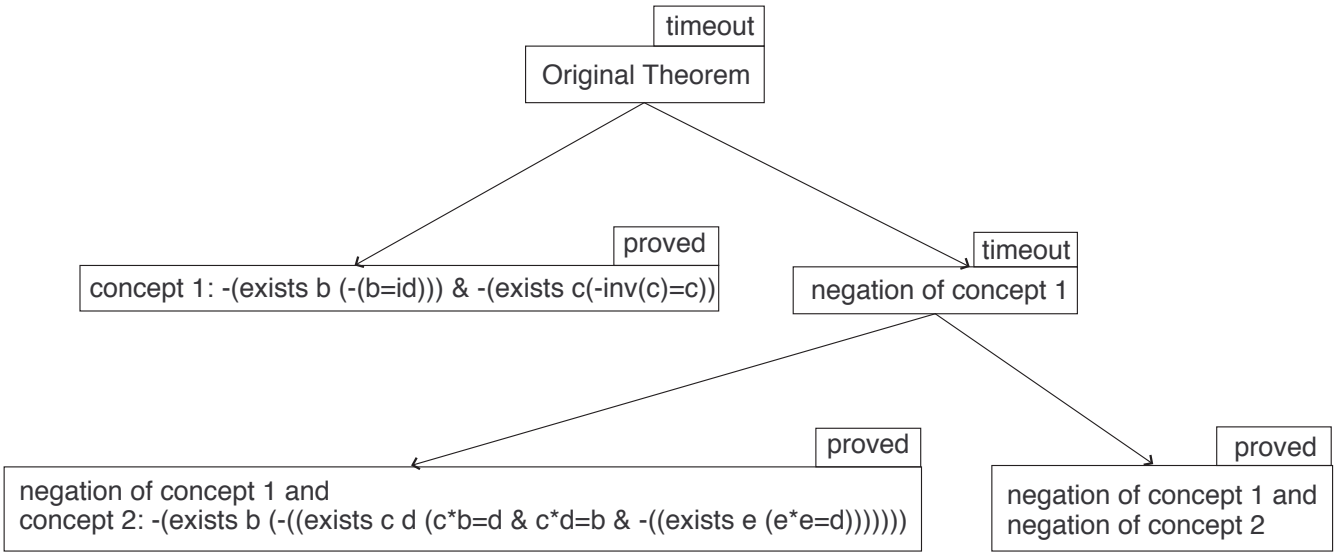

Figure 1: Case split tree for theorem GRP119-1.p

It seems apparent from our initial testing that the success of the case splitting procedure is dependent on the ordering of the concepts – as other orderings we tried were unsuccessful – and on the time Otter is allowed for each case (too little and it fails to prove easy cases, too much and it tends to spend too long on cases which ultimately do not lead to a solution). Interpreting the concepts in Figure 1 is non-trivial, but not difficult. We see that the first concept specialises the algebras into trivial and non-trivial, and the second concept states that every element x has a counterpart a for which $a*(a*x) = x$, yet $x * a$ does not appear on the diagonal. It is not clear why this latter concept enabled the case splitting to work where others failed. Note that a similar but different hand-ordering of the concepts available for case splits produced a proof with more case splits in around 30 seconds.

These preliminary results are encouraging because (a) there is a seven fold increase in efficiency (b) the number of case splits is small and the complexity of the concepts is low, and (c) two separate orderings gave significant speed-ups, which may mean that the procedure is more robust that we originally expected.

4 Future Work

Naturally, the next step is to automate the hand ordering of the concepts available for the case splits. We intend to try many different heuristics for this ordering, including (a) using the case splitting methods on many theorems from the TPTP library to estimate the expected speed increase of each concept, and using this to order them (b) choosing concepts which split the models into roughly equal positive and negative sets (c) choosing concepts which highly specialise the domain, e.g., concept 1 above. Also, at present, we are limited to binary case splits. In future, we plan to extend into general case splits, which will mean that we need to use Otter to prove that all models satisfy the disjunction of all the concepts. Moreover, there are many opportunities to use a first order predictive machine learning system such as the Progol ILP program [7]. In particular, using such a predictive learner would enable us to be more pro-active in the choice of case splits: we could choose which models should go in the positive/negatives of the concept, and then learn a definition to fit the models. We also plan much more experimentation to determine the effectiveness of this procedure in general. We hope to show, once more, that combined reasoning systems have advantages over their stand-alone counterparts.

References

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

[2] S Colton, A Meier, V Sorge, and R McCasland. Automatic generation of classification theorems for finite algebras. In *Proceedings of the International Joint Conference on Automated Reasoning*, pages 400–414, 2004.

[3] S Colton and I Miguel. Constraint generation via automated theory formation. In *Proceedings of CP-2001*, pages 575–579, 2001.

[4] S Colton and A Pease. The TM system for repairing non-theorems. In *Proceedings of the IJCAR'04 Disproving Workshop*, pages 13–26, 2004.

[5] I Lakatos. *Proofs and Refutations: The logic of mathematical discovery*. Cambridge University Press, 1976.

[6] W McCune. The OTTER user's guide. Technical Report ANL/90/9, Argonne National Laboratories, 1990.

[7] S Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.

[8] G Sutcliffe and C Suttner. The TPTP problem library: CNF release v1.2.1. *Journal of Automated Reasoning*, 21(2):177–203, 1998.

[9] L. Wos. *The Automation of Reasoning: An Experimenter's Notebook with OTTER Tutorial*. Academic Press, 1996.