Term Space Mapping for DISCOUNT

Stephan Schulz

Institut für Informatik
Technische Universität München
Arcisstrasse 20, D-80290 München, Germany
schulz@informatik.tu-muenchen.de

April 30, 1998

Abstract

We present a learning inference control heuristic implemented for the equational theorem prover DISCOUNT. Information about necessary and superfluous facts generated during successful proof searches (*positive* and *negative* examples) is compiled into a *term space map*, a recursive structure describing aspects of the topology of term sets. This map is then used to compute evaluations for facts in new proof searches. We present results of the learning system on a set of TPTP examples and demonstrate its usefulness.

1 Introduction

The last years have seen an increase in the use of formal methods for research and development. These methods are even beginning to make inroads into industrial use. In particular, theorem provers like DISCOUNT [DKS97] and SETHEO [LSBB92] are being used for the verification of protocols [Sch97] and the retrieval of software components [FS97] and mathematical theorems [DW97] from libraries. They are also integrated in interactive systems for the support of mathematicians [DGHW97].

A major requirement for the application of theorem provers is the performance in terms of proofs found within a practical time limit. While new calculi and refined implementations can improve the performance (as shown by the success of SPASS [WGR96] and Waldmeister [HBF96]), the proper use of search control knowledge offers at least the same potential for improvements.

Search control knowledge in most current theorem provers is represented by various, often parameterized heuristics for the selection of certain inference steps. Optimization of the theorem prover for a given domain consists in the selection of an existing heuristic (with suitable parameters), or even in the manual coding of a new heuristic based on the experience of a user with the domain. Both tasks are tedious, and expensive in terms of time and manpower[1]. Our aim is to adapt a theorem prover to a domain by capturing information about good search strategies from records of successful proof searches in *term space maps*.

In this paper we first give a short overview of DISCOUNT, the theorem prover used for our implementation. We then describe the general set-up of the learning theorem prover and the selection of training examples for the acquisition of search control knowledge. Sec. 3.3 describes term space maps and their application. Experimental results are presented next, and we conclude with a short evaluation and some remarks on future work.

2 The DISCOUNT System

DISCOUNT is a purely equational theorem prover, based on *unfailing completion* [HR87, BDP89]. It works on a set of equations (derived from the axioms) and an equational goal (the hypothesis). The basic algorithm repeatedly selects an unprocessed equation, rewrites it to normal form w.r.t. the already processed equations, and uses it to build critical pairs (new unprocessed equations). The equation is then used for interreduction and added to the set of processed equations. The process stops when either the goal has been proven or all equations have been processed[2].

The selection of the next equation to process is the most crucial decision for the efficiency of the proof procedure. It is controlled by one of multiple heuristics. Two of these heuristics (which are used for comparison later on) are *AddWeight*, which uses (weighted) symbol counting, and *Occnest* [DF94], a goal-oriented heuristic which prefers equations structurally similar to the goal. *AddWeight* is one of the strongest general purpose heuristics implemented in DISCOUNT. *Occnest* is not quite as successful in finding proofs, but performs better in certain domains and usually is much faster if it succeeds.

One of the strength of the DISCOUNT system is its ability to record proofs in the PCL format [DS94b, DS96b]. A variety of programs use these protocols for proof transformation, analysis and presentation. PCL protocols also serve as the basis for a variety of learning approaches, including the one presented in this paper. The basic setup of the DISCOUNT environment (for the sequential case) is shown in Fig. 1.

DISCOUNT was implemented in 1992 as a prototypical system for the evaluation of the teamwork method [Den93, Den95]. Its inference engine was fully adequate for this task, however, in raw throughput, it cannot compete with

[1]Attempts to automate these tasks at least partially are described in [Fuc95, Fuc97].

[2]For this work we ignore the ability of DISCOUNT to distribute deduction processes with team work, and we also do not make use of the ability to deal with non-ground goals.

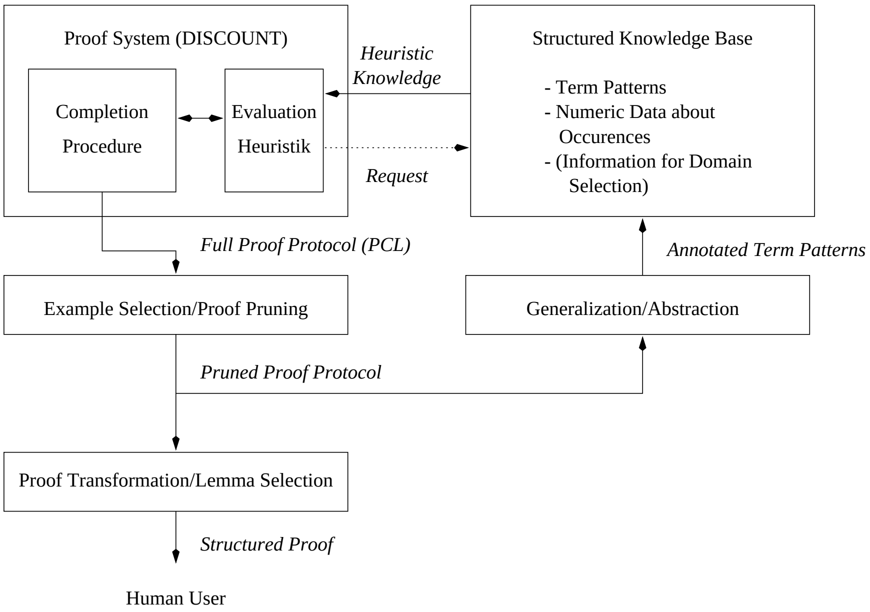

Figure 1: The DISCOUNT System

current provers as SPASS or Waldmeister. We found, however, that it gave us an excellent framework for adding and evaluating a variety of AI methods. Due to these improvements, DISCOUNT remains a viable and competitive theorem prover despite its (by now) rather weak inference engine.

3 Acquisition of Search Control Knowledge

Our approach to learning for DISCOUNT tries to extract search control knowledge from listings of inference steps. Our basic assumption is that the equations occurring in a successful proof search contain enough information to describe the proof adequately for reproduction, and that information about the exact structure of the proof is less important. This assumption is supported by the success of *learning by pattern memorization* [DS96a] particularly in reproducing proofs. It should also be kept in mind that much of the relevant structure of the proof is given implicitly by the calculus (inferences can only be performed after all the necessary preconditions are fulfilled).

Our learning algorithm is structured into three phases (again, compare Fig 1). First, a proof protocol of a successful proof search is analyzed, and positive and negative examples are selected from the equations generated in a proof search. In the second phase, the selected equations are generalized and stored in a knowledge base, annotated with some data about their role in the proof. Finally, the generalized equations from multiple proofs are compiled into a term space map, which is used in the main theorem prover to evaluate facts during the search for further proofs.

3.1 Example Selection

During an average proof search, a very large number of facts is generated, most of which do not contribute to the final proof. In typical non-trivial examples, between several 10000 and several 100000 inference steps are performed, only around 100-200 of which are usually needed for the proof (compare [DS94a] for a detailed discussion and [DS94b, DS96b] for some examples). This makes it impractical to use all these facts as training examples. We therefore use only a subset of equations *close* to the actual proof. The complete protocol of all inferences in a proof search is viewed as a directed graph, were the nodes are labeled by the equations generated during the proof and edges go from facts used in the premise of an inference to the equation generated by it. This graph is first pruned by removing all equations immediately proven redundant. In the remaining graph, equations contributing to the proof (those on a *proof path* from the axioms to the (proven) goal) are considered positive examples, while facts up to a certain number of edges removed from any proof path are used as negative examples. All other facts are discarded.

3.2 Annotated Term Patterns

In order to abstract from the particular signature used for a given example without resolving to a potentially expensive signature match, the equations are stored as generalized *representative patterns.* A representative pattern for a term t is computed by normalizing the variables in the term and substituting its function symbols in the way that the ith original function symbol of arity j occurring in t is replaced by the new symbol f_{ji}[3]. As an example, $\mathtt{pat}(f(x,a,a)) = f_{31}(x_1, f_{01}, f_{01})$. A representative pattern for an equation is computed by first orienting the equation according to some ordering stable with respect to the pattern transformation and then treating it as a single term.

To store additional properties of facts, we *annotate* term patterns. Annotations of term patterns carry e.g. information about the number of positive/negative occurrences of terms corresponding to a given pattern in a proof, or the distance of a fact from a proof path.

We represent a proof by a set of annotated patterns, corresponding to the facts selected as examples. These sets are stored in a knowledge base. Currently,

[3]This definition specializes the one given in [DS96a], ensuring that each new symbol is only used with one arity even in independently generated patterns.

we use a flat, unstructured knowledge base for term space mapping, that is, patterns from different proofs are not distinguished.

3.3 Term Space Maps

We use the annotated term patterns from multiple proofs to construct a *term space map* (TSM). A TSM is a recursive structure describing the distribution of certain properties among term nodes in a given set of terms (and thus describes some topological properties of the term set). It generalizes the concept of a *term evaluation tree* [DS96a]. More exactly, a term space map $T = \{A_1, \ldots, A_n\}$ is a set of *term space alternatives* (TSAs) such that any given function symbol occurs as the first element in at most one TSA (which represents the subset of terms with this top symbol). A TSA is a tuple $(f, (T_1, \ldots, T_n), a)$, where f is a function symbol of arity n, the T_i are TSMs corresponding to the principal argument positions of f, and a is an annotation accumulating the annotations of the terms in the term set corresponding to this TSA. Fig. 2 shows the graphical representation of a very simple example which uses natural numbers as annotations and accumulates them by addition.

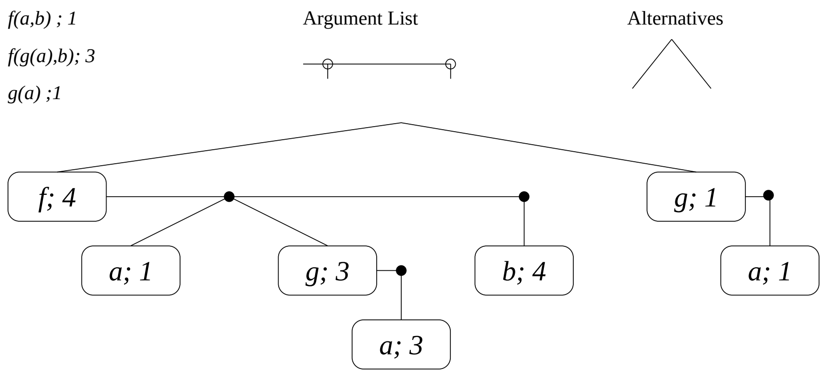

Figure 2: Example TSM for $\{(f(a,b);1),(f(g(a),b);3);(g(a);1)\}$

Even fairly large sets of example terms or equations can be compiled into the corresponding TSM efficiently[4]. A term can be mapped onto a TSM by first selecting the appropriate TSA (if it exists) and then mapping the arguments of the terms onto the corresponding sub-TSMs. This operation selects a number of TSM nodes, which can then be used to evaluate the term. For example, consider the TSM in Fig. 2 and use the sum of the annotations of nodes corresponding

[4]The knowledge base used in our experiments (Sec. 4) contains about 5000 term patterns, resulting from more than 40000 equations. It is loaded and compiled in about 1.5 seconds.

to a term as the evaluation of the term. Then the term $f(g(a), a)$ is evaluated as follows: $f(g(a), a)$ is mapped onto the left TSA, with the node $f;4$. The first argument, $g(a)$ is mapped onto the second TSA of the first argument TSM, i.e. the node labeled $g;3$. Similarly, the subterm a of $g(a)$ is mapped onto the node $a;3$. The second argument of the whole term, a, does not correspond to any TSA and remains unmapped (for a realistic evaluation function, we would assign a default weight here). The resulting evaluation of the whole term is $4 + 3 + 3 = 10$.

Our learning algorithm compiles the patterns of example equations into a TSM, using the number of positive and negative occurrences of an equation in a proof search as annotations. During the proof search, new equations are transformed into patterns and mapped onto the TSM. An evaluation for the equation is computed by assigning a bonus to nodes occurring primarily in positive examples and a penalty to nodes occurring primarily in negative examples, with a default evaluation for term nodes not represented in the TSM.

4 Experimental Results

Table 1 shows the performance of the two standard heuristics described above and the TSM-based heuristic in terms of problems solved in a given (per-problem) time limit. We used the set of all unit-equality problems with universally quantified goals from version 2.0.0 of the TPTP problem library [SSY94], i.e. all problems suitable for using ordinary unfailing completion. This set consist of 303 problems, 247 for which there are known proofs, 51 which should be provable, but no known proof exists, and 5 of entirely open status. Tests were performed on a SUN SPARCstation Ultra 2/200, all times given are cpu-times.

Time limit (sec.)	5	10	20	30	50	100	200	300
AddWeight	143	162	177	179	184	190	194	197
Occnest	155	155	168	171	174	183	187	188
TSM	161	173	186	189	199	204	207	207

Table 1: Comparative performance of different heuristics

The knowledge base contained proofs for 203 problems of the TPTP library. This set contains all problems that we were able to proof with DISCOUNT's other strategies. As the table shows, the learning heuristic performs better than both conventional ones at all time limits. Despite the fact that we used run time limits up to 1000 seconds and a large number of different strategies (including distributed teams) to find the proofs used in training, the single TSM-based strategy was able to proof all training problems in at most 160 seconds, with an average of only 6 seconds per problem. Moreover, the strategy was able to solve 4 problems previously unprovable by DISCOUNT despite all our efforts. It also solved 3 problems (LCL109-2, RNG009-5, and RNG009-7), for which no previous proofs had been recorded in the TPTP.

5 Conclusion

Our results once more confirm that patterns of equations contain enough information to describe equational proofs. Moreover, they show that term space maps still maintain enough of this information to allow a prover to efficiently reproduce the proofs and even allow for a generalization to new cases.

Our future work will concentrate on a couple of topics. First, we will work on improving TSM-related learning algorithms by aiming at stronger expressive power. Currently, TSM-based learning algorithms only allow us to express some rather simple concepts, namely distribution of certain function symbols and simple substructures in the training set.

We also found cases of over-training, i.e. degraded performance after adding additional examples, for most learning algorithms based on term space maps and term evaluation trees. We will address this problem by selecting suitable training examples (proof protocols) for a given proof problem, using both feature-based and TSM-based similarity measures. This will also support the use of more specialized heuristics for particular domains.

Another current focus of our work is the implementation of a more efficient inference engine capable of handling full clausal logic. This will allow us to evaluate our learning techniques for the more general case, and will hopefully also lead to an even more competitive prover.

References

[BDP89] L. Bachmair, N. Derschowitz, and D.A. Plaisted. Completion Without Failure. In *Coll. on the Resolution of Equations in Algebraic Structures, Austin, 1987*. Academic Press, 1989.

[Den93] J. Denzinger. *Teamwork: Eine Methode zum Entwurf verteilter, wissensbasierter Theorembeweiser*. PhD thesis, Fachbereich Informatik, Universität Kaiserslautern, 1993. (German Language).

[Den95] J. Denzinger. Knowledge-Based Distributed Search using Teamwork. In *Proc. of the ICMAS-95, San Francisco*, pages 81–88. AAAI Press, 1995.

[DF94] J. Denzinger and M. Fuchs. Goal Oriented Equational Theorem Proving Using Team Work. In *Proc. of the KI94, Saarbrücken*, number 861 in LNAI, pages 343–354. Springer, 1994.

[DGHW97] B.I. Dahn, J. Gehne, T. Honigmann, and A. Wolf. Integration of Automated and Interactive Theorem Proving in ILF. In W. McCune, editor, *Proc. of the 14th CADE, Townsville*, number 1249 in LNAI, pages 57–60. Springer, 1997.

[DKS97] J. Denzinger, M. Kronenburg, and S. Schulz. DISCOUNT: A Distributed and Learning Equational Prover. *Journal of Automated Reasoning*, 18(2):189–198, 1997. Special Issue on the CADE 13 ATP System Competition.

[DS94a] J. Denzinger and S. Schulz. Analysis and Representation of Equational Proofs Generated by a Distributed Completion Based Proof System. Seki-Report SR-94-05, Universität Kaiserslautern, 1994.

[DS94b] J. Denzinger and S. Schulz. Recording, Analyzing and Presenting Distributed Deduction Processes. In H. Hong, editor, *Proc. 1st PASCO, Hagenberg/Linz*, volume 5 of *Lecture Notes Series in Computing*, pages 114–123, Singapore, 1994. World Scientific Publishing.

[DS96a] J. Denzinger and S. Schulz. Learning Domain Knowledge to Improve Theorem Proving. In M.A. McRobbie and J.K. Slaney, editors, *Proc. of the 13th CADE, New Brunswick*, number 1104 in LNAI, pages 62–76. Springer, 1996.

[DS96b] J. Denzinger and S. Schulz. Recording and Analysing Knowledge-Based Distributed Deduction Processes. *Journal of Symbolic Computation*, 21:523–541, 1996.

[DW97] B.I. Dahn and C. Wernhard. First Order Proof Problems Extracted from an Article in the MIZAR Mathematical Library. In *Proceedings of the 1st FTP, Linz*, pages 58–62. RISC Linz, Austria, 1997.

[FS97] B. Fischer and J. Schumann. SETHEO Goes Software-Engineering: Application of ATP to Software Reuse. In W. McCune, editor, *Proc. of the 14th CADE, Townsville*, number 1249 in LNAI, pages 65–68. Springer, 1997.

[Fuc95] M. Fuchs. Learning Proof Heuristics by Adapting Parameters. In *Proc. of the 12th International Workshop on Machine Learning*, pages 235–243, San Mateo, CA, 1995. Morgan Kaufmann.

[Fuc97] M. Fuchs. Automatic Selection of Search-Guiding Heuristics. In D.D. Dankel II, editor, *Proc. of the 10th FLAIRS, Daytona Beach*, pages 1–5. Florida AI Research Society, 1997.

[HBF96] T. Hillenbrand, A. Buch, and R. Fettig. On Gaining Efficiency in Completion-Based Theorem Proving. In H. Ganzinger, editor, *Proc. of the 7th RTA, New Brunswick*, number 1103 in LNCS, pages 432–435. Springer, 1996.

[HR87] J. Hsiang and M. Rusinowitch. On Word Problems in Equational Theories. In *Proc. of the 14th ICALP, Karlsruhe*, number 267 in LNCS, pages 54–71. Springer, 1987.

[LSBB92] R. Letz, J. Schumann, S. Bayerl, and W. Bibel. SETHEO: A High-Performance Theorem Prover. *Journal of Automated Reasoning*, 1(8):183–212, 1992.

[Sch97] J. Schumann. Automatic Verification of Cryptographic Protocols with SETHEO. In W. McCune, editor, *Proc. of the 14th CADE, Townsville*, number 1249 in LNAI, pages 87–11. Springer, 1997.

[SSY94] G. Sutcliffe, C.B. Suttner, and T. Yemenis. The TPTP Problem Library. In *Proc. of the 12th CADE, Nancy*, number 814 in LNAI, pages 252–266. Springer, 1994.

[WGR96] C. Weidenbach, B. Gaede, and G. Rock. SPASS & FLOTTER Version 0.42. In M.A. McRobbie and J.K. Slaney, editors, *Proc. of the 13th CADE, New Brunswick*, number 1104 in LNAI, pages 141–145. Springer, 1996.