

## Strategy Parallelism and Lemma Evaluation

Joachim Draeger & Andreas Wolf

Institut für Informatik der Technischen Universität München  
D-80290 München  
Germany  
{draeger,wolfa}@informatik.tu-muenchen.de

### Abstract

Automated Deduction offers no unique strategy which is uniformly successful on all problems. Hence a parallel combination of several different strategies increases the chances of success. The efficiency of this approach can be increased even more by the exchange of suitable intermediate results. The paper offers a cooperative solution approach to this task. As a special kind of cooperation we present here the selection of suitable lemmata together with a model of a cooperative parallel theorem prover which combines different lemma selection techniques within a strategy parallel prover environment. We give a short assessment of the results of first experiments and an outline of the future work.

### Introduction

Up to now, sequential automated theorem provers (ATPs) have set a high standard; but when dealing with difficult problems they are still inferior to a human mathematician. An important technique to increase the performance is to employ parallelism. Possible parallelization concepts vary from the parallel use of different configurations of the employed provers to the partitioning of the proof task into subtasks that are tackled in parallel. Another promising technique is the use of lemmata for reducing the search space, which has to be processed for obtaining a solution. The serious drawback of a lemmatization is the introduced additional redundancy originating from the fact that during the ongoing search both lemma applications and lemma reproductions are possible. The amount of redundancy increases rapidly with the number of lemmata added to the original axiomatization. Hence, the lemmata supporting the proof process have to be selected very carefully.

The parallelism concept can be improved by interactions between the different parallel provers (cooperation) by exchanging intermediate results like lemmata. The advantages of such a combination of parallelism and lemmatization are twofold. At first, the exchange

of intermediate results will reduce the amount of redundancy contained in the parallel computations. Secondly, if the different strategies include different lemma selection schemes, the lemma selection treatment will profit, too. In many cases the specific effects of different lemma selection strategies are not known in advance; hence, a reliable assessment of these strategies requires practical experiences.

As we have seen, both parallelism and lemmatization can profit from the combination of several lemma selection strategies in a competitive manner. Thus our aim will be the realization of such a combination. For the evaluation of a combination of different lemma selection strategies, we will restrict our considerations to the model elimination calculus (Loveland 1968), which is based on problem decomposition. A given query is recursively decomposed into new sub-queries until each of them can be solved either by a unit-lemma of the axiomatization, or by one of the assumptions made during the decomposition. In this way possible decompositions are enumerated until a proof can be constructed.

The paper is organized as follows. In the next section we give a selection of related work. Then follow three sections dealing with our main topics: lemma generation and evaluation, lemma selection, and combination of lemma generation techniques. We conclude with a short assessment of first experimental results and with an outlook.

### Related Work

In the past years there have been several approaches for cooperation among bottom-up theorem provers. Some of them have been based on superposition, unifying completion, or resolution. Methods which partition the clauses among several sub-provers are used in DARES (Conry *et al.* 1990). The Teamwork (Denzinger 1995) method of DISCOUNT provides cooperation among complete competitive provers by periodically exchanging evaluated and selected results. Here, we also find a similarity based deduction guidance similar to our lemma similarity strategy. An approach for lemma generation and application by sequential cooperation among a saturating component and a top-down prover has been developed in DELTA (Schumann 1994).

## Lemma Evaluation

The generation and use of lemmata is a special kind of learning on the domain level. Their application yields a result while avoiding the search needed for their deduction. Hence, its effect is a reordering of the search space, supporting some solution constructions while limiting others. In the introduction this was explained by the ability of lemmata to guide the search as intermediate results. Now we discuss this effect in a more formal way. By separating parts of an original proof  $p$  as lemmata a modularization of both the proof and the search process is achieved. Technically a simple version of such a modularization can be realized as a procedure which generates unit-lemmata  $f_1, f_2, \dots$  and uses them for constructing the proof  $p'$  of the actual problem. The use of unit-lemmata  $f_1, f_2, \dots$  makes the proof  $p'$  smaller and hence easier to find than  $p$ . If the lemmata  $f_1, f_2, \dots$  are 'useful' with respect to the modularization of the actual proof task, the desired restriction of the search space is performed.

The above considerations yield a more abstract point of view. Lemmata can be considered as pieces of knowledge which are starting points for future exploration of the search space. If these starting points are chosen in a suitable way, e.g., if they represent intermediate results of the proof  $p$ , the necessary effort for finding a solution can be reduced dramatically.

Which lemmata can be considered to be useful? The lemmatization has to reduce the necessary effort for finding a proof by modularization as described before. Consequently, a lemma may only be considered as useful if it enables a separation of a *significant part* of the original proof. This is only possible, if the lemma itself requires a significantly complex proof. A suitable way to measure the proof complexity of a lemma  $f$  is the *proof length*  $p(f)$ , the number of inferences contained in the proof of  $f$ . Unfortunately, the proof length is no invariant. For example, the insertion of equivalence transformations into the proof can enlarge  $p(f)$  without limit making this parameter ill-defined. However, it is fair to say that additional inferences blowing up a proof are irrelevant. This leads to the use of the *minimal proof length*. In the following, we always use the term 'proof length' for the minimal proof length. A comparatively large value of this parameter will be our first selection criterion.

The proof length alone is not sufficient for an efficient lemma selection; in most cases an overwhelming number of lemmata requiring non-trivial proofs exist. Hence an additional selection criterion is needed. It can be based on the observation that the *potential* of separating a significant part of *some* proofs is not sufficient; the separation must actually *happen* in a proof of the *actual* problem. Consequently, we choose the *relevance*  $r(f)$  of a lemma  $f$  with respect to the actual proof task as second selection criterion. Contrary to the proof complexity, there is no obvious way for measuring the relevance of a lemma. Consequently, many

different methods for the relevance estimation are possible. Each one leads to a specific selection strategy. In this paper, we will discuss the following two main strategies.

**Strategy 1. Lemma Size.** In this strategy we assume that a lemma  $f$  with low syntactic complexity is more relevant than a lemma  $f$  with a high syntactic complexity. One can argue in the following way. The proof of a problem (if there is one) is finite; hence the number of useful lemmata is finite, too. On the other hand, the total number of lemmata, which are valid in the underlying theory, is typically not finite or exceeds at least the number of useful lemmata by far. In other words, the syntactic complexity of an arbitrary lemma is much larger than the syntactic complexity of a useful lemma on the average. Therefore limiting the syntactic complexity of a selected lemma will raise the probability that this lemma is useful for the construction of the actual proof.

The measurement of the syntactic complexity of a unit-lemma  $f$  can be performed in different ways. In this paper, we use the *symbol size*  $s(f)$ , i.e. the number of constant and function symbols contained in the assertion of  $f$ , and the *symbol depth*  $d(f)$ , i.e. the length of the longest path in the assertion of  $f$  represented as symbol tree. Both criteria symbol size  $s(f)$  and symbol depth  $d(f)$  are used as reciprocal values, i.e.  $r(f) = (s(f))^{-1}$  and  $r(f) = (d(f))^{-1}$ , respectively.

**Strategy 2. Lemma Similarity.** The second strategy principle is the relevance measurement based on the similarity of a considered unit-lemma  $f$  to the query  $q$ . The main idea of this strategy is the identification of lemmata, which are useful with respect to a step-by-step construction of the query  $q$ . Following this idea it is suggestive to consider a unit-lemma  $f$  to be the more useful, the stronger the similarity of  $f$  and  $q$  is. Hence we will demand a strong similarity of  $f$  and  $q$  in order to make the relevance of  $f$  for the construction of the desired proof as large as possible. This should also enable the prover system aiming at the query more explicitly.

For measuring the similarity of two literals  $a$  and  $b$  we use two methods. The first one measures the *structural similarity*  $u_q(f)$ , the second one the *signature similarity*  $g_q(f)$ . Let  $w_1, \dots, w_n$  be the maximal sub-terms contained both in  $f$  and  $q$ . The function  $u_q(f)$  is defined to be

$$u_q(f) = s(q) + s(f) - 2 \cdot s(w_1) - \dots - 2 \cdot s(w_n).$$

Similarly, the function  $g_q(f)$  counts the numbers  $n_q(a_i), n_f(a_i)$  of occurrences of each function, constant, and predicate symbol  $a_1, \dots, a_m$  contained in  $q$  and  $f$ . The value of  $g_q(f)$  is determined by

$$g_q(f) = |n_q(a_1) - n_f(a_1)| + \dots + |n_q(a_m) - n_f(a_m)|.$$





