Referees for Teamwork

Jörg Denzinger, Dirk Fuchs
Department of Computer Science
University of Kaiserslautern
Postfach 3049, 67653 Kaiserslautern
E-mail: {denzinge,dfuchs}@informatik.uni-kl.de

Abstract

Teamwork is a method to distribute automated theorem proving and is mainly based on the competition and cooperation of different search control heuristics. Together with the usual control heuristics that try to anticipate whether an inference step and the resulting fact is good, teamwork also employs assessment heuristics that judge the impact a fact has had on the search after a while. For this assessment teamwork uses so-called referees that allow it to throw away facts that did not meet the expectations. In this paper we discuss several referee concepts and compare them by experiments.

1 Introduction

The main goal when developing search strategies for automated theorem provers is to do as few unnecessary inference steps as possible. Unfortunately, experience has shown us that automated theorem provers do many more unnecessary steps than steps that are needed for a proof. Since unnecessary steps tend to produce more unnecessary steps, a research goal with high priority is to eliminate or forget unnecessary inference steps.

Research concerning this problem is aimed in two directions. The first direction is to reduce the applicability of inference rules that generate new facts while developing other rules that simplify or remove existing facts. A good example of this direction are the works of Bachmair and Ganzinger on ordered superposition and on general redundancy criteria (see [BG90]). The second direction is to develop criteria to throw away facts without any justification by inference rules, what we call *forgetting* of facts. An example of results of this direction is the limitation of the size of generated facts that can be found, for example, in the Otter prover (see [Mc94]). Unfortunatly almost every usable criterion results in losing the completeness of the prover. Furthermore, most of the criteria developed so far are very crude, which means, for example, that they do not take the problem one wants to prove into account.

With our teamwork method (see [AD93], [De95]) we developed a knowledge-based distribution method for automated theorem provers that concentrated on both search strategies to generate new facts and assessment heuristics to forget facts in order to avoid an explosion of the search space. The general idea of teamwork is to let several different search heuristics (so-called experts) work on the given problem in parallel and independently. Periodically referees judge the experts and select outstanding new facts from them. These facts are added to the set of facts of the best rated expert by a supervisor. Since the resulting search state is the new start state of all experts, all the experts' non-selected facts (without those of the best expert) are forgotten. The supervisor also uses the judgement by the referees to exchange experts for better suited ones, thus allowing for an adaptation of the system to the given problem.

After reports concerning special search heuristics ([DF94]) and the planning task of the supervisor when selecting the team for the next round ([DK94]) we will concentrate in this report on the referees. Obviously, the referees allow a synthesis of different search heuristics and strategies for forgetting facts. By selecting facts that have proven to be "good" (which has to be defined by the referees) and thus forgetting useless facts, a very sophisticated concept for deleting facts –that takes the problem to prove into account– is realized. This is enhanced by the competition of different search heuristics which is a result of the teamwork concept. In the following we will demonstrate several ways to design referees and compare the different referees by giving some examples.

0

2 Automated theorem proving and completion

Due to lack of space we can only give a very brief introduction to automated theorem proving, equational theorem proving and the completion method. For more details we refer to [CL73], [HR87] and [AD93].

Theorem proving means solving the following problem :

Given: A set A of axioms and a theorem T to prove.

Question: Is T a logical consequence of A ?

In equational proving A = $\{s_i = t_i |\ i=1,...,n\}$ is a set of (all-quantified) equations and T is an equation u = v, too.

All successful methods for automated theorem proving, if equality is involved, are based on two kinds of inference rules: generation rules and contraction rules. The generation inference rules add new facts to the data base. These facts are derived either from the axioms alone (as in the case of equational theorem proving by completion) or from both the axioms and the theorem T (as in the case of resolution and paramodulation). The contraction inference rules change or delete facts from the data base.

For equational reasoning, the area we applied teamwork to, the generation inference rule is the critical pair generation and the contraction inference rule is reduction (also known as demodulation). According to a so-called reduction ordering the equations are oriented to rules (if possible). A critical pair to two rules $l_1 \rightarrow r_1$ and $l_2 \rightarrow r_2$ is the equation $\sigma(l_1)[p \leftarrow \sigma(r_2)] = \sigma(r_1)$, where σ is the mgu of l_2 and the subterm of l_1 at position p (which mustn't be a variable) and where $\sigma(l_1)[p \leftarrow \sigma(r_2)]$ denotes the exchange of the subterm at position p in $\sigma(l_1)$ by $\sigma(r_2)$. Rules are also used to reduce terms. If there is a match μ from the left hand side of a rule $l \rightarrow r$ to a subterm of term t (at position p), then this subterm is replaced by $\mu(r)$ (written $t \Rightarrow t[p \leftarrow \mu(r)]$. An equation or rule is in normal form, if no reduction with any rule of the data base is possible.

From a theoretical point of view we need for an automated theorem prover, besides the inference rules, fairness criteria for the use of the inference rules. These criteria guarantee that each application of an inference rule that is enabled infinitely often will finally be executed. But, for challenging examples there are many possible inferences and a systematic application results both in enormous run times and the need of much (memory) space. It is not uncommon for a prover to require an agenda of 500,000 to 1,000,000 inference rule applications.

Therefore, implementations of automated theorem provers use strategies and heuristics to select the next inference rule to apply and the facts these rules should work on. A strategy guarantees theoretical completeness, whereas for heuristics it is possible that the prover will not find a proof even if there is one and enough time and space are provided. Heuristics are very important, because very often only the use of a heuristic allows finding a proof. The same can be said about simply forgetting facts that do not seem to be necessary. This allows a reduction of the search space but there is the danger that necessary information for finding a proof is deleted. Many theorem provers allow the user to choose between various strategies and heuristics. But two problems still remain. First, for a given problem there may be no appropriate strategy or heuristic implemented in the system. Second, even if there is a good one implemented, how does the user know which one is good? We tackled both problems with our teamwork method as we will demonstrate in the next section.

3 The teamwork method

The main intention of the teamwork method is to achieve cooperation between different strategies for automated theorem provers while using competition to direct the whole system in the most promising direction. The main instruments for achieving these goals are techniques from the areas of distributed systems, multi-agent systems and knowledge-based systems together with control strategies, heuristics and techniques from automated deduction.

A system based on teamwork consists of four types of components: experts, specialists, referees and a supervisor. Experts and some specialists are those components working on solving a given problem. Some other specialists, the referees and the supervisor control the team and are responsible for achieving cooperation and competition.

Experts are automated theorem provers. They differ either in the inference rules they use or in the control strategies or heuristics they employ (mainly the control of the generation rules). All known general control strategies and heuristics can be used, but –since they work in a team– also fragments of strategies or even specialized ideas. All experts use a common representation of their search state. So, each of the experts can continue its work using the state of another expert. During the so-called working period all experts work independently on the given problem.

In the case of equational theorem proving by completion the experts use different selection methods for critical pairs. Besides standard criteria, like the number of symbols in the pair or using functional interpretations of the symbols over $\mathbb{N}$, we also have developed criteria that define similarities between a critical pair and the goal (see [DF94]). These criteria are examples of specialized heuristics that can only solve very few examples when working alone but are very useful in cooperation with other experts.

Specialists can either be used to assist in finding the solution to the given problem or to help the supervisor in its controlling task. Some specialists can even contribute to both tasks. One of these specialists, the so-called domain detection specialist, will be described together with the supervisor. In contrast to experts, specialists needn't use the common search state representation. Specialists also work independently of each other and the experts during the working periods.

Referees are responsible for comparing experts, specialists and their results and therefore represent the competition aspect of teamwork. After a working period each expert and each specialist is judged by a referee. A referee computes for its expert a *measure of progress* and it selects the best facts generated by the expert during the last working period. A referee for a specialist only selects good facts of the specialist, provided that the specialist has generated any facts. Other information computed by a specialist, as for example the domain of the problem to prove, is passed on to the supervisor unjudged. In section 4 we will describe in more detail possible criteria that can be used by referees to fulfill their tasks. The measure of progress of each expert is transmitted to the supervisor. Then, after the best expert is determined, the experts' and specialists' selected facts are reported to the supervisor.

The *supervisor* is responsible for strategic decisions in the team and therefore for the adaptation of the team to the given problem. In addition, the supervisor also achieves the cooperation of the team members by generating a new start state for the next working period. In detail, the supervisor performs the following actions: First, it receives the measure of progress of each expert and the information from the specialists. Then the supervisor determines the best expert, i.e. the expert with the highest measure. After that it receives the selected facts of all other experts and the specialists and adds these facts to the search state of the best expert resulting in a new start state. Then the supervisor selects the members of the next team based on a long-term memory (consisting of information about domains of interest, good teams and good and bad expert combinations) and on a short-term memory (consisting of the information gathered by the specialists and the measure of progress of each expert for each working period so far). After transmitting the new start state to all members of the next team the supervisor also determines the length of the next working period and sends this information to the team members. Then the work of the supervisor is done and a new working period begins.

An important help for determining the new team members are the results of the *domain detection specialist*. A domain is described by a set of facts. With a domain are associated known consequences of the facts (which are the delivered to the referee of the specialist in case the domain is detected), a plan skeleton, hierarchical information about the domain and known good single experts and referees for this domain. The supervisor uses the plan skeleton as basis for its decision about the new team, but modifies this plan in reaction to the measures of the experts. For more about the planning task of the supervisor we refer to [DK94].

The teamwork method allows an efficient implementation with a very effective communication between processors. By representing experts/specialists, referees and supervisor as one process with different modi (supervisor modus always given to the processor running the best expert), the only interprocessor communication that is necessary are the reports of the referees to the supervisor and the transmission of the new start state. Since this transmission uses broadcast and allows for filtering the data on the side of the sender (eliminating unnecessary data) and ordering the data on the side of the receivers (typically using different orderings due to the different selection criteria), there is no big overhead due to communication.

Our experiments have shown that teamwork was indeed able to achieve cooperation between competiting strategies resulting in synergetic speed-ups and even in the ability to solve much more examples than all experts working alone (see [DF94]). Teamwork also provides a higher level of automatization than most other provers, which is due to the planning and adaptation ability of the supervisor (see [DK94]). Finally, we also have shown that teamwork can successfully be applied to other search problems, for example optimization problems. For the traveling salesman problem the same synergetic effects as for theorem proving can be achieved (see [De95]).

4 Referee concepts

Referees have two tasks, namely judging the whole work of an expert of a working period and selecting outstanding results of an expert. Therefore a referee consists of two functions, a *judge-expert* function and a *select-good-results* function. Select-good-results functions compute for each new fact a measure. Then the best n facts will be selected by the referee, provided their measure exceeds a given threshold. Since these measures can also be used to judge an expert (by, for example, using the mean value of all new facts as measure), in the following we will mainly concentrate on select-good-results functions and we only provide a short section about statistical criteria for judge-expert functions that have proven to be quite useful.

4.1 Judging an expert

In order to judge the complete work of an expert in a working period statistical criteria are the first obvious choice. And our experiments showed that these criteria are sufficient to allow successful proof runs with teamwork based systems. There is much statistical data that can be used in computing a measure of progress for an expert. In our experiments that will be described in section 5 we used only the judge-expert function *relational* that has two parameters, *red_factor* and *cp_factor*, and combines the number of critical pairs |CP|, the number of reductions made |Red|, the number of rules |R| and the number of equations |E|. The value of *relational* is computed as

$$red\_factor * |Red| - \frac{|CP| * cp\_factor}{|R| + |E|}.$$

The idea of *relational* is that many reductions obviously are a good sign, while many critical pairs mean more difficulties in selecting the right ones. But, if there are many rules and equations, then many critical pairs have to be expected while many critical pairs out of only a few rules and equations clearly indicate that experts with a better (i.e. lower) ratio should be preferred. We found it useful to start proof runs with referees using *relational* with a parameter combination, where *red_factor* >> *cp_factor* and then employing in later periods referees with growing *cp_factor*.

4.2 Select-good-results functions

Measuring facts is a task of both, experts and referees. On the one hand this makes finding criteria that can be used by select-good-results functions of referees easier, because the criteria of experts can be used as a starting point. On the other hand the question arises whether referees are different from experts at all and therefore possibly not necessary.

Experts try to measure the impact of a fact (or an inference) on the problem at hand. Since experts do not want to measure this impact by actually doing the inference leading to the fact with all consequences that are involved –this would be much too inefficient– they have to do a kind of guessing, which can be partially or even totally wrong.

On the other hand, referees measure facts after their impact on a search process has become known. They can observe what inferences resulted from the addition of a fact. The impact of a new fact is also measured by the number of facts that could be thrown away, because they have become redundant. Since facts selected by a referee are added to the search state of the best expert, a further criterion that should be taken into account by a referee is the probability that a fact enhances this search state. This means that selected facts should produce inferences that will be given a high priority by the winning expert, without treating them in a special way. So, referees offer a new dimension and are indeed necessary.

In the following we will present three concepts for developing select-good-results functions that have different priorities concerning the goals given above.

Last(number_of_facts)

The select-good-results function *Last* is quite simple, it selects the last $number\_of\_facts$ facts produced by the expert that were not redundant. The idea behind Last is that several experts that were not the best one may be members of the next team again. These experts should be able to continue their work without repeating most of the inferences they have done in the last working period. By including their last results into the new start state there is a high probability that they will concentrate on these results in future.

As we will see in section 5, this simple function can be used for several examples with some success. But for other examples the idea is just too simple.

Statistic(number_of_facts, crit_pair_penalty, reduction_bonus, threshold)

The aim of the select-good-results function *Statistic* is to measure the impact a fact has had on the search of an expert. The impact is observed by the number of critical pairs generated by a fact (weighted by *crit_pair_penalty*) and the number of reductions that have been made using the fact (weighted by *reduction_bonus*). Different reductions are counted differently when computing the number of reductions: Reductions of a goal count more (typically 100 times) than the reduction of a critical pair. Also the reduction of a rule or an equation counts more (typically 5 times). Statistic computes for each fact a weighted sum of the number of critical pairs and the number of reductions and selects the best *number_of_facts* facts (i.e. the facts with the highest sums), provided they are better rated as *threshold*.

As the names of the weights suggest, we intended to see many reductions as a bonus while the generation of many critical pairs should be given a penalty (i.e. *crit_pair_penalty* is negative). But it may be necessary to vary penalty and bonus during a proof run. At the beginning of a proof run the generation of many critical pairs should not be penalized, because a broad base of facts is needed. Therefore the number of reductions alone (i.e. *crit_pair_penalty*=0) or even a positive penalty for critical pairs can be used. Later on, using a positive value for weighting the number of critical pairs often becomes disastrous.

Nevertheless, Statistic has proven to be quite successful. For some examples an adjustment of the parameters allows for quite some improvements. Therefore there are some possibilities for the use of further knowledge for parameter adjustment. But there are standard parameters that result in a satisfactory performance.

Winner_Driven(number_of_facts, threshold)

The aim of *Winner_Driven* is to select results that have a high probability to help the best expert of the working period, the winner. The idea behind Winner_Driven is to use the selection function for critical pairs of the best expert to measure the facts of some of the other experts. Since the selection of the best facts can be delayed until the winner is determined, this idea can easily be implemented.

Provided that the expert whose best facts have to be selected differs very much from the best expert the following effect occurs: A heuristic for traversing a search space can be seen as following a kind of valley through a mountainous region. The mountains are those paths through the search space that have higher measures than the path that is actually taken. But behind the mountains may be a much deeper valley than the actual followed one that is not reached because of the mountain barrier (in our case facts that do not have a good measure but which produce other facts that would be measured good). Paths that are mountains for one expert may be valleys for another one. Therefore an expert may, after some time, produce facts that are very good according to the heuristic used by another expert but are also impossible to reach using this heuristic. Such facts are selected by Winner_Driven.

As we will see in the next section, Winner_Driven is a very generally usable select-good-results function. It is very stable under variation of the parameters (although an astronomically high *threshold* can greatly influence a proof run). A rip-off of the idea of Winner_Driven is to use not the function of the winner but always goal similarity functions (see [DF94]). This rip-off can also be very successful.

5 Experiments

In this section we will compare the select-good-results functions presented in section 4.2. For this purpose we have chosen several examples for which teamwork has been quite successful. These examples are taken from [ADF95] and [DK94]. We omitted those examples for which not the exchange of results caused the improvements but the change of the selection heuristic for critical pairs that is a result of choosing the best rated expert as basis for the new start state of a cycle.

Because Statistic has some parameters that can be adjusted to an example, we devote Statistic two columns in Table 1, one stating the run times with standard parameter values (St.st.) and another one stating the times for the best values found (St.b.). In each row of the table the same lengths of working periods and (of course) the same expert combinations were used, so that only the select-good-results functions of the referees had changed. In order to give a hint at what the gain provided by teamwork is, we give in the last column the run times of the best known experts for the examples, provided we have been able to find an expert that is capable of solving them.

Ex.	**St.st.**	**St.b.**	**Wi._Dr.**	**Last**	**b.exp.**
bool5b	75.2	70.0	57.9	—	—
sa2	28.2	16.8	10.7	15.1	—
ra2	41.4	41.4	42.3	49.7	230.0
lusk6	460.6	312.2	341.0	495.8	3019.0
cal3	87.5	79.8	81.9	81.7	297.2
cal4	275.8	171.0	111.7	96.5	—
p2.a	6.2	6.2	5.7	6.9	79.5
p8.b	40.6	39.4	40.6	97.4	—
p10	26.6	26.4	23.0	—	—

Table 1: Comparison of different select-good-results functions on 2 SUN-ELC, run times in sec.

The examples cover a wide range of domains: bool5b boolean algebra, sa2 groups, ra2 robbins algebra, lusk6 rings, cal3 and cal4 propositional calculus and p2.a p8.b and p10 lattice ordered groups.

The first and most important observation that can be made in Table 1 is that the two more "intelligent" select-good-results functions Statistic and Winner_Driven enable our system to solve all examples even those that couldn't be proved in a sequential run. The importance of cooperation as result of these functions is emphasized by examples bool5b and p10 that could not be solved by using Last, thus indicating that good cooperation is necessary to solve them. But Last has its merrits, as can be seen in example cal4.

If we take a closer look at the results of Winner_Driven and Statistic, we can observe that for 7 examples Winner_Driven is better (i.e. faster) than the standard parameter setting of Statistic. And for the remaining 2 examples it is (nearly) as good as the standard version. With respect to the reduction of support by the user this is a very good sign, since Winner_Driven doesn't have many parameters that have to be set. On the other hand there are 4 examples for which a little fiddling with the parameters of Statistic produced better run times than those using Winner_Driven. So there is definitely room for some improvements. These improvements can either be a combination of Winner_Driven and Statistic or a planned use of specialized Statistic referees using the concept developed for experts in [DK94].

As a consequence of our experiments so far we use the standard version of Statistic for all experts in teams that do not include goal-oriented experts. Teams that use goal-oriented experts use Winner_Driven.

6 Conclusion

A very crucial part of teamwork based systems are the referees. Only good referees allow for synergetic effects and the forgetting of useless results, features that are among the great advantages of teamwork. After presenting the main ideas for experts and the supervisor in other papers we concentrated in this paper on the referees. Since a good result is not determined by a good inference that generated this result but by the consequences a result has for a proof attempt, concepts for select-good-results functions of referees are not only important for teamwork, but for each theorem prover.

In this paper we presented several simple concepts for such selection functions that have proven to be successful. The use of statistical data offers both a good starting point and adjustment to particular examples, while expert oriented selection functions offer robustness and stability. Even selecting the newest facts can be useful. So statistical criteria are a basis that should be strengthened by including other, knowledge-based criteria. One possibility that should be investigated in the future is to combine Statistic with Winner_Driven to achieve this strengthened basis.

The development of more concepts for referees is also important because of the growing number of distribution and cooperation approaches for automated theorem proving. In both areas the central problem is the huge amount of communication that real systems have to deal with. Since communication is quite expensive a major concern should be to avoid communicating unnecessary results, which can be achieved by using referee-like concepts. Even if one is purely interested in search control heuristics, forgetting of facts remain a necessity.

References

[AD93] **Avenhaus, J. ; Denzinger, J.**: *Distributing equational theorem proving*, Proc. 5th RTA, Montreal, LNCS 690, 1993, pp. 62-76.

[ADF95] **Avenhaus, J. ; Denzinger, J. ; Fuchs, M.**: *DISCOUNT: A system for distributed equational deduction*, Proc. 6th RTA, Kaiserslautern, LNCS 914, 1995, pp. 397-402.

[BG90] **Bachmair, L.; Ganzinger, H.**: *On restrictions of ordered paramodulation with simplification*, Proc. 10th CADE, Kaiserslautern, Springer, LNCS 449, pp. 427-441.

[CL73] **Chang, C.L.; Lee, R.C.T.**: *Symbolic Logik and Mechanical Theorem Proving*, Academic Press, 1973.

[DF94] **Denzinger, J. ; Fuchs, M.**: *Goal oriented equational theorem proving using teamwork*, Proc. 18th KI-94, Saarbrücken, LNAI 861, 1994, pp. 343-354.

[DK94] **Denzinger, J. ; Kronenburg, M.**: *Planning for distributed theorem proving: The team work approach*, SEKI-Report SR-94-09, University of Kaiserslautern, 1994.

[De95] **Denzinger, J.**: *Knowledge-Based Distributed Search Using Teamwork*, Proc. ICMAS-95, San Francisco, AAAI Press, 1995, pp. 81-88.

[HR87] **Hsiang, J.; Rusinowitch, M.**: *On word problems in equational theories*, Proc. 14th ICALP, Karlsruhe, LNCS 267, 1987, pp. 54-71.

[Mc94] **McCune, W.W.**: *OTTER 3.0 Reference manual and Guide,* Tech. Rep. ANL-94/6, Argonne National Laboratory, 1994.