Inference Rights for Controlling Search in Generating Theorem Provers

Dirk Fuchs*
Fachbereich Informatik, Universität Kaiserslautern
Postfach 3049
67653 Kaiserslautern
Germany
E-mail: dfuchs@informatik.uni-kl.de

December 20, 1996

Abstract

We investigate the usage of so-called inference rights. We point out the problems arising from the inflexibility of existing approaches to heuristically control the search of automated deduction systems, and we propose the application of inference rights that are well-suited for controlling the search more flexibly. Moreover, inference rights allow for a mechanism of "partial forgetting" of facts that is not realizable in the most controlling aproaches. We study theoretical foundations of inference rights as well as the integration of inference rights into already existing inference systems. Furthermore, we present possibilities to control such modified inference systems in order to gain efficiency. Finally, we report on experimental results obtained in the area of condensed detachment.

*The author was supported by the *Deutsche Forschungsgemeinschaft (DFG)*.

1 Introduction

The original task of a theorem prover is to check efficiently if a goal is a logic consequence of a given set of axioms. If we want to use generating calculi (e.g. resolution, unfailing completion) to fulfill this task the usual proceeding is to modify the given start state of facts (the axioms and the goal) with rules for generation and deletion of facts until a designated end state is reached and hence the goal is proven. A common way to control the application of the rules of a calculus is to use heuristic methods. Experiments in several calculi (e.g. condensed detachment, unfailing completion, resolution) have shown that heuristic control of the search is a viable approach (see, e.g., [5]).

In spite of this fact some problems still remain: unfortunately experience tells us that even automated theorem provers with good heuristics do many more unnecessary steps than are needed for a proof. Performing unnecessary inference steps in order to delete or to manipulate facts increases the run time a little bit, but does not lead to a dramatic change of the proof run. The generation of useless facts, however, can entail aggravating consequences. Since usually it cannot be tested whether facts are really needed in order to prove the goal such facts persist throughout the search. Because of the fact that these generated facts can take part in further inferences more unnecessary facts can and will be generated. This way the number of facts that are not needed for a proof grows dramatically and often makes it impossible for the prover to prove the goal within given limits of time or space.

One method that is—in our opinion—well-suited to deal with the mentioned problems is the integration of so-called *inference rights* into existing inference systems. Thus, certain inferences can only take place if all facts involved in it have the right to perform it. By utilizing inference rights it is possible to refine the common saturation strategies and to control inferences more flexibly. Especially the problems previously mentioned can be tackled with inference rights. It is possible, e.g., to prevent facts that possibly do not contribute to the proof from generating inferences. This way, such inferences are delayed for a certain period of time and possibly unnecessary offspring cannot be generated. This shows that by means of inference rights some kind of "partial forgetting" can be realized. Facts that possibly do not contribute to a proof are not deleted and hence totally forgotten, but only certain inferences where such facts could be involved in are omitted.

In the sequel, we shall describe at first basics of generating theorem provers and especially the area of *condensed detachment* we chose to experiment in with inference rights. After that, we introduce in section 3 inference rights and give some remarks on the way in which they can be incorporated into already existing inference systems. Furthermore, we propose a method well-suited for controlling such modified inference systems. In section 4 we instantiate our abstract framework in the area of condensed detachment. We discuss especially conditions that are necessary to preserve the completeness of derivations even if inference rights are utilized. Experimental results obtained with these techniques are presented in this section, too. We conclude the report with a summary of our work and propose some possible future extensions.

2 Basics of Automated Theorem Provers

2.1 Fundamentals

The problem in automated theorem proving is given as follows: Given a set of facts Ax (axioms), is a further fact λ_G (goal) a logic consequence of the axioms? A fact may be a clause, equation, or a general first or higher-order formula. The definition of "logic consequence" depends heavily on the concrete problem one is interested in.

Commonly, automated theorem provers utilize certain *calculi* for accomplishing the task mentioned above. *Analytic calculi* attempt to recursively break down and transform a goal into sub-goals that can finally be proven immediately with the axioms. *Generating calculi* go the other way by continuously producing logic consequences from Ax until a fact covering the goal appears. We shall here concentrate on generating calculi.

Typically a generating calculus contains several inference rules which can be applied to a subset of the given facts (search state). Expansion inference rules are able to synthesize a new fact from known ones and add these facts to the current set. Contracting inference rules allow for the deletion of facts or replacing facts by other ones.

A common principle to solve proof problems algorithmically with a generating calculus is employed by most systems (algorithm `GTP`: generating theorem prover): Essentially, a theorem prover maintains a set F^P of so-called *potential* or *passive facts* from which it selects and removes one fact λ at a time. After the application of some contracting inference rules on λ, it is put into the set F^A of *activated facts*, or discarded if it was deleted by a contracting rule (*forward subsumtion*). Activated facts are, unlike potential facts, allowed to produce new facts via the application of expanding inference rules. The inferred new facts are put into F^P. We assume the expanding rules to be exhaustively applied on the elements of F^A. Initially, $F^A = \emptyset$ and $F^P = Ax$. The indeterministic selection or *activation step* is realized by heuristic means resulting in a search. To this end, a search-guiding heuristic $\mathcal{H}$ associates a natural number $\mathcal{H}(\lambda) \in I\!N$ with each $\lambda \in F^P$. Subsequently, that $\lambda \in F^P$ with the smallest weight $\mathcal{H}(\lambda)$ is selected. Ties are usually broken according to the FIFO-strategy ("*first in–first out*").

2.2 Condensed Detachment

A typical example for generating calculi is the inference system $\mathcal{CD}$ which contains the inference rule *condensed detachment* (`CondDet`) (see [14] and [7] for motivation and a theoretical background). Since $\mathcal{CD}$ contains only one expanding and one contracting inference rule it is very simple. But nevertheless resulting proof problems can be very challenging. Therefore, condensed detachment was chosen as a test domain by several researchers before ([10], [8], [11], [15]) and the choice of condensed detachment as our test domain surely is justified. The rules of the inference system $\mathcal{CD}$ manipulate first-order terms. These terms are defined as usual, involving a finite set $\mathcal{F}$ of function symbols and an enumerable set of variables $\mathcal{V}$. The inference system $\mathcal{CD}$ is defined as follows:

Definition 2.1 (The inference system $\mathcal{CD}$)
Let $f \in \mathcal{F}$ be a distinguished binary function symbol. Then $\mathcal{CD}$ contains the rules

$$\texttt{(Subsum)} \quad \Lambda \cup \{s,t\} \vdash \Lambda \cup \{s\}; \exists \text{ substitution } \sigma : \sigma(s) \equiv t$$

$$\texttt{(CondDet)} \quad \Lambda \cup \{s,t\} \vdash \Lambda \cup \{s,t,u\}; s \equiv f(s',u'), \sigma = mgu(s',t), u \equiv \sigma(u')$$

We denote $\Lambda \vdash \Lambda'$ if Λ' can be derived from Λ by the application of one inference rule. A sequence $(\Lambda_i)_{i\geq 0}$ with $\Lambda_j \vdash \Lambda_{j+1}$ $(j \geq 0)$ is called a $\mathcal{CD}$-derivation. A term t is $\mathcal{CD}$-derivable from Λ_0 if (and only if) a $\mathcal{CD}$-derivation $(\Lambda_i)_{i\geq 0}$ and an index j exist such that $t \in \Lambda_j$. t is called $\mathcal{CD}$-provable from Λ_0 iff a $\mathcal{CD}$-derivable term t' and a substitution σ exist such that $\sigma(t') \equiv t$. A proof problem $\wp = (Ax, t)$ is solvable iff t is $\mathcal{CD}$-provable from Ax. An important property of $\mathcal{CD}$-derivations is the *fairness*: we call a $\mathcal{CD}$-derivation $(\Lambda_i)_{i\geq 0}$ fair iff for each $\mathcal{CD}$-derivable term t a j, σ exists such that $t' \in \Lambda_j$ and $\sigma(t') \equiv t$. In the case that an algorithm only produces fair $\mathcal{CD}$-derivations each solvable proof problem can be solved with the help of the algorithm.
We have described before how to solve proof problems algorithmically. It is interesting in which way algorithms or heuristics have to be construed in order to produce only fair $\mathcal{CD}$-derivations. The following theorem formulates demands on a heuristic $\mathcal{H}$.

Theorem 2.1 (Fairness of a $\mathcal{CD}$-derivation)
The algorithm `GTP` *produces fair $\mathcal{CD}$-derivations if the associated heuristic guarantees that each fact being passive at one moment is activated or subsumed and discarded after a finite period of time.*
Proof: *We have to show that for each Λ_0 and for each term t which is $\mathcal{CD}$-derivable from Λ_0 holds: The algorithm computes a $\mathcal{CD}$-derivation $(\Lambda_i)_{i\in N}$ such that $t' \in \Lambda_j$ for a j and $\sigma(t') \equiv t$ for a substitution σ. Therefore, we show: t' is passive at a fixed moment. Let t be a term. Let $M_t = \{t_1, \ldots, t_n\}$ be a minimal set of terms—$\mathcal{CD}$-derivable from Λ_0—that is needed to derive t.*[1] *We apply induction over n:*
$n = 0$: Then $t \in \Lambda_0$, i.e. at the beginning passive.
$n > 0$: Since M_t is a (minimal) set needed to infer t, there are terms $t_j, t_k \in M_t$, such that t can be derived from t_j and t_k via `CondDet`*. Since t_j and t_k are $\mathcal{CD}$-derivable from Λ_0 and $|M_{t_j}|, |M_{t_k}| < n$ holds, there are two moments τ_j and τ_k such that t'_j and t'_k are passive. For these terms it holds that $\sigma_j(t'_j) \equiv t_j$ and $\sigma_k(t'_k) \equiv t_k$. Because of our precondition t'_j and t'_k do not remain infinitely long in F^P, i.e. there are moments $\tilde{\tau}_j$ and $\tilde{\tau}_k$ after that t'_j and t'_k are not passive any longer. Hence, at the moment* $\max(\{\tilde{\tau}_j, \tilde{\tau}_k\})$ *there exist two active facts $\tilde{t}_j$ and $\tilde{t}_k$ with $\tilde{\sigma}_j(\tilde{t}_j) \equiv t'_j$ and $\tilde{\sigma}_k(\tilde{t}_k) \equiv t'_k$. Because of the fact that a term t' with the required properties can be derived from $\tilde{t}_j$ and $\tilde{t}_k$ via* `CondDet` *and because of the fact that the rule* `CondDet` *is exhaustively applied on the active facts t' is passive at a moment.* □

Such heuristics can be construed quite easily, e.g. a heuristic $\mathcal{H}$ is fair if the set $M_z = \{\lambda \in Term(\mathcal{F}, \mathcal{V}) : H(\lambda) = z\}$ is finite for each natural number z. It is to be emphasized that the efficiency of the algorithm strongly depends on the heuristic and that the quality of heuristics depends on the given proof problem.

[1] Minimal means in this context that there is no set with fewer elements from which t can be derived.

3 Basics of Inference Rights

3.1 Discussion of the Algorithm GTP

If we take a closer look at GTP we can at first recognize that the algorithm allows for a clear and simple control of applications of the inference rules. Furthermore, GTP facilitates the control of rule applications because only one decision point exists—the choice of the next potential fact—which can easily be controlled by heuristic means. Thus, it is possible to develop very specialized heuristics (see, e.g. [2]) and therefore very specialized theorem provers. Moreover, GTP facilitates the development of distributed theorem provers. Constructing different cooperating heuristics, i.e. different cooperating incarnations of GTP, allows for a distributed theorem prover with a performance that is clearly superior to sequential implementations of GTP ([1]). Furthermore, it is possible to employ learned knowledge for controlling the indeterministic activation step which can improve the algorithm in many cases (see [6], [4]).

Nevertheless, the simple and inflexible scheme of GTP has some disadvantages. Because of the fact that the next fact to be activated is selected out of an ever growing set of facts and that only a few of these facts contribute to a proof, it is very probable that an unnecessary fact is selected and activated. Such a fact λ remains in the set F^A which often has serious consequences. On the one hand more unnecessary facts are generated because λ can be involved in a lot of applications of expanding inference rules in future. Thus, a lot of computation time is wasted. On the other hand, if such unnecessary descendants of λ are activated in future the number of facts that do not contribute to a proof can grow enormously. Since a large number of facts entails a high demand for memory and computation time it is possible that the proof is unnecessarily delayed or even not found.

Another main disadvantage of GTP is that no further investigation of activated facts—e.g. with another heuristic or with respect to new information—takes place. Thus, no a posteriori knowledge can be incorporated into the algorithm. The following example gives a rough overview of how such a posteriori knowledge could be utilized: Usually a lot of different proofs for a given proof problem exist. Thus, it is reasonable to search for short proof runs, i.e. proof runs where only few steps not contributing to the proof are performed. In the case that a fact λ has been activated which is involved in the application of many expanding inference steps it possibly does not contribute to short proof runs because a lot of possibly unnecessary "offspring" is generated. A modification of the search state which, e.g., forces λ to be involved only in contracting but not in expanding inferences could be the right way to cope with this problem. This is not an option in GTP, however, because *all* kinds of inferences are exhaustively applied to elements of F^A.

All in all it is sensible to preserve the main principles of GTP—the division of the inferred facts into F^A and F^P and the use of heuristics in order to activate facts—because of the advantages mentioned before. To deal with the disadvantages the two following aspects should be integrated into the algorithm: In order to integrate a posteriori knowledge the facts $\lambda \in F^A$ should be analyzed periodically. This way it should be possible to detect unnecessary facts, i.e. facts which do not contribute to any proof, or at least

facts that possibly do not contribute to short proof runs. The second step should be the "restructuring" of the current search state so as to avoid generating too many facts. Nevertheless, such a restructuring must neither destroy the simplification power of the system of activated facts nor the completeness. The latter means that finally all inferences that are, e.g., delayed for a certain period of time, will take place if they are needed to conclude the proof. Such a restructuring could be achieved by "partial forgetting" of unnecessary facts. This means that the information on the activation of unnecessary facts is not totally forgotten, i.e. possibly unnecessary facts are not deleted, but such facts are forgotten in such a manner that they cannot take part in certain inferences.

We will only sketch the first aspect because the detection of unnecessary facts depends heavily on the concrete calculus (see section 4) and so-called *referees* ([3]) are known to be fairly well-suited for judging facts. In the sequel, we shall hence concentrate on the second aspect, i.e. on the question how partial forgetting and hence a better control—especially of expanding inference rules—can be achieved. Our solution to this very issue are *inference rights* which allow for a finer grained control of inferences.

3.2 Inference Rights

The main idea of inference rights is to enrich the facts an inference system works on with rights to perform inferences. The intended use of these rights is as follows: If a fact is assumed to be possibly unnecessary or not contributing to short proof runs the generation power of this fact is restricted. This can be achieved by retracting the right to take part in expanding inferences. Additional conditions on the algorithm, however, are necessary to guarantee that finally all necessary inferences are performed. Thus, in the inference system it should be possible to recover rights to perform inferences.

Our method to model inference rights is to use annotations to a fact that determine the inferences in which a fact is allowed to be involved in:

Definition 3.1 (Inference Right, Fact with Inference Right)
Let I be an inference system, and $\mathcal{I}$ be the set of inference rules. Let λ be a fact. An inference right w.r.t. I is a set $C \subseteq \mathcal{I}$.
A fact with inference right is a pair (λ, C). In the sequel, we write $\lambda|C$ instead of (λ, C).

We give some remarks on the way inference rights can be incorporated into already existing inference systems. At first, inference rules do not work on sets of facts Λ any longer but on facts with rights $\Lambda^{\mathcal{R}}$. The rights stem from the original inference system and are subsets of the original inference rules. They restrict the applicability of inference rules. The expanding inference rule

$$(\texttt{Exp}) \qquad \Lambda \cup \{\lambda_1, \ldots, \lambda_n\} \vdash \Lambda \cup \{\lambda_1, \ldots, \lambda_n, \lambda\}; Cond(\lambda_1, \ldots, \lambda_n)$$

could be, e.g., modified in the following manner:

$$(\texttt{Exp}^{\mathcal{R}}) \quad \Lambda^{\mathcal{R}} \cup \{\lambda_1|C_1,\ldots,\lambda_n|C_n\} \vdash \Lambda^{\mathcal{R}} \cup \{\lambda_1|C_1,\ldots,\lambda_n|C_n,\lambda|\mathcal{I}\}; Cond(\lambda_1,\ldots,\lambda_n) \wedge \forall i, 1 \leq i \leq n : \texttt{Exp} \in C_i$$

So, facts can only be involved in an inference if they have the right to perform it. Moreover, further rules are necessary which are only needed to handle rights. Such rules could be for instance:

$$(\texttt{Retract}^{\mathcal{R}}) \quad \Lambda^{\mathcal{R}} \cup \{\lambda|D\} \vdash \Lambda^{\mathcal{R}} \cup \{\lambda|C\}; C \subset D$$

$$(\texttt{Recover}^{\mathcal{R}}) \quad \Lambda^{\mathcal{R}} \cup \{\lambda|C\} \vdash \Lambda^{\mathcal{R}} \cup \{\lambda|D\}; C \neq \mathcal{I} \wedge C \subset D$$

The rule $\texttt{Retract}^{\mathcal{R}}$ is needed, e.g., to forbid facts to perform the generation of facts. The rule $\texttt{Recover}^{\mathcal{R}}$ is needed to add rights. In the area of condensed detachment this is necessary to allow fair derivations (cf. section 4).

Note that inference rights offer only an abstract framework to enrich facts with further information. An important question is now in which way such an extended inference system should be controlled. Such a control should allow for a gain of efficiency in comparison with the original inference mechanism. Moreover, completeness should be guaranteed. Since these aspects depend mainly on the concrete inference system one is interested in, we will discuss them in the following section in more detail.

In the sequel, we will nevertheless give a rough idea of how inference rights could be used. Algorithm $\texttt{GTP}^{\mathcal{R}}$—utilizing inference rights—is an extension of GTP, i.e. it divides the facts into the sets F^A and F^P and performs inferences as described before. The main difference is that active facts are periodically judged and a certain number of bad facts is determined. Then, inference rule $\texttt{Retract}^{\mathcal{R}}$ is applied to these facts in order to forbid them to produce new ones via expanding inference rules ("deactivation"). This way, certain facts are partially forgotten and not allowed to generate new facts. If we deactivate facts that really do not contribute to any proof unnecessary offspring of these facts is avoided. If we deactivate facts that are able to generate a lot of facts it is possible that shorter proof runs occur. In order to preserve completeness it is necessary that all inferences (needed for completeness) which are delayed for a certain period of time finally take place. To this end, it is convenient to note which facts were involved in an application of the rule $\texttt{Retract}^{\mathcal{R}}$. Thus, it is possible to apply $\texttt{Recover}^{\mathcal{R}}$ on such facts after a certain period of time and to perform the inferences that were delayed before. The technical realization could be as follows: We introduce a *recover set* F^R and move, after the application of $\texttt{Retract}^{\mathcal{R}}$ on an active fact λ, this fact from F^A to F^R. The facts from F^R can be utilized for contracting but not for expanding inferences. Furthermore, the facts from F^R are possible candidates for activation steps, i.e. facts are not only selected from F^P but also periodically from F^R. If a fact $\lambda \in F^R$ is selected its inference rights is set to $\mathcal{I}$ via $\texttt{Recover}^{\mathcal{R}}$. After that it is processed analogously to a selected potential fact, i.e. inferences delayed previously can be performed.

A crucial step with regard to performance is the selection of facts from F^R. If we select facts too frequently, i.e. they are not forgotten for a long time, expanding inferences

where bad facts are involved in are delayed for a short time only, and we hence cannot expect to gain much efficiency. But if we select facts from F^R only very seldom and deactivate a fact that really contributes to a proof it is possible that important inferences are delayed for a long time and hence the runtime increases. Thus, a good compromise between these extremes has to be achieved. At least it is necessary that all facts from F^R are finally selected in order to preserve completeness (cf. section 4).

4 Inference Rights and Condensed Detachment

So far we have introduced inference rights as a general framework to modify inference systems. This modification is necessary in order to achieve a better control especially of the expanding inference rules. Since the concrete realization depends on the calculus one is interested in we have only given a few remarks on the way how to control such an inference system. Therefore, this section describes more precisely in which way inference rights can be utilized in the area of condensed detachment. We chose condensed detachment as a first test domain because the inference system $\mathcal{CD}$ used in this area is quite typical for generating provers.

At first it is necessary to integrate inference rights into the inference system $\mathcal{CD}$ resulting in a new inference system $\mathcal{CD}^{\mathcal{R}}$. The next step is to present a concrete algorithm for controlling $\mathcal{CD}^{\mathcal{R}}$. Note that we are still interested in solving $\mathcal{CD}$ proof problems and need an algorithm that is able to solve them. We describe a basic algorithm—using an abstract function β in order to detect unnecessary facts—and give sufficient conditions on the algorithm to guarantee that it only produces fair derivations. Finally, we introduce a possible realization of function β for judging facts and use this function in some experiments.

4.1 The Inference System $\mathcal{CD}^{\mathcal{R}}$

The inference system $\mathcal{CD}^{\mathcal{R}}$ is an extension of $\mathcal{CD}$. Therefore, the inference rules `CondDet` and `Subsum` must be adapted in order to work with facts with inference rights. Furthermore, the rules $\texttt{Retract}^{\mathcal{R}}$ and $\texttt{Recover}^{\mathcal{R}}$ are necessary to deal with rights (cf. section 3). In this context the set of inference rights is given as $\mathcal{R} = \{M : M \subseteq \mathcal{I} = \{\texttt{CondDet}, \texttt{Subsum}\}\}$. The following definition introduces the inference system $\mathcal{CD}^{\mathcal{R}}$.

Definition 4.1 (The inference system $\mathcal{CD}^{\mathcal{R}}$)
Let $f \in \mathcal{F}$ be a distinguished binary function symbol. Then $\mathcal{CD}^{\mathcal{R}}$ contains the rules

$(\texttt{Subsum}^{\mathcal{R}})$ $\quad \Lambda^{\mathcal{R}} \cup \{s|C_1, t|C_2\} \vdash \Lambda^{\mathcal{R}} \cup \{s|C_1\}; \exists\sigma : \sigma(s) \equiv t \wedge \texttt{Subsum} \in C_1$

$(\texttt{CondDet}^{\mathcal{R}})$ $\quad \Lambda^{\mathcal{R}} \cup \{s|C_1, t|C_2\} \vdash \Lambda^{\mathcal{R}} \cup \{s|C_1, t|C_2, u|\mathcal{I}\}; (s \equiv f(s', u'), \sigma = mgu(s', t),$
$u \equiv \sigma(u')) \wedge \texttt{Conddet} \in C_1, C_2$

$(\texttt{Recover}^{\mathcal{R}})$ $\quad \Lambda^{\mathcal{R}} \cup \{s|C_1\} \vdash \Lambda^{\mathcal{R}} \cup \{s|C_2\}; C_1 \neq \mathcal{I} \wedge C_1 \subset C_2$

$(\texttt{Retract}^{\mathcal{R}})$ $\quad \Lambda^{\mathcal{R}} \cup \{s|C_1\} \vdash \Lambda^{\mathcal{R}} \cup \{s|C_2\}; C_2 \subset C_1$

Inference system $\mathcal{CD}^{\mathcal{R}}$ follows exactly the principles pointed out in section 3. The expanding inference rule $\texttt{CondDet}^{\mathcal{R}}$ can only be applied if both facts that take part in it have the right to perform it. In order to subsume a fact it is only necessary to have the subsumtion right. Note that it is not required that the fact to be subsumed has this inference right. Inference rules $\texttt{Retract}^{\mathcal{R}}$ and $\texttt{Recover}^{\mathcal{R}}$ are exactly the same as described before.

In analogy to section 2 we write $\Lambda_i^{\mathcal{R}} \vdash \Lambda_{i+1}^{\mathcal{R}}$ if $\Lambda_{i+1}^{\mathcal{R}}$ can be derived from $\Lambda_i^{\mathcal{R}}$ by the application of one inference rule. $(\Lambda_i^{\mathcal{R}})_{i\geq 0}$ is called a $\mathcal{CD}^{\mathcal{R}}$-derivation iff $\Lambda_j^{\mathcal{R}} \vdash \Lambda_{j+1}^{\mathcal{R}} \forall j \geq 0$. A fact with inference right $t|C$ is $\mathcal{CD}^{\mathcal{R}}$-derivable from $\Lambda_0^{\mathcal{R}}$ iff a $\mathcal{CD}^{\mathcal{R}}$-derivation $(\Lambda_i^{\mathcal{R}})_{i\geq 0}$ and an index j exist such that $t|C \in \Lambda_j^{\mathcal{R}}$. We call a $\mathcal{CD}^{\mathcal{R}}$-derivation $(\Lambda_i^{\mathcal{R}})_{i\geq 0}$ fair iff for each $\mathcal{CD}$-derivable fact t there exists a j, σ such that $t'|C \in \Lambda_j^{\mathcal{R}}$ and $\sigma(t') \equiv t$. Note that by means of fair $\mathcal{CD}^{\mathcal{R}}$-derivations it is possible to solve $\mathcal{CD}$ proof problems. It is only necessary to perform $\mathcal{CD}^{\mathcal{R}}$-derivations until a fact appears that subsumes the goal. The right is ignored in that case since it is only needed to circumvent certain inferences during the inference process. It would be unwise to consider the right if a fact subsuming the goal is found because in such a case the proof might be unnecessarily delayed.

4.2 An Algorithm for Controlling $\mathcal{CD}^{\mathcal{R}}$

In the sequel, we introduce an algorithm that allows to control $\mathcal{CD}^{\mathcal{R}}$ easily and to solve $\mathcal{CD}$ proof problems very efficiently. It is to be emphasized that our algorithm is only one of a lot of different possible ones. In general, there might be many different ways in which way the new rules provided by the inference system can be utilized.

Basic Algorithm: In order to construe an algorithm for controlling $\mathcal{CD}^{\mathcal{R}}$ it is sensible to employ the algorithm $\texttt{GTP}^{\mathcal{R}}$ as described in section 3. Thus, it is still possible to perform the activation steps heuristically. Therefore, techniques like learning or goal orientation ([2]) can still be utilized. As described in section 3, $\texttt{GTP}^{\mathcal{R}}$ employs the inference rights to restrict the applicability of inference rules. This restriction takes place in such a manner that activated facts which do not appear to be contributing to a proof or only contribute to long proof runs are "deactivated" periodically. Thus, they are not allowed to be involved in expanding, but only in contracting inferences. To this end, a function β is needed that determines the active facts that behave "badly" w.r.t. a certain criterion. More exactly, $\beta(F^A) = \{\lambda_1|\mathcal{I}, \ldots, \lambda_m|\mathcal{I}\} \subseteq F^A$, i.e. β selects a fixed number of active facts. The realization of algorithm $\texttt{GTP}^{\mathcal{R}}$—well-suited for proving $\mathcal{CD}$ proof problems—uses β to determine the facts to be deactivated (see figure 1).

As one can see, algorithm $\texttt{GTP}^{\mathcal{R}}$ is instantiated in the following aspects: The periodical deactivation of facts is realized in such a manner that the function β determines—after a fixed number n of activation steps—a fixed number m of active facts that should not take part in expanding inferences any longer. In our experiments we employed a number m that is a certain percentage d of the number of activation steps n. The deactivation is achieved via the inference rule $\texttt{Retract}^{\mathcal{R}}$. The fact $\lambda|C$ originating from the application of $\texttt{Retract}^{\mathcal{R}}$ is moved from the set F^A to the recover set F^R. After that, it can only take part in $\texttt{Subsum}^{\mathcal{R}}$ but not in $\texttt{CondDet}^{\mathcal{R}}$. In order to achieve

```
begin
  F^A := ∅, F^R := ∅, F^P := {ax|I : ax ∈ Ax}, cnt := 0
  while F^P ∪ F^R ≠ ∅
    if cnt = n
      S := β(F^A)
      forall λ|I ∈ S
        F^A := F^A \ {λ|I}, F^R := F^R ∪ {λ|I \ {CondDet}}
      cnt := 0
    select λ|C ∈ F^P ∪ F^R with minimal H(λ)
    if λ|C ∈ F^P : F^P := F^P \ {λ|C}
    else F^R := F^R \ {λ|C}, C := I
    if λ|C was selected from F^P ∧ ∃λ'|D ∈ F^A ∪ F^R, σ: σ(λ') ≡ λ:
      delete λ|C
    else
      if μ(λ) ≡ λ_G for a μ: "proof found"
      M := {λ' : λ' CondDet descendant of λ, λ̃; λ̃|I ∈ F^A}
      F^P := F^P ∪ {λ'|I : λ' ∈ M}
      F^A := F^A ∪ {λ|C}, cnt := cnt + 1
  "proof failed"
end
```

Figure 1: Algorithm $\mathtt{GTP}^{\mathcal{R}}$

that expanding inferences were λ is involved in are only delayed for a certain period of time but not strictly forbidden, inference rule $\mathtt{Recover}^{\mathcal{R}}$ must be applied later, i.e. $\lambda|C$ has to be activated again. To this end it is sensible to select the facts that should be activated from $F^P \cup F^R$. As we have mentioned previously the duration a fact $\lambda|C$ remains in F^R influences heavily the performance of a prover. We chose the search-guiding heuristic $\mathcal{H}$ for accomplishing the task of selecting facts from F^R. Thus, we prefer small facts w.r.t. $\mathcal{H}$ that are possibly more important for the proof. Moreover, utilizing $\mathcal{H}$ for selection of facts allows to preserve completeness (see below). It is reasonable, however, to change the heuristic weight $\mathcal{H}(\lambda)$ of a fact $\lambda \in F^R$ to $\gamma \cdot \mathcal{H}(\lambda)$, $\gamma \in I\!N$, $\gamma > 1$. Otherwise it might be often the case that a fact that was deactivated is activated immediately.

Fairness of $\mathcal{CD}^{\mathcal{R}}$-derivations: In order to guarantee fairness of $\mathcal{CD}$-derivations—performed by algorithm GTP—we had to cope with the following two aspects. At first we formalized theoretical demands on algorithm GTP which were sufficient to achieve fairness, namely that potential facts must not remain passive infinitely long. Furthermore, we gave remarks on the way how these demands could be realized. Thus, we formalized some conditions on the heuristic responsible for the activation of facts. In the sequel, we will hence deal with the same aspects w.r.t. $\mathcal{CD}^{\mathcal{R}}$ and $\mathtt{GTP}^{\mathcal{R}}$.

Intuitively, the following precondition should be sufficient to preserve fairness of derivations: In order to guarantee that all inferences are finally performed potential facts must not stay in F^P infinitely long. But since $\mathtt{GTP}^{\mathcal{R}}$ allows for the deactivation of active facts, i.e. to insert them into the recover set F^R, we must also guarantee that

these facts do not remain in F^R infinitely long. Furthermore, we have to consider the following problem: If an infinite cycle of activation and deactivation of two facts occur, i.e. each time one fact is deactivated before the other fact is activated, possibly necessary descendants of these facts will never be generated. In order to circumvent such infinite cycles it has to be forbidden that a fact is deactivated infinitely often. We formalize this in the following definition.

Definition 4.2 (The Precondition $\mathcal{P}$)
The precondition $\mathcal{P}$ on algorithm $\mathtt{GTP}^{\mathcal{R}}$ is defined as follows: $\mathcal{P}$ holds true iff the algorithm is constructed in such a way (the heuristic and the deactivation function β are realized in such a manner) that no fact—being element of F^P or F^R at a certain point in time—stays infinitely long in F^P or F^R, respectively, and *that no fact $\lambda|\mathcal{I}$ is deactivated infinitely often.*

Precondition $\mathcal{P}$ is indeed sufficient to entail fair derivations. However, the proof of this property is more complex than the proof in section 2. We need one lemma that formalizes certain properties of algorithm $\mathtt{GTP}^{\mathcal{R}}$. In order to describe such properties we use numbers of activations to refer to certain periods of time, i.e. a moment τ corresponds to a number of activations n_τ. This is possible because the search state remains unchanged between two activation steps s and $s+1$.
The lemma shows that for a fact $\lambda|C$, $\lambda|C \in F^A \cup F^R$ at a moment τ, there is a moment $\tilde{\tau} \geq \tau$ such that $\lambda|\mathcal{I}$ remains in F^A for all moments following $\tilde{\tau}$.

Lemma 4.1
Let $\mathcal{P}$ be fulfilled. Let $(\Lambda_i^{\mathcal{R}})_{i\geq 0}$ be the $\mathcal{CD}^{\mathcal{R}}$-derivation produced by algorithm $\mathtt{GTP}^{\mathcal{R}}$. Let $t|C$ be a fact element of $F^A \cup F^R$ at moment τ. Then exists a moment $\tau' \geq \tau$ such that for all moments $\tilde{\tau} \geq \tau'$ holds: $t|\mathcal{I}$ is active.
Proof: *Since no fact being active or element of the recover set can be deleted, at each moment $\tilde{\tau} \geq \tau$ we have: $t|\mathcal{I} \in F^A$ or $t|C \in F^R$. Moreover, if at the moment $\tilde{\tau} \geq \tau$ the fact $t|\tilde{C}$ with such a property exists, there is at each moment $\hat{\tau} \geq \tilde{\tau}$ the fact $t|\hat{C}$ such that it holds: $t|\tilde{C} \succeq t|\hat{C}$ where*

$$t|\tilde{C} \succ t|\hat{C} \text{ iff } t|\hat{C} \in F^A \text{ and } t|\tilde{C} \in F^R \text{ or the number of deactivations of } t|\hat{C} \text{ is higher as the respective number of } t|\tilde{C}$$

Because of the fact that $\succ$ is Noetherian a moment $\tilde{\tau} \geq \tau$ and a fact with right $t|\tilde{C} \in F^A \cup F^R$ exists that is $\succ$-minimal. More exactly, $\mathcal{P}$ guarantees that $\tilde{C} = \mathcal{I}$ and $t|\mathcal{I} \in F^A$ (no fact remains in F^R infinitely long). Moreover, this fact will never be deactivated because it is $\succ$-minimal. □

Utilizing this lemma we can show—in analogy to section 2—that algorithm $\mathtt{GTP}^{\mathcal{R}}$ produces only fair derivations if precondition $\mathcal{P}$ is fulfilled.

Theorem 4.1 (Fairness of a $\mathcal{CD}^{\mathcal{R}}$-derivation)
Let $\mathcal{P}$ be fulfilled. Then algorithm $\mathtt{GTP}^{\mathcal{R}}$ produces only fair $\mathcal{CD}^{\mathcal{R}}$-derivations.
Proof: *We have to show that for each Λ_0 and for each term t which is $\mathcal{CD}$-derivable*

from Λ_0 it is true that the algorithm computes a $\mathcal{CD}^{\mathcal{R}}$-derivation $(\Lambda_i^{\mathcal{R}})_{i\geq 0}$ such that $t'|C \in \Lambda_j^{\mathcal{R}}$ for a j and $\sigma(t') \equiv t$ for a substitution σ. Therefore, we show: $t'|\mathcal{I}$ with $\sigma(t') \equiv t$ is passive at a certain moment. Let t be a term. Let $M_t = \{t_1, \ldots, t_n\}$ be a minimal set of terms—$\mathcal{CD}$-derivable from Λ_0—that is needed to $\mathcal{CD}$-derive t. We apply induction over n:
$n = 0$: Then $t|\mathcal{I} \in \Lambda_0^{\mathcal{R}}$, i.e. at the beginning passive.
$n > 0$: Since M_t is a (minimal) set needed to infer t, terms $t_j, t_k \in M_t$ exist, such that t can be derived from t_j and t_k via `CondDet`*. Since t_j and t_k are $\mathcal{CD}$-derivable from Λ_0 and $|M_{t_j}|, |M_{t_k}| < n$ holds, there are two moments τ_j and τ_k such that $t'_j|\mathcal{I}$ and $t'_k|\mathcal{I}$ are passive. For these terms it holds that $\sigma_j(t'_j) \equiv t_j$ and $\sigma_k(t'_k) \equiv t_k$. Because of our precondition $t'_j|\mathcal{I}$ and $t'_k|\mathcal{I}$ do not remain infinitely long in $\mathcal{F}^{\mathcal{P}}$, i.e. there are moments $\tilde{\tau}_j$ and $\tilde{\tau}_k$ after that $t'_j|\mathcal{I}$ and $t'_k|\mathcal{I}$ are not passive any longer. Hence, at the moment $\max(\{\tilde{\tau}_j, \tilde{\tau}_k\})$ two facts $\tilde{t}_j|C_j$ and $\tilde{t}_k|C_k$ exist, both are elements of $F^A \cup F^R$, with $\tilde{\sigma}_j(\tilde{t}_j) \equiv t'_j$ and $\tilde{\sigma}_k(\tilde{t}_k) \equiv t'_k$, $C_j, C_k \subseteq \mathcal{I}$. Lemma 4.1 guarantees that there is a moment $\hat{\tau}$ such that the facts $\tilde{t}_j|\mathcal{I}$ and $\tilde{t}_k|\mathcal{I}$ stay active. With the help of these two facts it is possible to derive via* `CondDet` *a fact $t'|\mathcal{I}$ that subsumes t. Because of the fact that $\tilde{t}_j$ and $\tilde{t}_k$ have rights to perform all inferences and all* `CondDet` *descendants are hence generated, $t'|\mathcal{I}$ is passive at a certain moment.* □

In order to fulfill precondition $\mathcal{P}$ it is at first necessary to employ a heuristic $\mathcal{H}$ which finally selects all of the potential facts and the facts from the recover set. In section 2 we showed in which way we can construct such heuristics. Moreover, we must avoid that an active fact $\lambda|\mathcal{I}$ is deactivated infinitely often. To this end, it is sensible to enrich each fact with a natural number that counts how often the respective fact has been deactivated. If a threshold of c_{max} is exceeded β is not allowed to select λ. Hence, λ cannot be deactivated in future.

Determination of bad facts: As we have described before β selects—after a fixed number of n activations—a set of facts $\{\lambda_1|\mathcal{I}, \ldots, \lambda_m|\mathcal{I}\}$ from the current set of active facts. It is reasonable to forbid β to select the following kinds of facts: At first facts whose number of deactivations has exceeded a certain threshold must not be selected for fairness reasons (see before). Furthermore, our experiments have shown that it is wise to forbid the deactivation of axioms. In the sequel, we will describe in which way β determines bad facts among the remaining ones. Remember, that we denote facts as bad if they contribute to no or only long proof runs.

Facts being involved in a lot of expanding inferences but only in a few contracting inferences contribute with a high probability to long proof runs. This is mainly because of the fact that they generate a lot of offspring. If this offspring is not needed for the proof the prover is forced to waste a lot of computation time to handle such facts. Hence, facts that possibly contribute to long proof runs can be detected by counting the inferences they were involved in.

The detection of facts that are not needed in any proof is much more difficult. It is to be emphasized that it is in general undecidable to predict if a fact contributes to a proof of a goal. Despite of the undecidability the probability is rather high to detect facts that are unnecessary for proving the goal. This is mainly due to the fact that in successful proof runs usually only a few activated facts ($\leq 5\%$) are needed. Moreover,

experiments have shown that facts needed for a proof are often quite general, i.e. they subsume a lot of deduced facts, and often have a small weight according to the heuristic of the prover ([3]). If we choose facts that have not subsumed many other facts or have a high weight according to the heuristic it is very probable that these facts are not needed for a proof. If we assume that we want to select facts periodically after n activations, i.e. at the moments $\tau_{0,n}, \tau_{1,n}, \ldots$, we can define the function $\beta = \beta_{\tau_{i,n}}$[2] for the selection of bad facts at moment $\tau_{i,n}$ as follows. Let $\mathcal{H}$ be the used heuristic, let $|gen_{\Delta\tau_{i,n}}(\lambda)|$ and $|del_{\Delta\tau_{i,n}}(\lambda)|$ be the number of facts that were generated and discarded, respectively, with the help of λ in the period between $\tau_{i-1,n}$ and $\tau_{i,n}$. Then set

$$\beta_{\tau_{i,n}}(\lambda) := |del_{\Delta\tau_{i,n}}(\lambda)| - |gen_{\Delta\tau_{i,n}}(\lambda)| - \mathcal{H}(\lambda)$$

A small value of $\beta_{\tau_{i,n}}$ is a sign that a fact λ behaves badly because a lot of facts have been generated but only a few facts have been subsumed.

This technique can be refined by considering facts λ as negative which contain terms $f(x,t')$ as subterms. In particular we must take the position such subterms occur at into consideration. If $\lambda \equiv f(x,t')$ we can derive a lot of terms with λ immediately, but if $\lambda \equiv f(s,f(x,t'))$, e.g., i.e. this special subterm occurs in a deeper position, $f(x,t')$ might be derived via `CondDet` and is only able to generate a lot of terms if it is activated later. Thereby the occurrence of $f(x,t')$ is less negative as before. The concrete technical realization is as follows: Let $O(\lambda)$ be the set of positions in λ, i.e. $\epsilon \in O(\lambda)$, if $\lambda \equiv f(t_1,\ldots,t_n)$ and $p \in O(t_i)$ $(1 \leq i \leq n)$ then $ip \in O(\lambda)$. Furthermore, $\lambda|p$ denotes the subterm of λ at position p, i.e. $\lambda|\epsilon \equiv \lambda$, if $\lambda \equiv f(t_1,\ldots,t_n)$ and $p \in O(t_i)$ $(1 \leq i \leq n)$ then $\lambda|ip \equiv t_i|p$. Furthermore, let $M \in I\!N$, $M > 1$. Then we can measure the depth of an occurrence of the subterm $f(x,t')$ in λ with the following function:

$$\theta(\lambda) = \begin{cases} \min(\{|p| : \lambda|p \equiv f(x,t')\}) & \text{, if } \lambda|p \equiv f(x,t') \text{ for a term } t' \text{ and } p \in O(\lambda) \\ M & \text{, otherwise} \end{cases}$$

If $\dot{-}$ denotes the non-negative difference, i.e. $a \dot{-} b = 0$ if $a < b$, we can refine the definition of $\beta_{\tau_{i,n}}$ to

$$\widetilde{\beta_{\tau_{i,n}}} = \beta_{\tau_{i,n}} - (M \dot{-} \theta(\lambda))$$

With the help of the parameter M we can decide until which depth subterms $f(x,t')$ are considered. Note that θ is usually only needed to break ties between facts that have the same weight according to $\beta_{\tau_{i,n}}$. Therefore facts are assessed mainly by means of statistical and not heuristical criteria.

Although this definition of function β is rather simple our results (cf. section 4.3) were fairly satisfactory. Nevertheless, it would be interesting to examine whether a more complex definition of β entails better results.

[2] Note that the function β depends strongly on the moment it is applied on the set of active facts. However, we make this dependency explicit only if it is necessary to avoid confusion.

Example	GTP(ϖ)		GTP$^{\mathcal{R}}(d = 10\%)$		GTP$^{\mathcal{R}}(d = 20\%)$		GTP$^{\mathcal{R}}(d = 30\%)$		OTTER
	time	activ.	time	activ.	time	activ.	time	activ.	
LCL006-1	43	544	36	534	32	532	27	528	244
LCL016-1	47	554	41	557	34	541	30	466	151
LCL045-1	60	670	53	665	43	644	32	615	1467
LCL058-1	59	710	54	710	46	708	37	703	423
LCL060-1	63	733	59	733	49	733	40	726	509
LCL068-1	121	982	109	983	89	977	148	1217	257
LCL069-1	80	773	73	774	61	774	25	501	5
LCL070-1	55	715	47	715	37	693	21	568	480
LCL071-1	55	687	51	687	36	665	33	656	511
LCL072-1	95	826	84	826	72	825	39	613	224
LCL111-1	79	852	68	852	56	835	45	832	5
LCL113-1	64	737	55	737	46	731	37	725	1468
LCL115-1	48	616	45	622	38	617	29	592	7

Table 1: GTP vs. GTP$^{\mathcal{R}}$

4.3 Experimental Results

In order to perform an experimental evaluation we used the program CODE ([5]) as our basic prover. We integrated into this program that originally used algorithm GTP inference rights employing the techniques presented in section 4. We applied CODE to proof problems stemming from the area of logic calculi which McCune and Wos tackled in [8] with their renowned prover OTTER. These problems can also be found in the TPTP library ([13]) version 1.2.1, namely in the LCL domain. In order to show that the speed-ups we achieved with our techniques are not due to the weakness of our basic prover we compare the results obtained with CODE with those of OTTER. The results with CODE were obtained on a SPARCstation ELC, the results with OTTER on a SPARCstation 1+ which is a comparable machine. The results of OTTER depicted in the following table 1 are those presented in [8], and were achieved by the best of up to six different heuristics. The results of CODE were obtained with heuristic ϖ explained in [5].

Results: In the sequel, we shall investigate whether speed-ups can be achieved if we use algorithm GTP$^{\mathcal{R}}$ instead of algorithm GTP. Deactivation of facts was performed after $n = 50$ activation steps. At each of these moments m facts were deactivated, with $m = d\%$ of n (d can be found in table 1). The parameter M that denotes the maximal penalty that can be given due to syntactical properties of a term was set to $M = 5$, the parameter γ to the value 2.

The table 1 compares the results obtained with CODE and OTTER. Columns 2 and 3 show the run time (in seconds) and the number of activations needed by heuristic ϖ, the next six columns show the corresponding values if we deactivate a certain percentage d of the activated facts. Column 10 gives the run time needed when using OTTER. Table 1 shows that algorithm GTP$^{\mathcal{R}}$ which utilizes inference rights outperforms GTP. If we deactivate only a small percentage of the activated facts the speed-ups are rather low, but if we increase the number of deactivated facts we can achieve higher speed-ups.

In table 1 we can find one problem where deactivation of facts causes longer run-times because important facts were deactivated. But such a situation is quite improbable and the proof could still be found because we do not loose completeness by deactivation.

Evaluation: If we take a closer look at the problems where we achieved speed-ups we can recognize that in nearly all cases the "real" proof of the goal did not change. As the real proof we denote the sequence of activated facts, ordered w.r.t. the moment of their activation, that were really necessary for proving the goal. This way it is obvious that our deactivation function is indeed well-suited in judging facts. Although the real proof did not change we could achieve speed-ups. These speed-ups stem from the following two aspects: On the one hand table 1 shows that when using $\texttt{GTP}^{\mathcal{R}}$ instead of GTP often less activation steps had to be performed. This is mainly due to the fact that after the deactivation of unnecessary facts their unnecessary offspring was not generated and hence not activated. On the other hand we find some examples where we achieved speed-ups although the number of activation steps did not decrease. This is possible because if we use $\texttt{GTP}^{\mathcal{R}}$ instead of GTP the period of time needed for one activation step is shorter: The periodical deactivation of facts entails that the number of facts that take part in time consuming expanding inferences increases only slightly in comparison to conventional approaches.

5 Conclusion and Future Work

Automated deduction systems have reached a considerable level of performance. Nevertheless, the use of conventional approaches to control deduction systems—heuristic control of the search and exhaustive application of inference rules (saturation strategy)—is sometimes problematic. Because of the difficulty to discover an appropriate heuristic for a lot of different examples it is very probable to employ a heuristic which activates many facts not contributing to a proof. This entails much overhead due to the common saturation strategy.

Our approach of controlling the search by means of inference rights can help to deal better with these problems. Integration of inference rights into already existing inference systems makes it possible to achieve a more flexible control of the search and hence to reduce the amount of time for processing unnecessary facts. Inference rights allow to introduce an efficient mechanism for forgetting of facts what usually is not an option in generating theorem provers. Despite of the fact that they cause a small overhead our experimental results in the area of condensed detachment were fairly satisfactory. Substantial speed-ups in comparison to standard methods were achieved.

In order to substantiate our work with further results future work should deal with experiments in different calculi. In particular we have to consider the fact that most calculi contain more inference rules as only one rule for generating and one rule for deleting of facts. Hence, further research is necessary to find out whether the restriction of the applicability of such rules via inference rights is sensible.

References

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

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

[3] **Denzinger, J.; Fuchs, D.:** *Referees for Teamwork*, Proc. FLAIRS '96, Key West, FL, USA, 1996, pp. 454–458.

[4] **Denzinger, J.; Schulz, S.:** *Learning Domain Knowledge to Improve Theorem Proving*, Proc. CADE-13, New Brunswick, NJ, USA, LNAI 1104, 1996, pp. 62–76.

[5] **Fuchs, M:** *Powerful Search Heuristics Based on Weighted Symbols, Level and Features*, Proc. FLAIRS '96, Key West, FL, USA, 1996, pp. 449–453.

[6] **Fuchs, M.:** *Experiments in the Heuristic Use of Past Proof Experience*, Proc. CADE-13, New Brunswick, NJ, USA, LNAI 1104, 1996, pp. 523–537.

[7] **Łukasiewicz, J.:** *Selected Works*, L. Borkowski (ed.), North-Holland, 1970.

[8] **McCune, W.; Wos, L.:** *Experiments in Automated Deduction with Condensed Detachment*, Proc. CADE-11, Saratoga Springs, NY, USA, 1992, LNAI 607, pp. 209–223.

[9] **McCune, W.:** *OTTER 3.0 Reference Manual and Guide*, Techn. Report ANL-94/6, Argonne Natl. Laboratory, 1994.

[10] **Peterson, G.J.:** *An automatic theorem prover for substitution and detachment systems*, Notre Dame Journal of Formal Logic, Vol. 19, Number 1, January 1976, pp. 119–122.

[11] **Slaney, J.:** *SCOTT: A Model-Guided Theorem Prover*, Proc. IJCAI '93, Chambery, FRA, 1993, pp. 109–114.

[12] **Sutcliffe, G.:** *A Heterogeneous Parallel Deduction System*, Technical Report ICOT TM-1184, Proceedings of FGCS'92 Workshop W3, 1992.

[13] **Sutcliffe, G.; Suttner, C.; Yemenis, T.:** *The TPTP Problem Library*, Proc. CADE-12, Nancy, FRA, 1994, LNAI 814, pp. 252–266.

[14] **Tarski, A.:** *Logic, Semantics, Metamathematics*, Oxford University Press, 1956.

[15] **Wos, L.:** *Searching for Circles of Pure Proofs*, JAR 15, 1995, pp. 279–315.