Goal oriented equational theorem proving using team work*

Jörg Denzinger, Matthias Fuchs
Department of Computer Science
University of Kaiserslautern
Postfach 3049
67653 Kaiserslautern
Email: {denzinge, fuchs}@informatik.uni-kl.de

Abstract

The team work method is a concept for distributing automated theorem provers and so to activate several experts to work on a given problem. We have implemented this for pure equational logic using the unfailing Knuth-Bendix completion procedure as basic prover. In this paper we present three classes of experts working in a goal oriented fashion. In general, goal oriented experts perform their job "unfair" and so are often unable to solve a given problem alone. However, as a team member in the team work method they perform highly efficient, even in comparison with such respected provers as Otter 3.0 or REVEAL, as we demonstrate by examples, some of which can only be proved using team work.

The reason for these achievements results from the fact that the team work method forces the experts to compete for a while and then to cooperate by exchanging their best results. This allows one to collect "good" intermediate results and to forget "useless" ones. Completion based proof methods are frequently regarded to have the disadvantage of being not goal oriented. We believe that our approach overcomes this disadvantage to a large extend.

Keywords :
Automated theorem proving, distributed deduction, goal oriented completion.

*This work was supported by the "Forschungsschwerpunkt Deduktion" of the DFG

1 Introduction

The Knuth-Bendix completion procedure ([KB70]) is known to be highly efficient for proving that a given equation $s = t$ is a logical consequence of a set E of equations, provided that the completion of E stops after a finite amount of time and produces a confluent and terminating rewrite system R. In this case, $s = t$ is a logical consequence of E iff the R-normal forms of s and t are identical. That is to say, the preprocessing of E restricts the search space for proving a logical consequence to be linear.

Unfortunately, this preprocessing of E does often not stop after a reasonable (or even finite) amount of time. So Bachmair, Dershowitz and Plaisted ([BDP89]) have developed an unfailing Knuth-Bendix completion procedure that is sound and complete for proving $s = t$ to be a logical consequence of E. This method inherits the main advantage of the original completion procedure: It has strong simplification capabilities to keep the relevant information as small as possible. But, from the theorem proving point of view, it has the disadvantage of not being goal oriented. In principle, it generates all logical consequences of E until $s = t$ is generated. This results in a huge search space.

There are some general methods to deal with this problem. One can apply special strategies and heuristics for determining the next inference step and so to guide the search. The strategies are based on statistic measures while the heuristics can also use semantic measures (see [De93] and [AD93]). There are also some proposals for explicit goal-directed search (see [Bl86] and [CH93]), but these attempts, as far as they are implemented, were not very successful due to the following facts:

- intermediate results needed for the proof were not found,
- goals with almost no structure did not provide enough information for guiding the search,
- fixed sequences of inferences were repeated very often,
- completeness results under reasonable conditions were hard to establish.

We believe that it is hard to solve these problems by using one fixed search strategy or heuristic. Instead we believe that some progress can be achieved by following different search strategies in parallel in such a way that important intermediate results or intermediate system states can be exchanged and so some cross-fertilization becomes possible.

We have developed the team work method to distribute automated theorem provers and so to allow several experts (following their own search strategy) to work on the given problem. We have implemented this for pure equational logic using the unfailing Knuth-Bendix completion procedure as basic prover. There is a fixed schedule (set up by a supervisor) to activate a team of experts, to let them work independently (thus competing), then to exchange their most important results (to cooperate) and to start a new round. This architecture has the following advantages:

- restricted, but efficient control and communication,

- measurement of intermediate results and evaluation of experts,

- forgetting of "useless" intermediate results.

There are theoretical results ([AD93]) that allow for using "unfair" experts (which are unable to find solutions alone) without losing completeness of the whole system. This allows us to incorporate into a team goal oriented experts which are typically unfair. If such a goal oriented expert is well suited for the given problem it may produce valuable intermediate results to shorten the search of other experts or even complete the proof using results of the other experts. If it does not produce reasonable results, it's results are just forgotten and the expert may be replaced by another one.

In this paper we present three goal oriented experts which proved to be highly efficient in experiments. We describe them in detail, discuss how they work and demonstrate their power by examples. We present problems that we could not solve without these goal directed experts but which were easy to solve using them. One of these experts even proved to perform highly efficient as stand-alone prover. We compare our results with those produced (or sometimes not produced) by other well known provers, i.e. Otter 3.0 ([Mc94]) and REVEAL ([CA94]).

This paper is organized as follows: After this introduction we describe in section 2 the completion algorithm we use as basis for our goal oriented heuristics. Then we give a brief overview of the team work method and its application to completion (section 3). In section 4 we describe in detail the three goal oriented heuristics which were used to obtain the results of section 5. In this section 5 we will not only state the results, but we will also analyze why goal oriented heuristics allow one to get these results. In section 6 we will describe other approaches to goal oriented equational theorem proving. Finally, we will sketch possible further improvements of our method.

2 Equational deduction by completion

Equational theorem proving is concerned with the following problem:

Input: A set E of equations and a goal $s = t$ over a fixed signature sig
Question: Is $s = t$ a logical consequence of E ?

As usual, sig = (F,τ) consists of a set F of function symbols and a function τ denoting the arity of the elements of F. Here we restrict to the unsorted case, the extension to the many sorted case is simple. We denote by Term(F,V) the set of terms over F and a set V of variables. If s is a term and p a position in s then s/p denotes the subterm of s at position p and the term s[p$\leftarrow$t] results from s by replacing s/p with t.

A *reduction ordering* $\succ$ on Term(F,V) is a well-founded partial ordering on Term(F,V) that is compatible with substitutions and the term structure, i.e. s $\succ$ t implies $t_0[p\leftarrow \sigma(s)] \succ t_0[p\leftarrow \sigma(t)]$ for any term t_0, position p in t_0 and substitution σ. A *ground reduction ordering* is a reduction ordering that is total on the ground terms, i.e. terms without variables, denoted by Term(F). By $\equiv$ we denote the identity on Term(F,V).

The completion procedure is based on two basic inference steps: (1) generation of critical pairs (paramodulation) and (2) simplification of terms (demodulation) according to a fixed reduction ordering. To explain this, let $l_1 = r_1$ and $l_2 = r_2$ be two equations with no variables in common, let p be a position in l_1 such that l_1/p is no variable and $\sigma =$ mgu(l_1/p,l_2) exists. If neither $\sigma(r_1) \succ \sigma(l_1)$ nor $\sigma(r_2) \succ \sigma(l_2)$ then $(\sigma(r_1),\sigma(l_1[p \leftarrow r_2]))$ is a *critical pair* of these equations. We denote by CP(E_0) the set of critical pairs induced by the equations in E_0 wrt. $\succ$. If s is a term, l = r an equation and σ a substitution such that $s/p \equiv \sigma(l)$ and $\sigma(l) \succ \sigma(r)$, then $s \Rightarrow s[p \leftarrow \sigma(r)]$. We call $\Rightarrow \; = \; \Rightarrow_{E0}$ the *rewrite relation* of E_0 wrt. $\succ$. We call s reducible if $s \Rightarrow t$ for some term t. We call t a *normal form* of s if $s \Rightarrow^* t$ and t is irreducible. Here $\Rightarrow^*$ denotes the reflexive and transitive closure of $\Rightarrow$. An equation l = r is *interreduced* if l and r are irreducible.

In order to simplify notations and to save comparisions of terms wrt. $\succ$, we split the set E_0 of equations into the set $R = \{ l \rightarrow r \mid l = r \text{ in } E_0, l \succ r\}$ of rules (orientable equations) and the set $E = \{ l = r \mid l = r \text{ in } E_0, \text{neither } l \succ r \text{ nor } r \succ l\}$ of unorientable equations. Note that we have $\sigma(l) \succ \sigma(r)$ for each rule $l \rightarrow r$ and each substitution. If l = r in E then there may be substitutions σ_i such that $\sigma_1(l) \succ \sigma_1(r)$ and $\sigma_2(r) \succ \sigma_2(l)$.

We now describe our basic prover. It takes as input a set E_0 of equations, a goal $s = t$ and a reduction ordering $\succ$ that can be extended to a ground reduction ordering. It works on three sets: the set R of rules, the set E of unorientable equations and the set CP of critical pairs that are unprocessed. Initially, both R and E are empty and CP = E_0. Now the basic prover consists of the following loop.

basic prover:
while CP is not empty and the normal forms of s and t are not identical do
- choose an equation u' = v' from CP.
- let u and v be the normal forms of u' and v'.
- if not $u \equiv v$ then
  - if either $u \succ v$ or $v \succ u$ then
    - let l be max(u,v) and r be min(u,v).
    - interreduce R and E with $l \rightarrow r$.
    - $R := R \cup \{ l \rightarrow r\}$.
    - let CPnew be the set of critical pairs of $l \rightarrow r$ and R and $l \rightarrow r$ and E.

  else
    - interreduce R and E with u = v.
    - $E := E \cup \{ u = v\}$.
    - let CPnew be the set of critical pairs of u = v and R and u = v and E.
  - CP := CP $\cup$ CPnew.

There is one basic point of indeterminism within this prover: How to select the equation u' = v' from CP to be processed next. The way how this indeterminism is fixed determines the search for the proof by the prover (the so-called search plan of [BH91]). This allows one to integrate different search strategies and heuristics into the prover and so to realize different experts (see below). A search plan that guarantees to find a proof if there is one (i.e. a "fair" plan, see [BDP89] and [AD93] for the definition of fairness) will in the following be called a strategy while unfair search plans are heuristics. A general way to describe a strategy or heuristic is to define for each equation u = v in CP a value or weight (an integer) val(u = v) and to select the equation u = v in CP with minimal weight to be processed next. Ties are broken arbitrarily. We give some standard strategies by defining val as already used in [Hu80]. Here |s| denotes the length of the term s.

val(u = v) =	|u| + |v|	smallest sum strategy
val(u = v) =	MAX{|u|,|v|}	smallest maximum strategy

In section 4 we will present and discuss goal oriented heuristics (Note that the two strategies mentioned above are totally independent from the goal $s = t$ to be proved).

3 The team work method

The team work method (see [De93], [AD93]) is a general approach to distribute theorem proving procedures. It has been inspired by human project teams and has been used to distribute equational theorem proving by completion. A proof system based on team work models human project teams by use of multiple processes running on different processors.

A *team* consists of a single supervisor and a number of experts, each accompanied by a referee evaluating his work. Usually each expert is working on a problem without communication with the other team members. Only at *team meetings*, scheduled by the supervisor results are exchanged.

The *supervisor* is selecting the experts to work on a given problem, initially by judging their previous successes on related problems, later by using the referees' evaluation of their performance in dealing with the given problem. Also he determines the lengths of the working phases of the experts between the team meetings.

The *referees* are evaluating the achievements of the different experts. Their assessments are used for selecting a new team and important results (from their respective experts) during the team meetings.

The *experts* are the members of the team working directly on the problem. In our case each of them is using the completion algorithm as described in section 2. They differ in the methods used to choose the next critical pair. At team meetings, the system, i.e. the sets R, E and CP, of the best expert is chosen as the basis for further work. As only one system survives completely, the experts are competing for the best result.

However, competition is only one aspect of the team work method. The second important element is the cooperation between the experts. Cooperation is achieved by integrating outstanding results from inferior experts (as chosen by their referees) into the system of the best expert. This is accomplished by the supervisor during the team meetings, before he presents a new and updated problem description to the experts for the next working phase. In our case of a completion based theorem prover the outstanding results - rules and equations - are handled as new critical pairs to be processed immediately.

It is important to note that most of the results generated by the inferior experts are dropped or *forgotten*. We believe that one of the reasons for the success of team work is this feature of forgetting that avoids blowing up the search space. Furthermore, it allows the use of very specialized selection heuristics for critical pairs that are only capable of generating a few of the necessary results to prove a goal. But these results are generated as early as possible without many unnecessary steps. Our heuristics based on goal similarity are members of this category of experts.

Although goal oriented experts are in the center of this paper, we want to mention that also the other components of teams have goal oriented features. Firstly, referees can ground their judgement of intermediate results on goal similarity. Secondly, the supervisor can decide when to choose goal oriented experts to become team members. But in the rest of the paper we will concentrate on goal oriented experts.

Note that one can prove the completeness of the team work method under relatively weak conditions (see [AD93] for the definition of *team-fairness*. These conditions allow one to have unfair experts in the team (even as winner of a round), one only has to guarantee that in an infinite computation infinitely often a team-fair expert becomes the winner of a competition round. This is easy to achieve, since for example experts following the smallest sum strategy or the smallest maximum strategy are team-fair. For this reason one can activate unfair experts frequently (and design special experts for different proof stages) without losing completeness for the whole team.

4 Goal oriented selection heuristics for completion

In this section we present two kinds of goal oriented heuristics. As explained in section 2, these heuristics are used to select the next equation in CP to be processed by the basic prover. So they define an expert.

The first kind of heuristics is based on the notion of structural complexity of terms and equations. The idea behind the definition of these heuristics is the hypothesis that, given a fixed structural complexity measure, simple statements only need simple lemmata to be proved. So those equations in CP should be preferred whose structural complexity does not exceed that of the goal. Clearly, this hypothesis does not hold true in general, but it turns out to lead to good heuristics (as demonstrated by Table 1 in section 5).

The second kind of heuristics is based on the notion of similarity between terms. We define several notions of similarity between an equation and a goal and measure this similarity by a natural number. Now the heuristics prefer those equations in CP which have a high similarity to the goal. It turns out that this kind of heuristics performs well at the end of a proof.

4.1 The structural complexity expert occnest

Our heuristics based on structural complexity derive from the measures proposed in [AA90] to guide the search in a completion based prover. The measures of a term t in

[AA90] are based on the occurrences of a function symbol f in t, see the function occ(f,t) defined below as one example. This defines a value of an equation and this value is combined in a lexicographical way with other measures on the equation. According to the order induced by these measures the smallest equation is selected to be processed next. In this way the measures act as a sequence of filters.

Our experiments showed that these sequences of filters define an ordering that is not smooth enough. The combination of different values can be improved by using an arithmetic combination instead of a lexicographic one. We propose the combination of the following values: (1) ϕ(t), which is a modified length of t, (2) occ(f,t), which denotes the number of occurrences of f in t and (3) nest(f,t), which is equal to the maximum number of direct following occurrences of f on a branch of t, when t is presented as a tree. (2) and (3) lead to a value for each element of F. The combination of these values will define our goal oriented heuristic occnest(u=v). We make this precise.

Definition 4.1 (Weight of a term, occ, nest)

a) *The weight $\phi(t)$ of a term t is recursively defined by*

$$\phi(t) = 1, \text{ if } t \text{ is a variable,}$$
$$\phi(t) = 2 + \phi(t_1) + \ldots + \phi(t_n), \text{ if } t \equiv f(t_1,\ldots,t_n).$$

b) *The number of occurrences occ(f,t) is recursively defined by*

$$occ(f,t) = 0, \text{ if } t \text{ is a variable,}$$
$$occ(f,t) = occ(f,t_1) + \ldots + occ(f,t_n), \text{ if } t \equiv g(t_1,\ldots,t_n),\ f \not\equiv g,$$
$$occ(f,t) = 1 + occ(f,t_1) + \ldots + occ(f,t_n), \text{ if } t \equiv f(t_1,\ldots,t_n).$$

c) *The nesting nest(f,t) is recursively defined by*

$$nest(f,t) = 0, \text{ if } f \text{ is a constant}$$
$$nest(f,t) = hnest(f,t,0,0), \text{ if } f \text{ is not a constant, where}$$
$$hnest(f,t,cur,abs) = MAX(\{cur,abs\}), \text{ if } t \text{ is a variable or a constant,}$$
$$hnest(f,t,cur,abs) = MAX(\{hnest(f,t_i,0,MAX(\{cur,abs\})) \mid 1 \leq i \leq n\}), \text{ if } t \equiv g(t_1,\ldots,t_n),\ f \not\equiv g,$$
$$hnest(f,t,cur,abs) = MAX(\{hnest(f,t_i,cur+1,abs) \mid 1 \leq i \leq n\}), \text{ if } t \equiv f(t_1,\ldots,t_n)$$

To give an example, if t $\equiv$ f(g(a,x),f(a,g(b,y))), then ϕ(t) = 16, occ(f,t) = occ(g,t) = occ(a,t) = 2 and nest(f,t) = 2 and nest(g,t) = 1. For $u_1 \equiv$ f(x,y) and $u_2 \equiv$ f(a,y) we have $\phi(u_1) = 4$ and $\phi(u_2) = 5$. This way, an equation u_1 = v would be preferred over u_2 = v (using the smallest sum strategy and also in our goal oriented heuristics). The reason why we prefer this is that u_1 = v is more general and therefore more powerful for deduction than u_2 = v.

We will use ϕ(t) as the basic value of a term t and occ(f,t) and nest(f,t) for refinements to express the structure of t. Next we extend occ and nest to equations u = v by occ(f,(u,v)) := MAX({occ(f,u) , occ(f,v)})

nest(f,(u,v)) := MAX({nest(f,u) , nest(f,v)})

Finally we define the value occnest(u=v) according to the following idea: We start with the weight of the equation u = v and modify it by a penalty to describe the difference of the structural complexity of u = v and the goal $s = t$. There are several ways to combine these values. One easily sees that it is reasonable to add the penalty in the form of an factor. This factor has to be at least 1; this is ensured by using the function

ψ(x) = 1, if x$\leq$0,

ψ(x) = x + 1, else.

To be flexible, we also allow a set D $\subseteq$ F to describe which operators f in F should contribute to the value occnest(u=v).

Definition 4.2 (occnest)
Let $u = v$ be a critical pair, $s = t$ the goal and ϕ the weighting function for terms as defined earlier. Let furthermore $D \subseteq F$, where F is the set of all function symbols of sig. We define for all $f \in F$:

m_f := 1, if $f \notin D$,

m_f := ψ(occ(f,(u,v)) - occ(f,(s,t)) $*$ ψ(nest(f,(u,v)) - nest(f,(s,t))), otherwise.

Then we have :

$$occnest(u = v) = (\phi(u) + \phi(v)) * \prod_{f \in F} m_f.$$

Clearly, there are many other ways to define structural complexity and to relate the complexity of the goal $s = t$ to that of an equation. We have experimented with occnest and present some experimental results in section 5.

4.2 Experts based on goal similarity

There are many ways to define the notion "similarity". They differ in the operations under which the objects to compare should be similar. Although one could say that the measures of the last section can be used as a function to measure similarity of equations, in this section we are interested in the similarity of two equations with respect to the basic operations of completion, namely reduction and generation of critical pairs. Therefore we want to say that two equations are *similar* if they can be transformed into each other by means of several reductions or the generation of several critical pairs. Note that this definition depends on all other rules and equations that are known to be valid.

Now, if we know that an equation and a goal are similar with respect to this criterion then we already have the proof of the goal. Therefore we have to relax this criterion such that we try to estimate the possibility that such a proof can be found. This relaxation should lead to a function measuring similarity that only needs to know the two equations to compare and not any other rules and equations. We developed such functions by means of matching and weighting superfluous parts of terms.

In the following we will make this idea more precise. Because we have two basic operations, we also have two similarity functions. These functions, **Goal-in-CP** and **CP-in-Goal**, weight equations according to their similarity to a given goal which is assumed to be ground. First we need to define the similarity of a term u to a goal term s.

Definition 4.3 (Similarity of terms)
Let s be a ground term (goal term) and u a term.
The CP-in-Goal-Similarity sim_{CGp} at position p with substitution σ is defined by

$$sim_{CGp}(u,s,\sigma,p) = \phi(s) - \phi(s/p), \text{ if } \sigma(u) \equiv s/p$$

$$sim_{CGp}(u,s,\sigma,p) = \infty, \text{ otherwise.}$$

The CP-in-Goal-Similarity sim_{CG} with substitution σ is

$$sim_{CG}(u,s,\sigma) = MIN_p(sim_{CGp}(u,s,\sigma,p)).$$

The Goal-in-CP-Similarity sim_{GCp} at position p with substitution σ is defined by

$$sim_{GCp}(u,s,\sigma,p) = \phi(u) - \phi(u/p), \text{ if } \sigma(u/p) \equiv s \text{ and } \phi(u/p) \geq \text{min-struct},$$

$$sim_{GCp}(u,s,\sigma,p) = \infty, \text{ otherwise.}$$

The Goal-in-CP-Similarity sim_{GC} with substitution σ is

$$sim_{GC}(u,s,\sigma) = MIN_p(sim_{GCp}(u,s,\sigma,p)).$$

The reason for introducing the parameter *min-struct* is to prevent sim_{GC} from using positions in terms that are variables or terms of the form $f(x_1,...,x_n)$ which would be similar to all goals resp. all goals with top-level symbol f. By choosing min-struct $\geq$ n+3 this can be achieved (see the definition of ϕ).

If we want to extend our two similarity notions to an equation u = v and a goal $s = t$ then we encounter the following three cases (note that equations are symmetric, so that u and v can be exchanged) :

We use sim for either sim_{GC} or sim_{CG}.

I. There is a σ, such that sim(u,s,σ) $< \infty$ and sim(v,t,σ) $< \infty$.

II. There is a σ, such that either sim(u,s,σ) $< \infty$ or sim(v,t,σ) $< \infty$.

III. There is no σ, such that sim(u,s,σ) $< \infty$ or sim(v,t,σ) $< \infty$.

It is obvious that case I is the most desirable of these three. In order to achieve a distinction between these cases while nevertheless describing each critical pair with one value, we use in the following definition two factors, namely *single-match* and *no-match*, that are greater than 1. These factors can be considered as handicaps for the cases II and III. In our implementation we use single-match = 5 and no-match = 50 as default.

Definition 4.4 (Goal-in-CP , CP-in-Goal)
Let u=v be a critical pair and s = t be the goal.

$$\begin{aligned} \textit{CP-in-Goal}(u{=}v,s{=}t) = \mathit{MIN}_\sigma \quad & (sim_{CG}(u,s,\sigma) + sim_{CG}(v,t,\sigma), \\ & sim_{CG}(u,t,\sigma) + sim_{CG}(v,s,\sigma)), \end{aligned}$$

if there are σ with respect to case I.

$$\begin{aligned} \textit{CP-in-Goal}(u{=}v,s{=}t) = \mathit{MIN}_\sigma \quad & (sim_{CG}(u,s,\sigma) + \phi(t), \\ & sim_{CG}(u,t,\sigma) + \phi(s), \\ & sim_{CG}(v,s,\sigma) + \phi(t), \\ & sim_{CG}(v,t,\sigma) + \phi(s)) * \textit{single-match}, \end{aligned}$$

if there are σ with respect to case II.

$$\textit{CP-in-Goal}(u{=}v,s{=}t) = (\phi(u) + \phi(v)) * \textit{no-match},$$

otherwise.

$$\begin{aligned} \textit{Goal-in-CP}(u{=}v,s{=}t) = \mathit{MIN}_\sigma \quad & (sim_{GC}(u,s,\sigma) + sim_{GC}(v,t,\sigma), \\ & sim_{GC}(u,t,\sigma) + sim_{GC}(v,s,\sigma)), \end{aligned}$$

if there are σ with respect to case I.

$$\begin{aligned} \textit{Goal-in-CP}(u{=}v,s{=}t) = \mathit{MIN}_\sigma \quad & (sim_{GC}(u,s,\sigma) + \phi(v), \\ & sim_{GC}(u,t,\sigma) + \phi(v), \\ & sim_{GC}(v,s,\sigma) + \phi(u), \\ & sim_{GC}(v,t,\sigma) + \phi(u)) * \textit{single-match}, \end{aligned}$$

if there are σ with respect to case II.

$$\textit{Goal-in-CP}(u{=}v,s{=}t) = (\phi(u) + \phi(v)) * \textit{no-match},$$

otherwise.

As in the case of occnest, the heuristics Goal-in-CP and CP-in-Goal choose the critical pair with the resp. lowest value.

Note that although we use ϕ(s), resp. ϕ(t), in the definition of CP-in-Goal for the cases I and II, we define the weight of the critical pair in case III, if there is no similarity, by ϕ(u) and ϕ(v). Otherwise, each critical pair classified into case III would be given the same weight.

As we stated before presenting the definitions, we want to define similarity of critical pairs to the goal with respect to the operations reduction and generation of critical pairs. Goal-in-CP measures the similarity with respect to the generation of critical pairs and CP-in-Goal with respect to reductions. We will demonstrate this with the following example.

Example :
Let s $\equiv$ f(g(a),f(a,g(b))) = g(f(a,b)) $\equiv$ t be the goal.

1. Let u $\equiv$ g(f(f(g(x),f(y,z)),d)) = g(f(y,b)) $\equiv$ v be a critical pair and g(f(f(g(a),x),d)) $\rightarrow$ f(g(a),y) be a rule.
   If we assume that the critical pair is ordered from left to right then the now two rules have, besides others, the following critical pair :
   u' $\equiv$ f(g(a),f(y,z)) = g(f(y,b)) $\equiv$ v',
   which proves the goal.
   Although the terms u and v are quite big, Goal-in-CP weights u = v with a value of 6, which results in an early selection. Among the several critical pairs that can be generated from the two rules the pair u' = v' has a Goal-in-CP value of 0 and will therefore be processed immediately (Note that also the CP-in-Goal value of u' = v' is 0).

2. Let f(a,x) = f(x,b) and f(a,x) = f(a,b) be two critical pairs. They are nearly identical and would be weighted by ϕ with 10 and 11.
   CP-in-Goal makes a better distinction by weighting them with $5 * 14 = 70$ and $6 + 2 = 8$.

It is obvious that both heuristics based on similarity are highly specialized. Although there are some examples that can be proved using one of these heuristics working alone, their effectiveness lies in their use in teams. The general idea is that other experts generate a situation in which there are critical pairs that can be classified into case I or II for one of the heuristics. Then this heuristic is very often capable of finishing the proof very fast (see next section).

Note that a reduction of the goal should actually result in the necessity to recompute the weights for all critical pairs. However, as we mainly use these heuristics in teams we can neglect this fact, because during a team meeting this recomputation has to be done anyway.

5 Experiments

The problems the lack of goal orientation causes for completion can be best observed when one is trying to prove several theorems in a fixed domain of interest. The known (non-goal oriented) selection strategies for critical pairs generate for each theorem exactly the same sequence of steps as for other theorems that appear later in the enumeration done by the strategy. Even worse, if there are equations that are suspected to be unnecessary for obtaining a proof for a theorem then the consequences of these equations will nevertheless be generated. This is the reason why one is interested in goal orientation, because such equations (and their successors) would not be considered if the goal would be taken into account.

Therefore the usefulness of goal oriented selection heuristics can be best demonstrated by proving many theorems in such a domain of interest. We have chosen the domain of *lattice ordered groups* (see [KK74]) for our examples. After a short introduction to this domain, we will first demonstrate that our goal oriented heuristics, namely occnest, can solve many examples very fast, even several examples where standard strategies have tremendous difficulties.

For each theorem prover and nearly each domain there are theorems that are beyond the limits of the prover. This is also the case for our goal oriented heuristics and the lattice ordered groups. But we will demonstrate that the use of team work enables us to reach beyond the limits of all our sequential experts. We were able to improve run times significantly for problems that can be solved sequentially and even to prove theorems that could not be proved by any of our experts working alone. These theorems are very hard, indeed, which will be demonstrated also by results (or better the lack of) obtained with the provers REVEAL 1.0 (see [CA94]) and Otter 3.0 (see [Mc94]). Finally, we will also give a few examples from other domains where team work in combination with goal oriented heuristics has been successful.

Lattice ordered groups combine the axioms of two mathematical structures, namely lattices and groups, as the name suggests. So, we encounter for many theorems the problem mentioned above, that indeed not all axioms are needed for a proof. In order to axiomatize lattice ordered groups we need the operator f of arity 2 for the group operator, its neutral element 1 and an operator i of arity 1 denoting the inverse of an element. A lattice is based on a partial ordering $\leq$ and two binary functions l and u, the greatest **l**ower bound and the least **u**pper bound of two elements. The two functions l and u can be used to get rid of the partial ordering $\leq$ with the help of the definition

$x \leq y$ iff $l(x,y) = x$ or $x \leq y$ iff $u(x,y) = y$.

So, we get the following equational axiomatization for lattice ordered groups :

$f(f(x,y),z) = f(x,f(y,z))$	$f(1,x) = x$	$f(i(x),x) = 1$
$l(l(x,y),z) = l(x,l(y,z))$	$l(x,y) = l(y,x)$	$l(x,x) = x$
$u(u(x,y),z) = u(x,u(y,z))$	$u(x,y) = u(y,x)$	$u(x,x) = x$
$f(x,l(y,z)) = l(f(x,y),f(x,z))$	$u(x,l(x,y)) = x$	$f(l(x,y),z) = l(f(x,z),f(y,z))$
$f(x,u(y,z)) = u(f(x,y),f(x,z))$	$l(x,u(x,y)) = x$	$f(u(x,y),z) = u(f(x,z),f(y,z))$

In Table 1 and Table 2 we use the theorems listed below. As there are two ways to eliminate $\leq$, we derive two problem sets, one using the operator l, denoted by the suffix .a, and one using the operator u, denoted by the suffix .b. (Because p10 does not contain $\leq$, we get only one example from this theorem.) As usual, the theorems must be negated and skolemized, conditions are added to the set of axioms. (More theorems can be found in [Fu94].)

Problems:

mono1:	$f(x,z) \leq f(y,z)$, if $x \leq y$	mono2:	$f(z,x) \leq f(z,y)$, if $x \leq y$
p1:	$f(i(z),f(x,z)) \leq f(i(z),f(y,z))$, if $x \leq y$	p2:	$x \leq y$, if $i(y) \leq i(x)$
p3:	$f(x,z) \leq f(y,w)$, if $x \leq y$, $z \leq w$	p4:	$1 \leq f(x,y)$, if $1 \leq x$, $1 \leq y$
p6:	$1 \leq f(i(x),f(y,x))$, if $1 \leq y$	p10:	$i(u(x,y)) = l(i(x),i(y))$
p9:	$l(x,f(y,z)) = l(x,z)$, if $1 \leq x$, $1 \leq y$, $1 \leq z$, $1 \leq l(x,y)$	p8:	$l(x,f(y,z)) \leq f(l(x,y),l(x,z))$, if $1 \leq x$, $1 \leq y$, $1 \leq z$

The run-times given below were obtained on SUN-ELC workstations, the team runs on a network of two such workstations.

Table 1 compares for most of the examples the run-time of occnest with the run-time of the best so-called standard strategy. Standard strategies are based on the term weight ϕ. Since a critical pair has two sides, there are several possibilities to combine the ϕ-values of these sides, sum them up (the Smallest-Component strategy of [Hu80]), use the maximum of the sides or use the weight of the side that is bigger with respect to the used reduction ordering (see section 2). We call these strategies standard, because they do not use any interpretations of symbols and can therefore be used without changing parameters for all possible examples.

Table 1 shows that goal orientation as defined by occnest is able to solve most of the examples dramatically faster than the standard strategies. Even several examples that took over 3 hours run-time with the standard strategies (and were therefore stopped, because we set this as time limit) can be solved in less than 5 seconds.

Example	**occnest**	**std. strategy**	**Example**	**occnest**	**std. strategy**
mono1.a	0.045	45.639	mono1.b	0.045	46.668
mono2.a	0.030	43.995	mono2.b	0.030	44.917
p1.a	0.272	—	p1.b	0.281	—
p3.a	4.135	—	p3.b	2.547	—
p4.a	1.840	32.437	p4.b	1.712	9.263
p6.a	0.388	—	p6.b	0.157	160.049
p9.a	19.568	209.102	p9.b	50.953	207.176

Table 1: run-time comparison occnest *vs* standard strategies (in seconds)

But, as it is the case with all heuristics, there are examples that also occnest can not solve alone. Table 2 shows that most of these examples can be solved (quite fast!) using team work and our goal oriented heuristics. We want to mention that the best (and only) sequential heuristic that was able to prove p2.a was Goal-in-CP, showing that also our heuristics based on goal similarity can solve examples working alone, but only very few. This is the reason why we did not include these heuristics in Table 1.

Example	**team**	**goal heuristic used in team**	**best seq. heuristic**	**REVEAL**	**Otter 3.0**
p2.a	5.413	occnest	79.516	591.06	17
p2.b	5.381	occnest	—	592.08	16
p8.b	56.837	Goal-in-CP	—	—	sos-empty
p9.a	8.659	occnest	19.568	—	sos-empty
p9.b	8.440	occnest	50.953	—	sos-empty
p10	23.203	Goal-in-CP	—	—	—

Table 2: run-time comparison team *vs* sequential experts *vs* REVEAL *vs* Otter 3.0 (in seconds)

Table 2 shows that team work enables us to prove more examples than our sequential system can. In order to show that the reason for this is not a weak implementation of unfailing completion we tried to prove these examples using the REVEAL 1.0-system and using the new Otter 3.0-system. Because both systems allow various parameter adjustments, we used the default settings of REVEAL and the auto-mode of Otter. One should note that REVEAL allows completion modulo the theory AC (which is not the case for Otter or our system) and the run-times reported in Table 2 were obtained using AC-completion. Without AC-completion REVEAL was not able to prove any of the examples of Table 2. Otter's very sophisticated auto-mode encountered several problems while trying to prove examples p8.b, p9.a and p9.b. Otter restrained itself to only generating critical pairs that had a limited size which resulted in emptying the set-of-support list and therefore in termination without success.

The teams we used consisted of two experts, a standard strategy and a goal oriented heuristic. For the domain lattice ordered groups occnest and Goal-in-CP were useful (as indicated by the column 'goal heuristic used ...'). Table 3 shows that in other domains also CP-in-Goal was useful, and there are even examples (namely luka3) that can be proved using a team consisting of goal oriented heuristics only.

Example	**team**	**goal heuristic used in team**	**best seq. heuristic**
bool5b	72.859	Goal-in-CP	—
ra2	125.373	CP-in-Goal	227.876
sa2	10.745	Goal-in-CP	—
herky3	6.811	occnest	16.091
luka3	81.680	Goal-in-CP, CP-in-Goal	—

Table 3: run-times for teams using goal oriented heuristics in other domains (in seconds)

Example bool5b states that in a boolean ring the associativity axioms are redundant. Example ra2 is taken from [LW92], sa2 from [BH93] and herky3 from [Zh93]. Example luka3 uses the equational axiomatization of the propositional calculus given in [AD93] to prove that $(\mathrm{not}(x) \rightarrow \mathrm{not}(y)) \rightarrow (y \rightarrow x)$.

In order to explain the success of our goal oriented heuristics we used the proof presentation tools developed in [DS94]. An analysis of the runs with which the results of Table 1 were obtained showed that the better performance of occnest is due to a faster selection of the results needed for a proof. As stated before, standard strategies have a very limited view on a problem and therefore many unnecessary critical pairs were selected. Although our analysis showed that occnest also does select many critical pairs that are not needed in the proof the ratio between needed and not needed pairs is much higher than that of standard strategies.

The even bigger success of team work can be explained by two typical behaviours our teams showed. Teams in which occnest is a team member show the characteristic that in most team meetings occnest is rated as the best expert. Due to the general problem that there are examples that need equations for a proof that a goal oriented heuristic does not rate good although they are in the list of critical pairs, occnest alone can not find a proof (or needs quite some time for it). But in combination with a standard strategy, that does rate these equations good and therefore allows a referee to judge the impact of these equations on the problem, success can be achieved. A typical behaviour of such teams is that during the team meetings always a few equations of the standard strategy are used to improve the system of the winner occnest until occnest is able to finish the proof.

Teams that use Goal-in-CP or CP-in-Goal show a different behaviour. In the first team meetings the used goal similarity heuristic generates nothing useful. But, also in the later team meetings the standard strategy expert gets better ratings by the referees. Nevertheless, there is always a team meeting after which the goal similarity heuristic, using the system of the other expert, can finish the proof. So for these teams the standard strategy is always the winner, but the goal oriented heuristic completes the proof. As mentioned earlier, this is exactly the behaviour we did want to achieve with Goal-in-CP and CP-in-Goal.

Finally we want to mention that one could be led to the assumption that team work is not necessary to achieve the results of Tables 2 and 3. One other way could be to change the used selection heuristic during a sequential run (for examples where the goal similarity heuristics were useful). A second possibility could be a new heuristic combining a standard strategy and a goal oriented heuristic. Both suggestions can not be realized. There is no way to tell during a run whether a change of the selection heuristic should be made. The important point in the typical behaviour of teams with goal similarity heuristics is that after each team meeting such a change is tried (by means of distribution) and its success is determined in the next (or better before the next) team meeting.

We can not prove that there is no means by which the second suggestion can be realized, but from the definitions of our goal oriented heuristics it should be obvious that we tried to embed as much as possible the notion of term weight (which is the basis of the standard strategies) into them. Trying to mix the behaviour of a standard strategy and a goal oriented heuristic leads in nearly all cases to selection heuristics that behave worse than both "parents".

6 Related Work

A first attempt to develop goal oriented equational deduction was documented by Bläsius in [Bl86]. The general idea was to start with the two sides of the goal and then to use the given equations to reduce the difference (in the term structure) of these two terms. This was repeated until an equational chain between the terms was constructed. But the few examples where this approach was useful can be proved easily by completion (without goal orientation). Without a concept for reduction one could observe that many steps were repeated very often. Moreover, there are many examples where a proof can only be found using equations that do not reduce the difference between the sides of the goal, but instead increase the difference before all differences can be reduced by other equations.

The approach of Cleve and Hutter has to face similar problems (see [CH93]). Although they introduce several notions of difference between terms (and heuristics when to use which one), they are bound to reduce with each step at least one of them. As we know at the moment of no implementation, no comparisons of this approach to completion or to our work are possible. The lack of a concept for redundancy elimination (i.e. term rewriting) in their approach seems to favour completion. Nevertheless, the different notions of difference between terms could be very useful for developing goal oriented selection heuristics for critical pairs. Furthermore, a combination with standard completion techniques using our team work concept would also overcome the lack of any completeness results for the approach of Cleve and Hutter. The examples sketched in [CH93] suggest that the approach can also be used with sets of conditional equations, that will be very interesting in the future.

The measure approach of [AA90] was already presented in section 4. Here we want to point out our modifications, namely the new measure nest and the combination of various measures and ϕ into one selection function based on the comparison of one value for each critical pair.

Finally, there is an approach to goal oriented completion by Socher (see [So91]) that tries to incorporate the goal into the inference rules of completion. Although this set of inference rules can be shown to be complete, the practical problem which inference rule to use next remains. It is not clear whether the standard selection strategies have the same success for these inference rules as in the case of completion.

7 Conclusion

We presented several goal oriented selection heuristics for critical pairs that can be used to incorporate goal orientation into equational deduction by unfailing completion. The heuristics ranged from methods comparing statistical aspects of goal and critical pairs to methods applying similarity criteria.

The team work method for distributing deduction processes allowed us to combine these goal oriented heuristics with standard selection strategies. This resulted in a system that combines the advantage of working goal oriented and thus reducing the search space with the advantages of completion, namely minimal representation of the actual problem state and the generation of strong rules (independent of the goal). Even completeness of this combination can be ensured. So, none of the known disadvantages of goal oriented equational deduction remain.

We demonstrated the success of our approach with many examples from the domain lattice ordered groups and also with examples from other domains. The important observation was that the distributed system is able to prove several examples that can neither be proved by goal oriented heuristics nor by other selection strategies working alone.

Further improvements of our results seem to be possible in two directions. Firstly, the supervisor can be improved to enable the system to determine which goal oriented heuristic should be used in the team. The analysis of section 5 suggests that occnest is useful from the beginning of a proof attempt on, while CP-in-Goal and Goal-in-CP have a better usage in later stages of a proof.

The second improvement is the development of other goal oriented heuristics. Besides the ideas of Cleve and Hutter there is also the possibility to develop heuristics that concentrate on only a few stages of a proof (leaving the other stages for their colleagues in the team). Goal-in-CP and CP-in-Goal are a first step in this direction. The prerequisite for the use of such heuristics is a good selection of team members by the supervisor.

References

[AA90] **Anantharaman, D. ; Andrianarievelo, N.:** *Heuristical criteria in refutational theorem proving,* Proc. DISCO '90, LNCS 429, 1990, pp. 184-193.

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

[BDP89] **Bachmair, L., Dershowitz, N., Plaisted, D.A.:** *Completion without Failure,* Coll. on the Resolution of Equations in Algebraic Structures, Austin (1987), Academic Press, 1989.

[BH91] **Bonacina, M.P. ; Hsiang, J. :** *On fairness of completion-based theorem proving strategies,* Proc. 4th RTA, Como, LNCS 488, 1991, pp. 348-360.

[BH93] **Bonacina, M.P. ; Hsiang, J. :** *The clause diffusion methodology for distributed deduction,* Proc. DISCO '92, LNCS 722, 1993, pp. 272-287.

[Bl86] **Bläsius, K.H.:** *Equality reasoning based on graphs,* Ph.D. thesis, University of Kaiserslautern, 1986.

[CA94] **Chalin, J. ; Anantharaman, S. et al. :** *REVEAL - a user's guide,* Tech. rep. LIFO.94-12, University of Orleans, 1994.

[CH93] **Cleve, J. ; Hutter, D. :** *Guiding equational proofs by attribute functions,* SEKI-Report SR-93-15, University of Saarbrücken, 1993.

[De93] **Denzinger, J.:** *Teamwork : A method to design distributed knowledge based theorem provers (in German),* Ph.D. thesis, University of Kaiserslautern, 1993.

[De93] **Denzinger, J. ; Schulz, S.:** *Analysis and Representation of Equational Proofs Generated by a Distributed Completion Based Proof System,* SEKI-Report (to appear).

[Fu94] **Fuchs, M.:** *The application of goal-oriented heuristics for proving equational theorems via the unfailing Knuth-Bendix completion procedure. A case study: lattice ordered groups,* SEKI-Report SR-94-02, University of Kaiserslautern, 1994.

[Hu80] **Huet, G.:** *Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems,* J. of ACM 27, No. 4, 1980, pp. 798-821.

[KB70] **Knuth, D.E. ; Bendix, P.B.:** *Simple Word Problems in Universal Algebra,* Computational Algebra, J. Leech, Pergamon Press, 1970, pp. 263-297.

[KK74] **Kokorin, A.I. ; Kopytov, V.M. :** *Fully ordered groups,* Halsted Press, 1974.

[LW92] **Lusk, E. ; Wos, L. :** *Benchmark problems in which equality plays the major role,* Proc. CADE-11, LNAI 607, 1992, pp. 781-785.

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

[So91] **Socher, R.:** *A Goal Oriented Strategy Based on Completion,* Tech. rep. TR#91/18, SUNY at Stony Brook, 1991.

[Zh93] **Zhang, H.:** *Automated proofs of equality problems in Overbeek's competition,* JAR 11, 1993, pp. 333-351.