Guiding equational proofs by attribute functions

Technical Report

Jürgen Cleve*
University of the Saarland
Department of Computer Science
P.O.Box 1150
D-66041 Saarbrücken
Federal Republic of Germany

Dieter Hutter †
German Research Center for
Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3
D-66123 Saarbrücken
Federal Republic of Germany

July 1993

Content Areas : Deduction, Automated Reasoning, Equational Reasoning, Difference Reduction

Abstract

This report presents a methodology to guide equational reasoning in a goal directed way. Suggested by rippling methods developed in the field of inductive theorem proving we use attributes of terms and heuristics to determine bridge lemmas, i.e. lemmas which have to be used during the proof of the theorem. Once we have found such a bridge lemma we use the techniques of difference unification and rippling to enable its use.

*Tel. (+49 681) 302 5316, email: cleve@dfki.uni-sb.de
†Tel. (+49 681) 302 5317, email: hutter@dfki.uni-sb.de

1 Introduction

Automated theorem provers suffer from their inability or inefficiency in solving problems. This effect is especially the case if equality is included.
The most promising approach to mechanize equality reasoning stems from Knuth and Bendix [KB70] who restricted the application of equations (by using them as rewrite rules) and formulated their completion calculus which led to an considerable decrease of the search space. The original completion procedure of Knuth and Bendix is restricted to directable unit equations. This strong restriction was weakened by Bachmair, Dershowitz, Plaisted [BDP87] introducing unfailing completion. Completion was "lifted" to an complete first-order calculus by Zhang, Kapur, Bachmair and Ganzinger [ZK88], [BG90].
One important disadvantage of the completion-based approaches is their forward reasoning character. A goal-oriented style of completion-based provers could only be simulated by cutting off parts of the search space [AA90], [SA92] or by structuring the search space in dependence of the goal [Den91].
A constructive goal-directed technique to control equational reasoning is the approach of "difference reduction". The general idea is to find a sketch of the proof and then, to tackle the resulting subproblems [NSS59]. Now, the problem arises how to break down a proof in such a hierachical manner. The existing approaches of RUE-resolution [Dig79] and the calculus of Bläsius [Blä86] do this in a simple syntactical way. In order to equalize two terms first, the top-level symbols have to be equalized and then, the corresponding arguments of both terms have to be adapted.
While these two general approaches of difference reduction were no success another goal-oriented approach called rippling / colouring terms [Bun88], [Hut90] has been developed in the field of inductive theorem proving. This technique turned out to be very successful to enable the use of the induction hypothesis inside the induction conclusion. In a first step the syntactical differences between hypothesis and conclusion are shaded. E.g. we obtain a formula $\Phi(\boxed{p(x)})$ as a hypothesis and $\Phi(x)$ as a conclusion. Furthermore, the syntactical differences between both sides of the given equations are shaded. Depending on the locations of the shaded areas inside the white expressions we call these (C-)equations context-moving (shaded areas on both sides), context-creating or context-deleting (shaded areas only on one side). Now, using these C-equations we are able to move, insert, or delete shaded areas

within the conclusion in order to manipulate the conclusion. Basin and Walsh [BW92] developed a method to determine syntactical differences of two arbitrary terms by so-called difference matching or difference unification which now allows to extend the scope of rippling to general theorem proving. The aim of this paper is to develop generalisations of the mentioned difference-reduction techniques in order to find appropriate bridge lemmas which have to be used during the proof. In other words we search for "interesting" equations and try to enable their application. This is done by introducing properties on terms which are computed by so-called *attribute functions* ω. Furthermore, we label those axioms which change the ω-value of a term after being applied to it. Suppose, both sides of an equation to be proved disagree in their ω-value. Thus we have to use one of the labeled equations during the proof. Selecting appropriate attribute functions in dependence of the actual problem reduces the set of labeled axioms one of which has to be used to prove the theorem. In order to enable their application we use difference unification and rippling which manipulate the goal to apply a dedicated axiom.

Throughout this paper we use paramodulation / resolution as the underlying calculus. Hence, we prove a theorem by refuting its skolemized negation. We call a negated equation (to be proven) a *goal equation*.

The techniques presented here are suitable for guiding the equational part of the deduction process. Of course they have to be combined with strategies for guiding the resolution process. Thus, handling conditional equations fits in the presented framework.

We want to emphasize that the presented approach is not restricted to the paramodulation calculus but we use it as an exemplary one to illustrate our approach. [1]

[1] We use the traditional notation for terms, i.e. $\mathcal{T}(\Sigma, \mathcal{V})$ is the set of terms w.r.t. the signature Σ and the set of variables $\mathcal{V}$, $x, y, z, \ldots$ denote variables, $a, b, c, \ldots$ constant symbols, and $f, g, h, \ldots$ function symbols. By $s[\pi \leftarrow r]$ we denote the replacement of the term s at the position π by r. We use $\Sigma(t)$ to denote the set/multiset of all symbols occurring in t, $\mathcal{V}(t)$ to denote all variables of t and $\Delta(t) = \Sigma(t) \cup \mathcal{V}(t)$. We use *substitutions* not only on terms but also on symbols of the signature (in which case they are just the identity for non-variable symbols.)

2 An example

As mentioned in the introduction we try to solve equality problems with the help of difference reduction. Given an equational problem $\neg\ s = t$ (goal equation) we look for an equation $q = r$ (as a *bridge lemma*) out of the set of given formulas which has (likely) to be used during the proof. The selection of this equation is guided by syntactical properties of the given terms and several heuristics (for instance, concerning the origin of the axioms). In this section we shall illustrate this approach by an example. We use a straightforward equality representation.
Suppose, we want to characterize some aspects of the arithmetic of integers which results in the following set of axioms:

Example 2.1

$$\forall x : x - x = 0 \tag{1}$$
$$\forall x : x - 0 = x \tag{2}$$
$$\forall x : x + 0 = x \tag{3}$$
$$\forall x, y, z : x - (y + z) = (x - z) - y \tag{4}$$
$$\forall x, y, z : x + (y + z) = (x + y) + z \tag{5}$$
$$\forall x, y : x + y = y + x \tag{6}$$
$$\forall x, y, z : (x + y) - z = x + (y - z) \tag{7}$$

The task is to prove $\forall x, y, z : (x + z) - (y + z) = x - y$. By negating and skolemizing we obtain the goal equation (a, b, c skolem constants) :

$$\neg\ (a + c) - (b + c) = a - b \tag{8}$$

In a first step we compare the constants of both sides of the goal equation. Since c occurs only on the left-hand side we have (definitely) to apply an equation which removes c in order to transform the left-hand side to the right-hand side. The only way to remove c from the left-hand side of the goal equation is to use equation (1) from left to right on the left-hand side. In a second step we try to enable the use of this selected equation (1). Hence, we look for the syntactical differences between the terms $(a + c) - (b + c)$ and $x - x$ and try to minimize these differences. For this purpose we use the notions of difference unification and difference matching [BW92]. In

contrast to [BW92] we restrict difference unification to solutions which do not shade any non-variable symbol of the equation to be applied. I.e. we obtain an instance of the left-hand side of the equation if we remove the shaded expressions inside the term to be manipulated.
Using this kind of difference unification we shade the syntactical differences between $(a+c)-(b+c)$ and $x-x$ and obtain the following coloured version of the left-hand side of the goal equation:

$$(\boxplus{a}{c} - \boxplus{b}{c}) \tag{9}$$

In a next step we use rippling techniques in order to move the shaded areas outside of all white expressions without changing the white expressions.
Using a coloured version of equation (7) results in

$$\boxplus{a}{(c - \boxplus{b}{c})} \tag{10}$$

Applying equation (4) we obtain

$$\boxplus{a}{\boxminus{(c - c)}{b}} \tag{11}$$

Now equation (1) is applicable

$$a + (0 - b) \tag{12}$$

and we succeeded in removing c from the term $(a + c) - (b + c)$.
Next, the same process is started recursively. Again, we compare the constants of $a + (0 - b)$ and $a - b$ and determine 0 to be removed.
Candidates for doing this are equations (1), (2), (3). Now we use heuristics to select an appropriate equation. Equation (1) would introduce the subterm $x - x$ which is unifiable with the term $c - c$ of term (11) (which we just removed) so we won't select equation (1). Difference unification (wrt. our restriction above) of $a + (0 - b)$ and $x - 0$ (equation 2) fails, so we use equation (3) and get via difference unification the shaded term

$$(a + \boxminus{0}{b}) \tag{13}$$

Rippling proposes equation (7) and we obtain

$$\boxminus{(a + 0)}{b} \tag{14}$$

which enables the use of equation (3). Finally, we obtain the new goal equation $\neg\ a - b = a - b$ and thus, the theorem is proved.

Summing up the previous example, we controlled the prover by :

- selecting appropriate bridge lemmata by syntactical properties of the goal equation (in combination with heuristics), determining the direction in which the equation has to be applied and determining one term of the two terms of the goal equation (which has to be manipulated to enable the application of the bridge lemma)
- determining the syntactical differences of the term (which we have to manipulate to enable the selected equation) and the "left" side of the equation to be applied,
- enabling the application of the equation by rippling techniques controlled by the syntactical differences determined by the previous step.

3 Formal Definitions

As seen in the previous example we are interested in attributes of terms which allow us to determine so-called bridge lemmas, i.e lemmas which have to be used during the proof. In general, an attribute is a function ω on terms which satisfy special requirements. In order to refute the goal equation $\neg\ s = t$ we have to use equations in the set of axioms to manipulate the goal until we obtain a formula $\neg\ r = r$. Now suppose, $\omega(s) \neq \omega(t)$ holds. Thus, there must be a paramodulation step which changes the value of the left or right side of our goal since obviously $\omega(r) = \omega(r)$ holds. We are interested in that deduction step (and the used equation) which coincides the ω-values of both sides. With the help of the special requirements of the so-called *attribute function* ω we will be able to trace back this dedicated step to the use of an equation $q = r$ with $\omega(q) \neq \omega(r)$ which we call an ω-*changing* equation. We call an equation which is not ω-changing an ω-*invariant* equation and extend this notion to negated equations.
We use attribute functions in terms of a resolution calculus with paramodulation. In order to prove an equation it is negated and skolemized. We call this resulting negated equation $\neg\ s = t$ the **goal equation**. We assume in the following that the goal equation is ground (i.e. there are no existentially quantified variables in the theorem).
We now give a formal definition of the special requirements mentioned above. They ensure that (if the goal equation is ω-changing) at least one ω-changing equation has to be applied.

We have to guarantee that the application of an ω-invariant equation to

- an ω-invariant equation results in an ω-invariant equation,
- the goal equation (or an antecedent of it) leads to an ω-changing equation.

Using an equation $q = r$ we have to instantiate it by a substitution σ. Hence, using an ω-invariant equation we have to ensure that $\sigma(q) = \sigma(r)$ is ω-invariant too. Precisely, we define:

Definition 3.1 *Let $\mathcal{W}$ be a set.*
A function $\omega : \mathcal{T}(\Sigma, \mathcal{V}) \to \mathcal{W}$ is **substitution-stable** *iff*
for all $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$ and for all substitutions σ :
$\omega(s) = \omega(t)$ *implies* $\omega(\sigma(s)) = \omega(\sigma(t))$.

Using an ω-invariant equation $q = r$ we replace a subterm $\sigma(q)$ of the goal by a term $\sigma(r)$. Thus, we also demand that replacing a subterm by a term with same ω-value does not change the ω-value of the whole term:

Definition 3.2 *Let $\mathcal{W}$ be a set.*
A function $\omega : \mathcal{T}(\Sigma, \mathcal{V}) \to \mathcal{W}$ is **subterm-stable** *iff*
for all $q, r, t \in \mathcal{T}(\Sigma, \mathcal{V})$: $\omega(q)=\omega(r)$ and $t|\pi=q$ implies $\omega(t)=\omega(t[\pi \leftarrow r])$.

Both, substitution-stability and subterm-stability form the requirements of attribute functions:

Definition 3.3 *A substitution-stable and subterm-stable function is called an* **attribute function**.

The next lemma states the *soundness* of our defined requirements, i.e. paramodulation of an ω-invariant goal equation with an ω-invariant equation results again in an ω-invariant (negated) equation.

Lemma 3.1 *Let ω be an attribute function and $q=r$, $\neg\ s=t$ be ω-invariant. If σ is a unifier of $s|\pi$ and q then $\neg\ \sigma(s)[\pi \leftarrow \sigma(r)] = \sigma(t)$ is again ω-invariant.*

Proof 3.2 Since ω is substitution-stable also $\sigma(q) = \sigma(r)$ and $\neg\ \sigma(s) = \sigma(t)$ are ω-invariant.
Thus, $\omega(\sigma(s)[\pi \leftarrow \sigma(r)]) = \omega(\sigma(s)[\pi \leftarrow \sigma(q)]) = \omega(\sigma(s)) = \omega(\sigma(t))$ □

The next lemma states the *completeness* of the requirements, i.e. we have to apply at least one ω-changing equation. Here we have to assume that the (goal) equation we want to manipulate and all its instances are ω-changing. Assuming the goal equation to be ground this holds trivially if the goal equation is ω-changing. But we could not guarantee that free variables are not introduced during the proof into the goal equation. So we prove that applying an ω-invariant equation to an equation which is ω-changing (independent of the applied substitution) leads to an equation which is ω-changing for all substitutions too.

Lemma 3.3 *Let ω be an attribute function, $q{=}r$ ω-invariant and $\sigma(s){=}\sigma(t)$ be ω-changing for all substitutions σ. If ρ is a unifier of $s|\pi$ and q then $\varphi(\rho(s[\pi \leftarrow r])) = \varphi(\rho(t))$ is ω-changing for all substitutions φ.*

Proof 3.4 Let φ be an arbitrary substitution.
First we note that $\varphi(\rho(s)) = \varphi(\rho(t))$ is ω-changing.
With the substitution-stability of ω we get : $\omega(\varphi(\rho(q))) = \omega(\varphi(\rho(r)))$.
So we get $\omega(\varphi(\rho(s))) = \omega(\varphi(\rho(s[\pi \leftarrow q]))) = \omega(\varphi(\rho(s))[\pi \leftarrow \varphi(\rho(q))]) =$
$= \omega(\varphi(\rho(s))[\pi \leftarrow \varphi(\rho(r))])$ (because of the subterm-stability)
$= \omega(\varphi(\rho(s[\pi \leftarrow r]))).$ □

4 Examples of Attribute Functions

In this section we first give examples of attribute functions and then, we show how to use attribute functions to guide equational theorem proofs.

4.1 Attribute functions based on symbol disagreement

As seen in the proof of the example in section 2 an obvious idea to get appropriate properties to guide the proof of an goal equation is to examine the "symbol differences" of its both sides (which we call **symbol disagreement**). Let us illustrate this idea by another

Example 4.1 ([Pel86], No. 64) Given the axioms

$$\forall x : 0 + x = x \tag{15}$$
$$\forall x : (-x) + x = 0 \tag{16}$$
$$\forall x, y, z : (x + y) + z = x + (y + z) \tag{17}$$

we have to refute (c, b are skolem constants) :

$$b + c = 0 \tag{18}$$
$$\neg\ c + b = 0 \qquad \text{Goal equation} \tag{19}$$

Comparing the symbols of both sides of (19) we have to remove b and c on the left side in order to equalize them. Obviously we have to apply (18) or (16) because these are the only equations which are able to delete occurrences of b or c.
The question arises how to characterize these symbol disagreement with the help of attribute functions.
The idea is to collect all symbols of a term as a multiset and to characterize equations by comparing the symbol multisets of the two terms. Obviously ω would not fulfill the restriction of being substitution-stable if we only collect those symbols in which the goal terms differ. Therefore, we have to extend the definition of ω by the collection of variables.
First, we need a notion for the *symbol disagreement* of the goal terms.

Definition 4.1
Let $\Delta(t)$ be the multiset of all symbols (incl. variables) occurring in the term t. Then, the symbol disagreement Ψ of two terms s and t is defined by

$$\Psi(s,t) = (\Delta(s) \backslash \Delta(t)) \ \cup \ (\Delta(t) \backslash \Delta(s)).$$

$\cup$ and $\backslash$ denote the multiset union and difference.

Let $\neg\ s = t$ be a goal equation. Based on the notion of symbol disagreement we use a subset $M \subset \Psi(s,t)$ to construct an attribute function ω.

Definition 4.2 Attribute function ω_M
Let M be a multiset, $\mathcal{W}$ be the set of all multisets over $\Sigma \cup \mathcal{V}$ and $v \in \mathcal{T}(\Sigma, \mathcal{V})$. The function $\omega_M : \mathcal{T}(\Sigma, \mathcal{V}) \rightarrow \mathcal{W}$ is defined by
$\omega_M(v)$=$\{x \in \Delta(v) |$ *there is a substitution σ and $y \in M$ such that* $\sigma(x)$=$\sigma(y)\}$

Assume the goal equation $\neg\ s = t$ is ground. Then the sets $\Delta(s)$ and $\Delta(t)$ are ground too. Thus, the set $\omega_M(v)$ (for nonempty M) is the multiset of all constant symbols of v occurring in M and of all variables of v.
Note that the function ω_M is independent of M being a multiset or a set, i.e. we get the same ω_M for $M = \{a, a\}$ and $M = \{a\}$.

Lemma 4.1 *Let M be non-empty. Then ω_M is an attribute function.*

Proof 4.2 Let $s, t \in \mathcal{T}(\Sigma, \mathcal{V})$, $\omega_M(s) = \omega_M(t)$.
First we note that $\omega_M(s) = \omega_M(t)$ implies that the multiset $\mathcal{V}(s)$ of all variables of s is equal to the multiset $\mathcal{V}(t)$. This makes the substitution-stability obvious.
Let $u \in \mathcal{T}(\Sigma, \mathcal{V})$ and $u|\pi = s$. We show the subterm-stability by induction.
If $\pi = \lambda$ we have $u = s$ and $\omega_M(u) = \omega_M(s) = \omega_M(t) = \omega_M(u[\lambda \leftarrow t])$.
If $\pi = i.\pi'$ and $u = f(t_1, \ldots, t_n)$ we have $\omega_M(u) = \omega_M(u[i.\pi' \leftarrow s]) =$
$\overline{\omega_M}(f) \cup \omega_M(t_1) \cup \ldots \cup \omega_M(t_{i-1}) \cup \omega_M(t_i[\pi' \leftarrow s]) \cup \omega_M(t_{i+1}) \cup \ldots \cup \omega_M(t_n)$
(where $\overline{\omega_M}$ denotes the canonical extension of ω_M to Σ).
By the induction hypothesis we have $\omega_M(t_i[\pi' \leftarrow s]) = \omega_M(t_i[\pi' \leftarrow t])$ which completes the proof. □

Another attribute function ω_M^{set} can be defined similar to ω_M by defining Ψ using *set* instead of *multiset*. To distinguish both versions we denote ω_M by ω_M^{mset} if Ψ is defined wrt. multisets.

Definition 4.3
Let $\Delta^{set}(t)$ be the set of all symbols of the term t. Then the symbol disagreement Ψ^{set} of two terms s and t is defined by

$$\Psi^{set}(s,t) = (\Delta^{set}(s) \backslash \Delta^{set}(t)) \ \cup \ (\Delta^{set}(t) \backslash \Delta^{set}(s)).$$

$\cup$ and $\backslash$ denote the set union and difference.

The definition of ω_M^{set} is similar to Definition 4.2. The proof of ω_M^{set} being an attribute function is analogue to the proof for ω_M on multisets.
Note that an equation which changes the ω_M^{set}-value of a term changes the ω_M-value of the term (wrt. a fixed set M) too.

4.2 Attribute functions based on relative positions

Both kinds of attribute functions above are based on the comparison of the (multi-)sets of the symbols of the terms s and t of the goal equation $\neg\ s = t$. Because of the property of being *attribute functions* we can use these criteria to determine a set of equations where one of these has to be applied at least once during the proof. But this only works if the symbol disagreement of the goal terms is non-empty. For instance, in case of the proof of

Example 4.2 ([Pel86], No.65)

$$\forall x : 0 + x = x \tag{20}$$
$$\forall x : x + 0 = x \tag{21}$$
$$\forall x : (-x) + x = 0 \tag{22}$$
$$\forall x, y, z : (x + y) + z = x + (y + z) \tag{23}$$
$$\forall x, y, z : x + x = 0 \tag{24}$$

negated theorem (c, b are skolem constants) :

$$\neg\ b + c = c + b \qquad \text{Goal equation} \tag{25}$$

could not be guided by the attribute functions above because $\Psi^{mset}(b+c, c+b)$ and $\Psi^{set}(b + c, c + b)$ are empty.
How can we characterize (using an attribute function) what should happen during the proof ? In the example above it is obvious that the relative position of b and c change. So we look at the leaves of the (term) tree and keep one of the skolem constants fixed, for instance c. Now we see that we have to apply an equation which removes b on the left-hand side of c.

In the following we will formalize the idea described above. First we define a function which collects all symbols of the term as a list (we have to save the relative positions of the symbols). Let $\bullet$ denote the concatenation of lists.

Definition 4.4 Symbol collecting function Π
Let t be a term, Then $\Pi(t)$ is defined as :

$$\Pi(t) = \begin{cases} [t], \text{ if } t \in \mathcal{V} \cup \Sigma \\ [f] \bullet \Pi(t_1) \bullet \ \ldots\ \bullet \Pi(t_n), \text{ if } t = f(t_1, \ldots, t_n) \end{cases}$$

Now we define the function which collects all symbols "to the left" of a given symbol. Because we are not interested in all symbols of the list $\Pi(t)$ (in fact we are only interested in *one* symbol to be removed, in the example above b wrt. the fixed symbol c) we define the function with respect to a given set of "interesting" symbols. The fixed symbol will be denotated by β ($\in \Sigma$), the set of "interesting" symbols by W ($\subset \Sigma$).
We define the *symbol selecting* function on lists of symbols (i.e. on $(\Sigma \cup \mathcal{V})^*$).

Definition 4.5 Symbol selecting function Φ $\Phi_{W,\beta}$ *is defined by :*

$$\Phi_{W,\beta}([]) = []$$

$$\Phi_{W,\beta}([t] \bullet T) = \begin{cases} [t] \bullet \Phi_{W,\beta}(T) & \text{if } t \neq \beta \text{ and } t \in W \cup \mathcal{V}, \\ [t] & \text{if } t = \beta, \\ \Phi_{W,\beta}(T) & \text{else.} \end{cases}$$

The ω-function based on the functions defined above is the combination of $\Phi_{W,\beta}$ and Π.

Definition 4.6 Attribute function $\omega_{W,\beta}$
Let β be a symbol ($\in \Sigma$) and $W \subset \Sigma$. The attribute function $\omega_{W,\beta}$ is defined by : $\omega_{W,\beta}(t) = \Phi_{W,\beta}(\Pi(t))$

For the proof of $\omega_{W,\beta}$ being an attribute function we need some technical lemmata.
Given a substitution σ on terms we need for technical reasons the corresponding equivalent substitution $\overline{\sigma}$ on lists, i.e. the substitution which replaces a variable x in a list by $\Pi(\sigma(x))$. Note that for every substitution σ and every term s it holds : $\overline{\sigma}(\Pi(s)) = \Pi(\sigma(s))$.

$$\begin{array}{ccc} \sigma : t & \longrightarrow & \sigma(t) \\ \Big\downarrow \Pi & & \Big\downarrow \Pi \\ \overline{\sigma} : \Pi(t) & \longrightarrow & \Pi(\sigma(t)) \end{array}$$

Lemma 4.3 *For all $L_1, L_2 \in (\Sigma \cup \mathcal{V})^*$ it holds:*

$$\Phi_{W,\beta}(L_1 \bullet L_2) = \begin{cases} \Phi_{W,\beta}(L_1) & \text{if } \beta \in L_1 \\ \Phi_{W,\beta}(L_1) \bullet \Phi_{W,\beta}(L_2) & \text{else} \end{cases}$$

Proof 4.4 We prove the assertion by induction on L_1.
For the empty list $L_1 = []$ it holds $\beta \notin L_1$ and $\Phi_{W,\beta}(L_1) = []$. With $[] \bullet L_2 = L_2$ the assertion follows.

So we have to prove the assertion for $L_1 = [a] \bullet L$ assuming the statement is valid for $L_1 = L$.
Case 1 : $a = \beta$: Then $\Phi_{W,\beta}([a] \bullet L \bullet L_2) = [a] = \Phi_{W,\beta}([a] \bullet L)$.
Case 2 : $a \neq \beta$ and $a \in W$ or $a \in \mathcal{V}$: $\Phi_{W,\beta}([a] \bullet L \bullet L_2) = [a] \bullet \Phi_{W,\beta}(L \bullet L_2)$.
So we could apply the induction hypothesis.
If $\beta \in L$ (which implies $\beta \in [a] \bullet L$) we get $[a] \bullet \Phi_{W,\beta}(L \bullet L_2) = [a] \bullet \Phi_{W,\beta}(L)$.
If $\beta \notin L$ (which implies $\beta \notin [a] \bullet L$) we achieve $[a] \bullet \Phi_{W,\beta}(L \bullet L_2) =$
$= [a] \bullet \Phi_{W,\beta}(L) \bullet \Phi_{W,\beta}(L_2) = \Phi_{W,\beta}([a] \bullet L) \bullet \Phi_{W,\beta}(L_2)$.
Case 3 : $a \neq \beta$, $a \notin W$ and $a \notin \mathcal{V}$: Because of $\Phi_{W,\beta}([a] \bullet L \bullet L_2) = \Phi_{W,\beta}(L \bullet L_2)$ the assertion trivially holds. □

The next lemma states that $\Phi_{W,\beta}$ is idempotent.

Lemma 4.5 *For all $L \in (\Sigma \cup \mathcal{V})^*$ it holds : $\Phi_{W,\beta}(\Phi_{W,\beta}(L)) = \Phi_{W,\beta}(L)$.*

Proof 4.6 The proof is done similar to the proof of Lemma 4.3 by induction on the length of the list. □

Lemma 4.7 *For all $L_1, L_2 \in (\Sigma \cup \mathcal{V})^*$: $\Phi_{W,\beta}(L_1 \bullet L_2) = \Phi_{W,\beta}(L_1 \bullet \Phi_{W,\beta}(L_2))$.*

Proof 4.8 Using Lemmata 4.3 and 4.5 the proof is trivial. □

Lemma 4.9 *For all $L \in (\Sigma \cup \mathcal{V})^*$ and all substitutions $\overline{\sigma}$:*
$\Phi_{W,\beta}(\overline{\sigma}(\Phi_{W,\beta}(L))) = \Phi_{W,\beta}(\overline{\sigma}(L))$.

Proof 4.10
We prove Lemma 4.9 by induction on the length of the list L.
For the empty list [] holds $\Phi_{W,\beta}(\overline{\sigma}(\Phi_{W,\beta}([]))) = \Phi_{W,\beta}(\overline{\sigma}([])) = []$.
Thus we have to prove the assertion for $[a] \bullet L$ assuming it is valid for L.
Case 1 : $a = \beta$: Then $\Phi_{W,\beta}([a] \bullet L) = [a]$. Since β is not a variable it holds $\sigma(a) = a$. Thus the assertion is valid.
Case 2 : $a \neq \beta$ and $a \in W$: Then $\Phi_{W,\beta}([a] \bullet L) = [a] \bullet \Phi_{W,\beta}(L)$. Thus we get
$\Phi_{W,\beta}(\overline{\sigma}(\Phi_{W,\beta}([a] \bullet L))) = \Phi_{W,\beta}(\overline{\sigma}([a] \bullet \Phi_{W,\beta}(L)))$
$= \Phi_{W,\beta}([a] \bullet \overline{\sigma}(\Phi_{W,\beta}(L))) = [a] \bullet \Phi_{W,\beta}(\overline{\sigma}(\Phi_{W,\beta}(L)))$
$= [a] \bullet \Phi_{W,\beta}(\overline{\sigma}(L))$ (by the induction hypothesis)
$= \Phi_{W,\beta}([a] \bullet \overline{\sigma}(L)) = \Phi_{W,\beta}(\overline{\sigma}([a] \bullet L))$.
Case 3 : $a \neq \beta$, $a \notin W$ and $a \notin \mathcal{V}$: Because of $\Phi_{W,\beta}([a] \bullet L) = \Phi_{W,\beta}(L)$ and $\sigma(a) = a$ the assertion trivially holds.
Case 4 : $a \neq \beta$, $a \notin W$, $a \in \mathcal{V}$:

$\Phi_{W,\beta}(\overline{\sigma}(\Phi_{W,\beta}([a] \bullet L))) = \Phi_{W,\beta}(\overline{\sigma}([a] \bullet \Phi_{W,\beta}(L))) = \Phi_{W,\beta}(\overline{\sigma}([a]) \bullet \overline{\sigma}(\Phi_{W,\beta}(L)))$
$= \Phi_{W,\beta}(\overline{\sigma}([a]) \bullet \Phi_{W,\beta}(\overline{\sigma}(\Phi_{W,\beta}(L))))$ (Lemma 4.7)
$= \Phi_{W,\beta}(\overline{\sigma}([a]) \bullet \Phi_{W,\beta}(\overline{\sigma}(L)))$ (induction hypothesis)
$= \Phi_{W,\beta}(\overline{\sigma}([a]) \bullet \overline{\sigma}(L))$ (Lemma 4.7)
$= \Phi_{W,\beta}(\overline{\sigma}([a] \bullet L))$ □

For the proof of the subterm-stability we need the property that replacing the subterm of a term t at position γ by s is compatible with Π in the following sense :

Lemma 4.11
For all $t, s \in \mathcal{T}(\Sigma, \mathcal{V})$, $L_1, L_2 \in (\Sigma \cup \mathcal{V})^*$ *and subterm positions* γ
$\exists L_1, L_2 \in (\Sigma \cup \mathcal{V})^* : \quad \Pi(t) = L_1 \bullet \Pi(t|\gamma) \bullet L_2$ *and* $\Pi(t[\gamma \leftarrow v]) = L_1 \bullet \Pi(v) \bullet L_2$

Proof 4.12 Proof by induction on the subterm position. □

Using the lemmata we prove Theorem 4.1.

Theorem 4.1
The function $\omega_{W,\beta}$ *is an attribute function,*
i.e. $\omega_{W,\beta}$ *is substitution-stable and subterm-stable.*

Proof 4.13
First we prove the **substitution-stability** of $\omega_{W,\beta}$.
Assuming $\omega_{W,\beta}(s) = \omega_{W,\beta}(t)$ we have
$\omega_{W,\beta}(\sigma(s)) = \Phi_{W,\beta}(\Pi(\sigma(s))) = \Phi_{W,\beta}(\overline{\sigma}(\Pi(s)))$ (because of $\Pi(\sigma(s))=\overline{\sigma}(\Pi(s))$)
$= \Phi_{W,\beta}(\overline{\sigma}(\Phi_{W,\beta}(\Pi(s))))$ (Lemma 4.9)
$= \Phi_{W,\beta}(\overline{\sigma}(\omega_{W,\beta}(s))) = \Phi_{W,\beta}(\overline{\sigma}(\omega_{W,\beta}(t)))$
$= \Phi_{W,\beta}(\overline{\sigma}(\Phi_{W,\beta}(\Pi(t)))) = \Phi_{W,\beta}(\overline{\sigma}(\Pi(t)))$ (Lemma 4.9)
$= \Phi_{W,\beta}(\Pi(\sigma(t))) = \omega_{W,\beta}(\sigma(t))$.

For the proof of the **subterm-stability** we assume $t|\gamma = u$ and $\omega_{W,\beta}(u) = \omega_{W,\beta}(v)$. Then there exist L_1 and L_2 (Lemma 4.11) such that
$\omega_{W,\beta}(t) = \Phi_{W,\beta}(\Pi(t)) = \Phi_{W,\beta}(L_1 \bullet \Pi(t|\gamma) \bullet L_2) = \Phi_{W,\beta}(L_1 \bullet \Pi(u) \bullet L_2)$
and
$\omega_{W,\beta}(t[\gamma \leftarrow v]) = \Phi_{W,\beta}(\Pi(t[\gamma \leftarrow v])) = \Phi_{W,\beta}(L_1 \bullet \Pi(v) \bullet L_2)$.
From $\omega_{W,\beta}(u) = \omega_{W,\beta}(v)$ it follows that $\beta \in \Pi(u)$ if and only if $\beta \in \Pi(v)$.
With Lemma 4.3 we achieve $\Phi_{W,\beta}(L_1 \bullet \Pi(u) \bullet L_2) = \Phi_{W,\beta}(L_1 \bullet \Pi(v) \bullet L_2)$. □

Let us return to the example No.65 of [Pel86] of proving

$$\neg\ b + c = c + b$$

We use the attribute function $\omega_{W,\beta}$ with $W = \{b\}$ and $\beta = c$. The goal equation is $\omega_{\{b\},c}$-changing ($\omega_{\{b\},c}(b+c) = [b, c]$, $\omega_{\{b\},c}(c+b) = [c]$). Searching for equations which might be able to equalize the $\omega_{W,\beta}$-value of both sides we obtain equations (22) and (24) as $\omega_{\{b\},c}$-changing equations. The first goal to remove b in front of c in the left-hand term of (25) fails since difference unification and rippling cannot enable the use of both equations (22) and (24). Hence, we try to generate b to the left of c in the right-hand term of the goal equation (25). Since $\omega_{\{b\},c}$-changing equations are (of course) the only equations which are able to generate b to the left of c we have to apply either (22) or (24) from right to left. Both equations (and one of those has to be applied during the proof !) are not applicable since 0 does not occur in the right-hand side of the goal equation. Using the same techniques as before we solve this problem with the help of the $\omega_{\{0\},c}$ attribute function. Equations (20), (21), (22) and (24) are $\omega_{\{0\},c}$-changing. Since we want to enable the use of (22) or (24) only (20) or (21) remains as possible equations. We use (20) to insert 0 in front of c (because it is directly applicable and inserts 0 to the left of c). Without further manipulation (20) is applicable to the goal equation and we obtain:

$$\neg\ b + c = (0 + c) + b \tag{26}$$

We return to the initial task of using equation (22) or (24) which both are now applicable. We choose equation (24) in favour of (22) because otherwise a new symbol $-$ would be introduced:

$$\neg\ b + c = ((x + x) + c) + b \tag{27}$$

The crucial point is now to find the right instantiation of x. The attribute function $\omega_{\{b\},c}$ suggests to instantiate x such that $\omega_{\{b\},c}(((x + x) + c) + b) = [b, c]$. Hence, x is replaced by the goal term $b + c$ which results in:

$$\neg\ b + c = (((b + c) + (b + c)) + c) + b \tag{28}$$

At this point the $\omega_{\{b\},c}$-values of both sides coincide and difference unification identifies the left-hand side of the goal equation inside the right-hand

side. Thus, the rest of the proof is done by rippling eliminating the shaded expressions in the coloured goal equation above:

$$\neg\ b + c = (((b + c) + (b + c)) + c) + b \tag{29}$$

which finally results in the trivial goal equation

$$\neg\ b + c = b + c \tag{30}$$

4.3 Strategies Using Attribute Functions

In the previous sections we introduced the notion of attribute functions and gave two examples of attribute functions to characterize "differences" of terms. The question arises how do we use the information obtained by an attribute function to guide a proof.

Looking for a bridge lemma The main application of an attribute function ω is to get an idea which equations have to be used during the proof. Given a goal equation $\neg\ s = t$ we look for an appropriate attribute function ω such that $\neg\ s = t$ is ω-changing. In order to refute $\neg\ s = t$ we have to deduce $\neg\ r = r$ which is ω-invariant. Thus, some application of an equation has to change the goal equation from an ω-changing into an ω-invariant equation. Due to section 3 this can only be done by the application of ω-changing equations. Hence, all ω-changing equations are potentially interesting equations. In order to reduce the search space we prefer those attribute functions with minimal number of ω-changing equations. Thus, the more attribute functions the prover knows the better the system will be:
Using ω_M^{mset} we can vary over M. Besides the given (class of) attribute functions ω_M^{mset} there are other attribute functions, for instance by defining ω_M^{mset} not as a multiset but as a set (ω_M^{set}).
The set of all ω_M^{mset}-changing equations behaves monotonic wrt. M, i.e. adding symbols to a set M results in at least the set of ω_M^{mset}-changing equations. Thus, using the attribute function ω_M^{mset} we consider ω_M^{mset} for all M which contain exactly one constant of $\Psi^{mset}(s,t)$. If there is an ω_M^{mset} with exactly one ω_M^{mset}-changing equation $q = r$ we choose this ω_M^{mset} and try to prove the goal equation by forcing the application of $q = r$.

The monotonicity wrt. M holds for ω_M^{set} too.
Note that any equation which changes the ω_M^{set}-value of a term is ω_M^{mset}-changing (wrt. a fixed set M) too. So we will look for ω_M^{set}-changing equations first.
The attribute functions $\omega_{W,\beta}$ are suitable in cases where each symbol of the left-hand side occurs also on the right-hand side and vice versa. These attribute functions consider the relative position of symbols with respect to some fixed symbol.
In general attribute functions determine only sets of ω-changing equations and the question arises which of these equations should be used as a bridge lemma. Choosing $M = \{b\}$ in the example of section 4.1 suggests both equations (16) and (18) as bridge lemmas. We use additional heuristics to reduce the number of possible bridge lemmata. E.g. a well-known (SOS) heuristic is to prefer equations which are part of the theorem if they are ω-changing. Another heuristic is to classify the equations by a *set of attribute functions* ω and to select that equation which is ω-changing wrt. the largest set of attribute functions ω. In the example of section 4.1 this forces the application of (18).
If some *measure* $(>)$ for the change of the ω-value of a term (resulting from the application of an equation) is available a third heuristic selects the "maximal" (wrt. to the measure) equation. An appropriate measure for $q = r$ (wrt. to the attribute function ω_M^{mset}) is the cardinality of the multiset difference of $\omega_M^{mset}(q)$ and $\omega_M^{mset}(r)$.
We may also use the measure $>$ to determine the direction in which the selected equation $q = r$ has to be applied. E.g. if $\omega(s) > \omega(t)$ and $\omega(q) > \omega(r)$ we have to apply $q = r$ from left to right on s to decrease the ω-value of s. In order to prove (19) by (18) the equation (18) has to be applied from left to right to the term $c + b$.

Enabling the use of a bridge lemma Let $\neg\ s = t$ be the goal equation. Once, we have selected an equation $q = r$ as a bridge lemma and suppose we have to apply the equation from left to right on the left-hand side s of the goal-equation we have to find an appropriate subterm of s which is unifiable with q. In general, we have to manipulate s with the help of given equations before we are able to find such a subterm such that the bridge lemma is applicable. In order to overcome this problem we use

difference unification between q and the given goal term s to determine both, an appropriate subterm of the goal equation and those parts of this subterm which prohibit the application of $q = r$. With the help of rippling techniques we try to eliminate these parts in order to enable the use of the bridge lemma. In doing so we are able to use all the sophisticated heuristics which have been developed in the context of induction proofs to enable the use of an induction hypothesis. Furthermore, if rippling gets stuck we can unblock the situation by recursively initiating the presented difference reduction technique.
Another heuristic suggests to use attribute functions to manipulate the goal equation by *ω-invariant equations*. Especially if we succeeded in removing an ω-difference between two terms we forbid the application of ω-changing equations. Suppose, given a goal equation with $\omega(s) \neq \omega(t)$ we have found a proof of $s = s'$ where $\omega(s') = \omega(t)$ holds. Then, we forbid the use of ω-changing equations while proving $s' = t$.

5 A Proof of SAM's Lemma

We will illustrate our approach by the well-known example of SAM's lemma [GOBS69] which is still a challenge equality problem [McC88] although it has already been solved by some resolution provers (e.g. [BBB$^+$84]) without paramodulation some years ago.
Given two associative, commutative, and idempotent functions f and g and additionally, the following set of axioms:

$$\forall x, y : f(x, g(x, y)) = x \quad (31) \qquad \forall x, y : g(x, f(x, y)) = x \quad (34)$$
$$\forall x : f(0, x) = 0 \quad (32) \qquad \forall x : f(1, x) = x \quad (35)$$
$$\forall x : g(0, x) = x \quad (33) \qquad \forall x : g(1, x) = 1 \quad (36)$$

$$\forall x, y, z : f(x, z) = x \rightarrow f(g(x, y), z) = g(x, f(y, z)) \quad (37)$$

the task is to refute the following formula:

$$f(b, f(c, d)) = 0 \quad \text{Premise 1} \quad (38)$$
$$f(a, g(c, d)) = 0 \quad \text{Premise 2} \quad (39)$$
$$\neg f(g(a, f(b, c)), g(a, f(b, d))) = a \quad \text{Goal equation} \quad (40)$$

where a, b, c, d are skolem constants.
As mentioned before the first step in proof planning is to look for bridge lemmata which have to be used. Comparing the occurring symbols of both sides of the conclusion we note that we have to manipulate the left-hand side in order to delete the occurrences of e.g. b, c, and d since they do not occur on the right-hand side. Using for instance the *attribute function* $\omega^{set}_{\{b,c,d\}}$ we obtain (31), (34), (32), (36), and both premises as possible bridge lemmata. Heuristics suggest to use in the first place equations of the theorem which results in the premises as interesting equations.
The next step is to enable the use of one of the premises in the conclusion. Since the symbols b, c, and d have to be removed the premises have to be applied from left to right. Neither of both premises is applicable without manipulating the conclusion. In order to guide this transformation we determine the syntactical differences between the left-hand sides of the premises and the left-hand side of the conclusion with the help of difference unification.
Comparing the left-hand side of (39) with the goal equation gives no hint how to apply it while the left-hand side of (38) difference-unifies (under ACI) against the left-hand side of (40). Hence, we try to enable its use with the help of rippling. Shading the syntactical differences between both terms we obtain the following coloured version of the goal equation:

$$\neg f(g(a, f(b, c)), g(a, f(b, d))) = a \tag{41}$$

In a next step we have to move the shaded expressions of the conclusion outside without changing the white expressions. Once, the white expressions will come together the manipulated conclusion has the form $...f(f(b, c), f(b, d))...$ and the second premise is applicable. Hence, we look for a context-moving equation which is able to move the shaded expressions outside. We obtain the following conditional C-equation of (37):

$$\forall x, y, z : f(x, z) = x \rightarrow f(g(x, y), z) = g(x, f(y, z))$$

Using this equation we are able to manipulate (41) to

$$\neg g(a, f(f(b, c), g(a, f(b, d)))) = a \tag{42}$$

since the instantiated condition of (37) $f(a, g(a, f(b, d))) = a$ holds trivially due to axiom (31). It remains the problem to move the second shaded area

outside. Again, the left-hand side of the conditional equation matches the appropriate subterm under the given theory but in this case the prover fails to establish the instantiated condition $f(a, f(b, c)) = a$ which now prevents the use of the equation. Also, no context-deleting C-equation is applicable and the rippling process gets stuck.
In order to get the process back on its feet we have to manipulate the obstructing shaded expression of the conclusion to enable the use of appropriate C-equations. Selecting, for instance, the context-deleting C-equation

$$g(0, x) = x$$

in order to delete the shaded expression of (42), we have to prove

$$g(a, f(f(b, c), g(0, f(b, d)))) = g(a, f(f(b, c), g(a, f(b, d)))).$$

Again we use attribute functions to guide this subproof. Comparing the occurring symbols of both sides we have to eliminate 0 in favour of a, i.e. we use $\omega^{set}_{\{0\}}$. Using the heuristic to prefer the premises we use $f(a, g(c, d)) = 0$ in order to delete the symbol 0 but also in order to involve the symbol a. Thus, we have to prove that

$$g(a, f(f(b, c), g(f(a, g(c, d)), f(b, d)))) = g(a, f(f(b, c), g(a, f(b, d))))$$

which is solved again using rippling: We have to delete the shaded areas in

$$g(a, f(f(b, c), g(f(a, g(c, d)), f(b, d))))$$

in order to obtain $g(a, f(f(b, c), g(a, f(b, d))))$. This is done with standard rippling-techniques: First, the shaded area is moved up by (37) obtaining

$$g(a, f(f(b, c), f(g(c, d), g(a, f(b, d)))))$$

and secondly, the shaded expression is deleted by the context-deleting C-equation of (31) which yields the wished term

$$g(a, f(f(b, c), g(a, f(b, d)))).$$

Hence, we have finished our subproof and return to the main problem of enabling the use of the second premise in (42).

Using the result of our lemma we obtain

$$\neg g(a, f(f(b,c), f(b,d))) = a$$

from (42) which already enables the use of the second premise. Applying it we obtain the trivial task

$$\neg g(a, 0) = a$$

which can be again solved either directly with the help of difference unification or by computing interesting equations with the help of $\omega_{\{g\}}$. Both result in the application of (33) which finishes the proof.
Summing up the previous example, combining rippling techniques and heuristics suggesting equations which are (likely) to be used during a proof states a powerful approach to control equational reasoning.

6 Conclusion

We presented a methodology to guide equational reasoning in a goal directed way. A bundle of attribute functions enable the prover to select appropriate bridge lemmas. To enable their use the syntactical differences which prevents them from application are determined and minimized by techniques of colouring terms and rippling. In cases the rippling process gets stuck attribute functions are also used to unblock the process. Due to the goal-directed approach additional heuristics to guide equational reasoning can easily be integrated into this method.
Until now this is only a first step towards a "theory" of equational reasoning from a somewhat different point of view. The presented approach suggests two tasks to do :

- A major goal is the definition of further attribute functions. The more attribute functions the prover "knows" the better it can determine the necessary bridge lemmas.
- A second task is to develop further techniques to enable the application of the selected lemmata.

References

[AA90] S. Anantharaman and N. Andrianarivelo. Heuristical criteria in refutational theorem proving. In A. Miola, editor, *Design and Implementation of Symbolic Computation Systems: Proc. of the International Symposium DISCO'90*, pages 184–193. Springer, Berlin, Heidelberg, 1990.

[BBB+84] Susanne Biundo, Karl Hans Bläsius, Hans-Jürgen Bürckert, Norbert Eisinger, Alexander Herold, Thomas Käufl, Christoph Lingenfelder, Hans Jürgen Ohlbach, Manfred Schmidt-Schauß, Jörg H. Siekmann, and Christoph Walther. The Markgraf Karl Refutation Procedure. SEKI-MEMO MK-84-01, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, Institut für Informatik I, Universität Karlsruhe, Postfach 6380, 7500 Karlsruhe 1, Germany, 1984.

[BDP87] Leo Bachmair, Nachum Dershowitz, and David Plaisted. Completion without failure. In H. Ait-Kaci and M. Nivat, editors, *Conference on Resolution of Equations in Algebraic Structures (CREAS)*, Lakaway, Texas, USA, 1987.

[BG90] Leo Bachmair and Harald Ganzinger. On restrictions of ordered paramodulation with simplification. In Mark E. Stickel, editor, *Proceedings 10^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Artificial Intelligence (LNAI) 449*, pages 427–441, Kaiserslautern, Germany, July 1990. Springer-Verlag, Berlin, Germany.

[Blä86] Karl Hans Bläsius. *Equality Reasoning Based on Graphs*. PhD thesis, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, 1986. Also published as SEKI-Report SR-87-01, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany.

[Bun88] Alan Bundy. The use of explicit plans to guide inductive proofs. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings 9^{th} International Conference on Automated Deduction (CADE),*

Lecture Notes in Computer Science (LNCS) 310, pages 111–120, Argonne, Illinois, USA, 1988. Springer-Verlag, Berlin, Germany.

[BW92] David Basin and Toby Walsh. Difference matching. In Deepak Kapur, editor, *Proceedings 11^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Artificial Intelligence (LNAI) 607*, pages 295–309, Saratoga Springs, New York, USA, June 1992. Springer-Verlag, Berlin, Germany.

[Den91] Jörg Denzinger. Distributed knowledge-based deduction using the team work method. SEKI-Report SR-91-12, Fachbereich Informatik, Universität Kaiserslautern, Postfach 3049, 6750 Kaiserslautern, Germany, 1991.

[Dig79] Vincent J. Digricoli. Resolution by unification and equality. In William H. Joyner, editor, *Proceedings 4^{th} Workshop on Automated Deduction*, pages 43–52, Austin, USA, February 1979.

[GOBS69] J. Guard, F. Oglesby, J. Bennett, and L. Settle. Semi-automated mathematics. *Journal of the Association for Computing Machinery (ACM), ACM, Inc., 1133 Avenue of the Americas, New York 10036, USA*, 16:49–62, 1969.

[Hut90] Dieter Hutter. Guiding induction proofs. In Mark E. Stickel, editor, *Proceedings 10^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Artificial Intelligence (LNAI) 449*, pages 147–161, Kaiserslautern, Germany, July 1990. Springer-Verlag, Berlin, Germany.

[KB70] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[McC88] William McCune. Challenge equality problems in lattice theory. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings 9^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Computer Science (LNCS) 310*, pages 704–709, Argonne, Illinois, USA, 1988. Springer-Verlag, Berlin, Germany.

[NSS59] Allen Newell, J. C. Shaw, and Herbert A. Simon. Report on a general problem-solving program. In *Proceedings of the International Conference on Information Processing*, pages 256–264. UNESCO, Paris, June 1959.

[Pel86] Francis Jeffry Pelletier. Seventy-five problems for testing automatic theorem provers. *Journal of Automated Reasoning (JAR), Kluwer Academic Publishers, 3300 AH Dordrecht, The Netherlands*, 2:191–216, 1986.

[SA92] Rolf Socher-Ambrosius. A goal oriented strategy based on completion. Report MPI-I-92-206, Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany, February 1992.

[ZK88] Hantao Zhang and Deepak Kapur. First order theorem proving using conditional rewrite rules. In Ewing L. Lusk and Ross A. Overbeek, editors, *Proceedings 9^{th} International Conference on Automated Deduction (CADE), Lecture Notes in Computer Science (LNCS) 310*, pages 1–20, Argonne, Illinois, USA, 1988. Springer-Verlag, Berlin, Germany.