Equalizing Terms by Difference Reduction Techniques *

Dieter Hutter †

Abstract

In the field of inductive theorem proving syntactical differences between the induction hypothesis and induction conclusion are used in order to guide the proof [BvHS91, Hut90, Hut]. This method of guiding induction proofs is called rippling / coloring terms and there is considerable evidence of its success on practical examples. For equality reasoning we use these annotated terms to represent syntactical differences of formulas. Based on these annotations and on hierarchies of function symbols we define different abstractions of formulas which are used for a hierarchical planning of proofs. Also rippling techniques are used to refine single planning steps, e.g. the application of a bridge lemma, on a next planning level.

1 Introduction

In the field of inductive theorem proving syntactical differences between the induction hypothesis and induction conclusion are used in order to guide the proof [BvHS91, Hut90, Hut]. This method of guiding induction proofs is called rippling / coloring terms and a large amount of papers have been published which report its success on practical examples.

Rippling is based on the notion of annotated terms which are terms with additional information at each occurrence of a symbol. We represent this information in coloring each occurrence of a symbol inside a term. Different colors represent different treatments of the corresponding bits of the term during the deduction process. Performing manipulations (rippling) on annotated terms white parts of a term stay invariant while the grey parts are subjects to change. In order to manipulate these annotated terms correctly also equations are annotated such that the white parts on both sides coincides while the grey parts represent the differences between both. An example of such an annotated equation is given in (1).

$$s(X) \times Y = Y + (X \times Y) \tag{1}$$

Using (1) we are able to perform the following rewriting on $s(s(a)) \times b$:

$$s(s(a)) \times b \;\rightarrow\; b + (s(a) \times b) \;\rightarrow\; b + (b + a \times b)$$

Notice that in all these annotated terms the white parts — namely $a \times b$ — coincide. Using annotating terms rippling preserves whose parts of the term which are colored

*This research was supported by an EPSRC-Visiting Fellowship under grant GR/K84981

†German Research Center for Artificial Intelligence (DFKI GmbH), Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany; Email: hutter@dfki.uni-sb.de. Visitor of the Department of Artificial Intelligence, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN; Email: dieterh@dai.ed.ac.uk

white. We call these parts the *skeleton* of an annotated term while the grey parts are called the *wave-front*. Precisely, the skeleton of such an annotated term t is a set of un-annotated terms which are obtained by successively replacing each subterm t' of t with a grey top-level symbol by some biggest subterm of t' with a white top-level symbol. For instance the skeleton of $p(x) \times y$ is $\{x \times y\}$ or the skeleton of $s(x + y)$ is $\{s(x), s(y)\}$. Thus, in case a colored term t contains no subterm with a white top-level symbol the skeleton is empty. For instance the skeleton of $p(x) \times y$ is the empty set $\emptyset$.

Using annotated terms for equality reasoning, the skeleton represent common parts of both sides of the equality to be proven. Thus, these common parts will be preserved during the rippling (or manipulation) process. Starting with two completely grey colored terms, we have to manipulate both terms and increase the skeleton, step by step, until no further grey parts occur on either side of the equation. Thus, the size of this common skeleton (compared to the size of the wave-front) constitutes a measure how far the proof has been pushed further.

Given two annotated formulas which have to be identified, the goal of difference reduction is to equate the corresponding wave-fronts in both formulas. Once, we succeed in identifying common parts within corresponding wave-fronts (i.e. wave-fronts at the same positions wrt. the skeleton) we add them to the common skeleton and continue the modification until no wave-front remains in either term.

Thus, the approach is an iteration over two steps: Firstly, both terms are modified preserving the common skeleton, and secondly, the skeleton is enlarged by common parts of both terms not yet belonging to the skeleton. If we suppose that the modification of the equation does not steadily increase the size of the wave-fronts then the iterated enlargement of the skeleton will result in step by step reduction of differences. The possible positions where rules can be applied are steadily reduced until no wave-fronts occur anymore in both terms and thus, both terms are syntactically equal. Figure 1 gives an outline of the sketched algorithm.

```
while not q = r do
 { rewrite q and r skeleton-preserving
   select two subterms q’, r’ with
       {top(q’) = top(r’) = f, f is a wavefront}
        add top(q’) and top(r’) to the skeleton}
```

Figure 1: Outline of the DR-procedure

As an example suppose we are interested in proving the following property about the length of concatenated lists:

$$\forall x, y{:}list\ length(app(x, y)) = length(app(y, x)) \tag{2}$$

We assume that $+$ denotes the addition on natural numbers and that the following properties about $+$, *length*, and *app* are given.

$$\begin{aligned} X + Y &= Y + X && (3) \\ length(app(X, Y)) &= length(X) + length(Y) && (4) \end{aligned}$$

In order to prove the equation

$$length(app(x, y)) = length(app(y, x)) \tag{5}$$

we may enlarge the empty skeleton on both sides without any modifications to $\{x, y\}$ which results in:

$$length(\,app(x\,, y\,)) = length(\,app(y\,, x\,)) \tag{6}$$

In a next step we try to contract the white parts by rippling out techniques which result in the following equation:

$$\boxed{length(x) \;+\; length(y)} \;=\; \boxed{length(y) \;+\; length(x)} \tag{7}$$

Comparing both sides we would like to increase the skeleton by the occurrences of + on both sides but the first argument of + on the left hand side corresponds to the second argument of + on the right hand side and vice versa. Hence, in order to obtain a common skeleton on both sides we have to permute the arguments of + on one side of the equation which yields

$$\boxed{length(x) \;+\; length(y)} \;=\; \boxed{length(x) \;+\; length(y)} \tag{8}$$

Now, adding + to the skeleton we are finished obtaining the trivial equality:

$$length(x) + length(y) = length(y) + length(x) \tag{9}$$

In this paper we will elaborate this idea of using rippling techniques for equality reasoning. In section 2 we give an overview how we use the calculus based on annotated terms to represent and maintain necessary information during the proof. In section 3 we discuss the implications of enlarging the skeleton to the proof process and in 4 we present a general procedure for our methodology. In section 5 we illustrate different abstraction techniques to manipulate the wave-fronts in order to ease the enlargement of the skeleton. Finally, we give several examples in the appendix A.

2 Rippling as a Calculus for Difference Reduction

The notion of annotated terms enables an explicit representation of differences between terms. Annotations are attached to the occurrences of each symbol indicating whether this occurrence belongs to the common part of both formulas or whether it is part of their differences.

Thus in contrast to describing differences by identifying minimal subterms which contain them (as in the RUE-approach), the notion of annotated terms allows one to represent the differences explicitly and therefore, seems to be a more adequate representation for implementing difference reduction techniques. Also, the explicit representation of differences is an important requirement in computing a measure of how far two terms or formulas differ which is a central aid in guiding (parts of) the proof search.

Once annotation has been computed describing the differences between two terms, both annotated terms share the same skeleton. Thus, the differences (or wave-fronts) can be localized relative to that skeleton. In our previous example of proving (2), all annotations are located within the argument of *length*. In order to prove the theorem we moved the annotations into the outermost positions on both sides of the equality. Thus the positions of the annotations reflect the progress of the proof in our example.

In general, measures computing the size of the differences may take into account several properties: the size of the skeleton, the positions of the wave-fronts relative to the skeleton, the size of the wave-fronts, the ordering on the symbols occurring in the wave-fronts, etc. In our example of proving (2), after applying the equation 4 the residual wave-fronts of both sides are located at outermost positions but the proof is still not finished. Thus, measures guiding the proof may be refinements of a lexicographical ordering on the size of the common skeleton and the positions of the wave-fronts.

According to our approach, two annotated terms have to be modified in order to enlarge their common skeleton. Hence, we are only interested in deduction steps which do not weaken the common skeleton.

The constraint to preserve the given skeleton reduces the search space of the deduction. Deduction steps have to respect the common skeleton and only the wave-fronts can be modified. Applying an annotated equation E to a subterm t requires that first, the left hand side of E matches (resp. unifies) the subterm t and second, that the skeleton of the left hand side of E matches the skeleton of t. As a consequence, the wave-fronts of the left hand side of E match the wave-fronts of t. From an operational point of view, the application of the equation E will remove the wave-fronts and create instances of the wave-fronts on the right hand side of E in t. The positions of the wave-fronts in the modified t will correspond to the positions of the wave-fronts on the right hand side of E.

Annotating terms is also used to focus the attention of the deduction process to specific subterms of a formula. Since the skeleton remains invariant, admissible deduction steps will only change the wave-front parts of the annotated term. Thus, declaring parts of a term to be part of the skeleton prevents them from being modified. In contrast to other approaches (e.g. RUE) this notion is not restricted to wave-fronts which are proper subterms or superterms of the formula. We are able to treat arbitrary wave-fronts regardless whether they are intertwined with skeletons or not. For instance in the example above the equation was applied to a subterm $\mathit{length}(\mathit{app}(x, y))$ which contains both, wave-front and skeleton.

3 Skeleton

In this section we address the issues arising from the expansion of a given skeleton by some common parts of the wave-fronts of both terms. Suppose an equality proof of two annotated terms q, and r. Hence, we start with the equality $q = r$ and modify left- and right hand side obtaining a sequence $q = r \to q^1 = r^1 \to \ldots \to q^n = r^n$ of equations. For each of these equations the skeleton of q_i is identical to r_i. Since we only enlarge or at least preserve the skeleton, the skeleton of q_i (r_i respectively) is embedded into the skeleton of each q_{i+k} (r_{i+k} respectively). Especially the skeleton of $q^n = r^n$ contains the initial skeleton of $q = r$. In other words, the skeleton of $q = r$ is a pattern of the goal equation $q^n = r^n$. Thus, each time we enlarge the skeleton we refine the pattern describing the goal term.

Roughly speaking, the skeleton of a term represents a set of occurrences of symbols in this term together with the subterm-relations in which the symbols occur. Once, we enlarge the common skeleton by an occurrence of a symbol new subterm-relations between the added symbol and already existing parts of the skeleton are also added. In order to achieve a common skeleton for both terms of the equality problem, these new subterm-relations have to coincide in both terms. In general there are two different kinds of clashes.

A vertical clash occurs if we add a symbol f to the skeleton and there is another symbol h within the skeleton such that in one term f is inside an argument of h and in the other term f is outside of any argument of f.

Suppose the terms $s(a+b)$ and $a+s(b)$. Adding s to the skeleton on both sides imposes a vertical clash between s and $+$. In order to obtain a common skeleton (containing s and $+$) on both sides we have either to move the occurrence of s in front of $+$ on the right-hand side or we have to move the occurrence of $+$ in front of s on the left-hand side.

A horizontal clash occurs if the added symbol f occurs within different arguments of a function h in both terms.

As an example of a horizontal clash consider the example of proving (2). We obtain two annotated terms $\mathit{length}(x) + \mathit{length}(y)$ and $\mathit{length}(y) + \mathit{length}(x)$ which agree on their skeleton $\{\mathit{length}(x), \mathit{length}(y)\}$ but enlarging the skeleton by the occurrences of $+$ would result in two different skeletons, namely $\{\mathit{length}(x) + \mathit{length}(y)\}$ and $\{\mathit{length}(y) + \mathit{length}(x)\}$. In order to equate both skeletons we have to permute the arguments of $+$ on one hand side.

Both types of clashes impose subgoals to the deduction process in order to obtain a common skeleton for both terms when enlarging the skeleton by the occurrences of a common symbol. On the other hand we may use the knowledge of possible clashes in order to guide the process of enlarging the skeleton such that we only allow those extensions of the skeleton which do not introduce horizontal or vertical clashes. The practical solution to that question will be discussed in section 4.

3.1 Divide and Conquer

Enlarging the skeleton by a common function symbol imposes also a link between the two selected occurrences of the symbol. During the ongoing deduction this link will be inherited and in case the proof succeeds it will finally link two identical subterms. Thus, this link (encoded in the enlargement of the skeleton) also divides the given equality problem into several "smaller" problems. We have to equate the corresponding arguments of both occurrences of the new skeleton symbol f in order to compute the arguments of this particular occurrence of f in the goal term q^n. Thus, we may use our procedure recursively to obtain identical terms in the corresponding arguments of f. In contrast to the original problem of identifying two terms we have now to modify two terms $f(q_1, \ldots, q_n)$ and $f(r_1, \ldots, r_n)$ until both share a common subterm $f(t_1, \ldots, t_n)$ which denotes the particular occurrence of f in the goal term q^n.

In general there are two possible alternatives to minimize the syntactical differences between corresponding argument positions of f.

vertical movement will reduce differences between both linked terms by rippling the differences in front of f or by rippling wave-fronts which are in front of f into the respective arguments of f. For instance suppose we want to equate $\mathit{double}(a) + b$ and $(a+a)+b$. After applying the associativity law on the right term and an annotated equation $\mathit{double}(X) + Y = X + (X+Y)$ on the left term we obtain $a + (a+b)$ on both sides.

horizontal movement will reduce differences within the arguments of the common symbols and try to enlarge the skeleton inside the terms. For instance suppose we want to equate $\mathit{double}(a) + b$ and $(a+a)+b$. Using the definition of *double* we obtain on both sides the term $(a+a)+b$. Enlarging the skeleton by the occurrence of $+$ will result in a common term $(a+a)+b$

In case there are already parts of the skeleton in $f(q_1, \ldots, q_n)$ and $f(r_1, \ldots, r_n)$ the possible movements are restricted since corresponding skeleton parts have to appear in corresponding arguments of f. In our example of proving (2) the way of horizontal movement in (7) was predicted by the skeleton parts $length(x)$ and $length(y)$ occurring in different arguments of $+$.

4 A More Elaborate Version

In this section we will discuss the procedural aspects of enlarging the skeleton and present an outline of a refined procedure for equalizing terms since the procedure sketched in 1 is rather vague.

The choice of appropriate skeletons is crucial to the success of the proof attempt. Thus, we would like to validate a given selection as soon as possible in order to avoid rather expensive backtracking. We consider a given correspondence between two occurrences $f(q_1, \ldots, q_n)$ and $f(r_1, \ldots, r_n)$ of f as useful if we succeed to manipulate both terms until they share a common subterm $f(t_1, \ldots, t_n)$. Thus, each time we draw a link between two occurrences of a function symbol f we have to tackle the task of manipulating both terms until they share the intended common subterm. Once we succeed, the common subterm $f(t_1, \ldots, t_n)$ represents the refined skeleton of both terms while the wave-front around the occurrences of this subterm are wave-fronts. Thus, any symbol of the wave-front will be outside of the skeleton which means that selecting a new link between occurrences of a symbol in the wave-front will not cause any vertical clashes.

We are left with the problem of adapting the arguments of f in both terms, i.e. we have to compute appropriate t_i for each argument i of f. In case skeleton parts occur already in some arguments of f we may have to use appropriate permutative equations in order to reorder the arguments in such a way that corresponding skeleton parts occur in corresponding argument positions of f. In case there are argument positions in which no skeleton part occurs we have to call our procedure recursively on the corresponding argument positions of f in order to obtain a common subterm for both subterms. At this point we have the choice whether we search only in corresponding argument positions for common subterms (thus avoiding horizontal clashes) or we search also between different argument positions and try to handle arising horizontal clashes afterwards. It turned out to be useful to examine the database in advance whether there are some permutative equations for f at all. In case there are no such equations we will not be able to handle any horizontal clashes and thus, common subterms are only computed between subterms in the same argument position of f. In case there are permutative equations we may search for common subterms in any argument positions and try to eliminate the horizontal clashes with the help of these equations.

Once we have established common subterms t'_i for each argument position i of f we have to get rid of possible wave-fronts still occurring in front of the common subterms. If we suppose that t'_i is the maximal common subterm[1] which we can establish between two corresponding arguments of f, then we have to ripple the wave-fronts either into other arguments of f and then try to enlarge the common subterm at that position (horizontal movement) or out of f (vertical movement) in order to get rid of the wave-fronts inside of f. Possible target positions for horizontal movements of wave wave-fronts in an argument q_i are restricted to those arguments q_j where the corresponding subterm r_j already contains a wave-front.

Horizontal movement will treat the differences (represented by the wave-fronts) locally at the denoted target argument position while the vertical movement will delay

[1] for example, one of the subterms contains no wave-fronts, i.e. one term is a subterm of the other

the problem of adapting the wave-fronts until we tackle the problem of equalizing the terms in which f occurs. Thus, vertical movement can also delay a possible clash which is then at least detected when equalizing the top-level terms.

To sum up we recape the general idea:

Suppose, $f(q_1, \ldots, q_n)$ and $f(r_1, \ldots, r_n)$ are the corresponding occurrences of f. In order to obtain a common skeleton for both they have to be manipulated until they share a common subterm $f(t_1, \ldots, t_n)$, i.e. we obtain annotated terms of the form $\ldots f(t_1, \ldots, t_n) \ldots$. In order to achieve this goal we first manipulate both terms to satisfy the pattern $f(\ldots t_1 \ldots, \ldots, \ldots t_n \ldots)$. Once we have reached this subgoal, the original goal can be obtained by using rippling techniques. We are left with the same problem of creating a common skeleton for the corresponding arguments (i.e. $q_1, \ldots, q_n$ and $r_1, \ldots, r_n$) of the related occurrences of f. Hence, we iterate the procedure and equalize the corresponding arguments of f and enlarge their skeletons in order to obtain a term like $f(\ldots t_1 \ldots, \ldots, \ldots t_n \ldots)$.

Thus the general outline of the difference reduction procedure looks as given in Figure 2. It is presented in a state-based manner such that the procedure operates "destructively" on `q` and `t`. Initially, both terms given to the procedure have no skeleton at all. Thus, `q` and `t` are both initially wave-fronts (without any wave-hole).

```
equate_with_wave_front (q, r) =

 while q =/ r and skeleton-expansion was possible
   { rewrite_1 q and r
     select two subterms q', r' with
        {top(q') = top(r') = f, f is a wavefront,
         and skel(q') = skel(r')}
      such that
       { add top(q') and top(r') to the skeleton
         let q' = ... f(q_1,..., q_n)...
             and r' = ... f(r_1,..., r_n)...
           {rewrite_2 q' and r'
             for all 1 <= i <= n: equate_with_wave_front(q_i, r_i)}
         remove wave-fronts in the arguments of f}
        backtrack to different subterms q' and r' if this fails}
```

Figure 2: Refinement of the DR-procedure

The general aim of the procedure is to enlarge the common skeleton of `q` and `r` step by step which is done with the help of adding common symbols `f` of the wave-fronts of `q` and `r` to the skeleton.

The purpose of `rewrite_2` is to prepare the divide and conquer step and to avoid horizontal clashes. In case of permutative functions it has to move corresponding terms of `q` and `r` into corresponding arguments of `f`.

In some cases the enlargement of the skeleton by `f` results in a horizontal clash and then there are two different skeletons for `q` and `r` since a common part of the old skeleton now occur in different arguments of f. In this case, the application of some permutative laws is necessary since the skeletons of `q` and `r` shall also coincide after the enlargement of the skeleton. Special strategies can be inbuilt in case of commutative or/and associative functions.

In case there are permutative laws for f available, heuristics are needed to judge the "best" correspondence between the `f`-arguments of `q'` and `r'`. They may take into account the set of occurring variables within an argument and how nested they occur within the argument. Consider for instance the equality problem

$$(x \times x \times x) + (x \times x) = (x \times x) + (x \times x \times x).$$

In order to find a proof of this theorem it is essential to relate the first argument of + on the left-hand side to the second argument on the right-hand side and vice versa.

Once common subterms have been identified for each argument position of f the task arises to get rid of all wave-fronts inside f in order to glue all the skeleton parts together to a single term. Vertical movements will ripple out the differences in front of f while horizontal movements will try to concentrate the occurrences of wave-fronts to specific argument positions.

In order to enlarge the skeleton first, we have to localize the wave-fronts in both terms at same positions (with respect to the common skeleton) and to identify the corresponding wave-fronts. Once parts of corresponding wave-fronts have been added to the skeleton these parts are split into wave-fronts and skeleton giving rise to a next iteration of adapting the position of the wave-fronts wrt. the skeleton.

Thus, a major goal of this part of the process is the movement of wave-fronts into appropriate argument positions. If we abstract from the shape of wave-fronts, the application of annotated equation will simply move wave-fronts with respect to the skeleton. In this sense the consequences of applying an equation (or in general a rule) is predictable before the result of application is calculated. Rules can be classified according to the direction in which they will move wave-fronts. Ignoring the shape of the wave-fronts results in an abstraction of (annotated) terms which allows one to plan parts of the deduction process. For instance we may specify in which directions wave-fronts have to be moved during the application as it is done within induction proofs. A planner may use such a knowledge in order to compute a way through the skeleton that the wave-front must be moved along. For instance applying the annotated equation (4) from left to right will always ripple the wave-front of the first argument of + in front of +.

5 Adapting Wave-Fronts by Abstraction

Up to now the different roles of rewriting and skeleton-extension are not yet specified. So, one possible (but rather useless) instance of this procedure would involve a "complete" equality-reasoner as the sub-procedure `rewrite_1` which performs the whole equality proof at once and then, adds both terms completely to the skeleton. In this case the equality procedure does not benefit from any search restriction which a common skeleton would impose on possible deductions. On the other hand the completeness of the approach is trivial unless the underlying equality prover used in `rewrite_1` is complete.

Our approach uses abstractions in order to guide the adaption of wave-fronts. Following [GW92a, GW92b] an abstraction is a pair of formal systems $\langle \Sigma_1, \Sigma_2 \rangle$ with languages Λ_1 and Λ_2 and a total function $f_\Lambda : \Lambda_1 \to \Lambda_2$. In our setting Λ_1 is the underlying annotated first-order calculus we use for equality reasoning while Λ_2 depends on the abstraction we choice to tackle the actual problem (see below).

In order to use abstractions we have to set up additional restrictions to the used abstraction. Since the manipulation of the terms q and r has to preserve the common skeleton, also the abstraction function f_Λ and the formal system Λ_2 have to maintain this information of invariant term-parts properly. The given problem of

$$\begin{array}{ccccccc} q = r & \rightarrow^*_{\Lambda_1} & q_1 = r_1 & \ldots & \rightarrow^*_{\Lambda_1} & q_n = r_n \\ \downarrow_{f_\Lambda} & & \uparrow & \ldots & & \uparrow \\ \Phi(a,b) & \rightarrow_{\Lambda_2} & \Phi(a_1,b_1) & \ldots & \rightarrow_{\Lambda_2} & \Phi(a_n,b_n) \end{array}$$

Figure 3: Abstracted proof search

equalizing two annotated terms q and r can then be transformed into a problem of manipulating two objects a and b in a formal system Λ_2 in order to obtain a so-called *solved form* $\Phi(a_n, b_n)$. Each step in the abstract space has to correspond to a sequel of steps in the ground space (Λ_1).

Thus once we find a solution in the abstract space, we have to transfer this solution to the given (first-order) calculus which imposes a set of different subtasks to be tackled. For each intermediate result $\Phi(a_{i+1}, b_{i+1})$ of the deduction in the abstract space we have to find an appropriate equation $q_{i+1} = r_{i+1}$ such that $q_i = r_i \rightarrow^*_{\Lambda_1} q_{i+1} = r_{i+1}$ holds.

Additionally we want to preserve this solution from backtracking in a later step when equalizing the obtained first-order terms. The best solution to that problem would be if we are able to encode the solved form $\Phi(a_n, b_n)$ (in the abstract space) in terms of the enlargement of the skeleton in the ground space. I.e. the enlargement of the skeleton would ensure that all following manipulations of the terms will not backtrack the adaption of both terms achieved by the use of this abstraction.

For instance in case we choose $\Lambda_1 = \Lambda_2$ and $f_\Lambda = id$ any solution in the abstract space will also solve the equality problem in the ground space. After a trivial mapping of the deduction-steps in Λ_2 to those of Λ_1, q_n and r_n are equal and the skeleton of both terms can be enlarged by all remaining wave-fronts.

On the other hand consider we choose an abstraction which maps an annotated term to the set of its maximal function symbols (wrt some *precedence ordering* on function symbols) occurring in some wave-fronts. Then the abstracted problem is to equalize two sets of function symbols with the help of given equations on function sets (derived from the abstraction of the database). The solved form $\Phi(a_n, b_n)$ indicates the set of maximal symbols which have to occur in the wave-fronts in both terms. These common maximal symbols would then be used to enlarge the skeleton in $q_n = r_n$.

Summing up, the procedure `rewrite_1` in Figure 2 has to manipulate wave-fronts in both terms in order to enable a "useful" enlargement of the skeleton. As mentioned above this enlargement corresponds to the solution in the abstract space. Thus, depending on the abstraction we use, we may shift the "burden" of proving the equation from a "lazy difference unification" to a "general equality reasoning" embedded in `rewrite_1`. In the following we sketch some useful abstractions which can be used in our approach. We illustrate these abstractions with the help of a common example in order to give the reader an impression of the different subgoals arising during the proofs.

In either of the cases below `rewrite_1` has to preserve the skeleton of the original task. Some of the heuristics above create subtasks to enable the use of some bridge lemma. Using difference matching between the subterm to be rewritten and the left-hand side of the bridge lemma will result in a common skeleton of both (which is in general *not* identical to the common skeleton of `s` and `t`). Rippling with respect to the subtask-skeleton is used to enable the use of the bridge lemma while care has to be taken not to change the skeleton of the original task.

5.1 Adaption of Maximal Symbols

The crucial point of enlarging the skeleton is the selection of appropriate common parts since in general we may enlarge the skeleton in several ways. As a result of many examples we now use the definition hierarchy of the symbols occurring in the wave-fronts to determine possible candidates for the skeleton. For example, since s and $+$ are used to define $\times$ it is awkward to enlarge the skeleton by occurrences of s or $+$ while $\times$ still occurs in some wave-front. Consider a wave-front like $s(x) \times s(x)$. If we add the first occurrence of $s(x)$ to the skeleton (obtaining $s(x) \times s(x)$), unfolding the definition of $\times$ becomes impossible due to the skeleton restrictions such that such a step would hamper rather than facilitate a proof.

Hence, we define the following definition hierarchy: given two symbols f and g, $f < g$ holds if f occurs in the definition of g. $<^*$ is defined as the transitive closure of $<$. A symbol f occurring in some wave-front c of an annotated term is called a *maximal* symbol of c iff there is no symbol g in c with $f <^* g$. Also, specific formulas give rise to a $<$-relation, like $\forall x, y\ f(x,y) = g(h(x), k(y))$ implies $g < f$, $k < f$, and $h < f$.

Thus, in order to equalize the wave-fronts of two terms q and r we map them into the multi-sets of maximal symbols of both wave-fronts obtaining an abstract form $\Phi(a,b)$. An expression $\Phi(x,y)$ is in *solved form* iff $x = y$ holds. Once we come up with a solved form $\Phi(a_n, b_n)$ for the abstract problem $\Phi(a,b)$, we map this solution into the task of modifying the original wave-fronts such that the symbols in a_n occur in both wave-fronts. Then the occurrences of these symbols in the wave-front are subject to the enlargement of the skeleton.

Thus, this heuristic manipulates the wave-fronts of both terms in order to achieve identical multi-sets of maximal symbols (wrt the definition hierarchy) on both sides. Some symbol in this intersection may be used to enlarge the skeleton. Comparing and adapting symbols of both terms results in a proof sketch which suggests to eliminate or create occurrences of specific symbols.

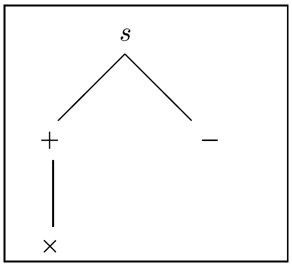

Figure 4: Symbol Hierarchy

Consider for instance the example of proving

$$(a+b) \times (a-b) = (a \times a) - (b \times b)$$

which is given in detail in section A. Due to the definition hierarchy (cf. Figure 4) $\times > +$ holds while $-$ and $\times$ (or $+$) are incompatible. After initial steps of enlarging the skeleton we obtain the equation:

$$\begin{aligned} & ((a \times a) - (a \times b)) + ((b \times a) - (b \times b)) \\ = \; & (a \times a) - (b \times b) \end{aligned}$$

The abstraction results in a form $\Phi(\{\times, \times, -, -\}, \{-\})$ which suggests the removal of occurrences of $\times$ and $-$ in order to identify the multisets of maximal symbols on both sides.

Exploring the database will result in some speculations about useful "bridge lemmas". Given an an annotated equation

$$(X - Y) + Y = X \tag{10}$$

we are able to remove at least an occurrence of $-$ in our example if we could apply the equation from left to right. Since the equation is not applicable we have to do some equality reasoning in order to refine the first abstract step on the ground space. Hence, we use a kind of difference matching between the left-hand side of the lemma and a subterm t containing the wave-front to be modified. We say a term q difference-matches a term t iff there is an annotated term t' with $erase(t) = erase(t')$ and a substitution σ such that $\sigma(q) = skel(t')$.

Since we may only change the wave-fronts of t the skeleton part of q have to match the skeleton of t. Thus we need a difference matching which takes into account the already existing annotations of the term t to be modified and of the bridge lemma as it only matches skeleton parts of t to skeleton parts of the bridge lemma.

The solution of difference matching the equation with the left-hand side of our goal equation is given in (11).

$$((a \times a) - (a \times b)) + ((b \times a) - (b \times b)) \tag{11}$$

while (12) illustrates the original annotation related to the overall equality reasoning problem

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

Notice, that we obtain two different annotations related to the two different tasks. While we enable the use of the bridge lemma (10) both skeletons have to be preserved. For instance the subterm $(b \times b)$ although part of the wave-front in (11) may not be changed since it is also part of the skeleton of the (12).

In order to ripple the wave-front in (11) outside in the above example we use an annotated equation

$$X + (Y - Z) = (X + Y) - Z \tag{13}$$

Applying this annotated equation results in

$$(((a \times a) - (a \times b)) + (b \times a)) - (b \times b) \tag{14}$$

The erasure of this annotated term can also be annotated in a different manner such that it preserves the skeleton of (12)

$$(((a \times a) - (a \times b)) + (b \times a)) - (b \times b) \tag{15}$$

which is the result of applying a differently annotated version of (13)

$$X + (Y - Z) = (X + Y) - Z \tag{16}$$

to (12). With (14) the use of (10) is enabled and its application identifies the multi-sets of maximal symbols on both sides of the original goal:

$$(a \times a) - (b \times b) = (a \times a) - (b \times b) \tag{17}$$

The example A.1 in the appendix will illustrate the complete proof of the lemma.

As we have seen in the example above, the abstraction of maximal symbols will usually allow only to plan one step ahead in the abstract space since the effects of applying equations to the goal term can hardly be traced on the abstract level. For instance removing $\times$ may result in $+$ being now a maximal symbol but in the abstract space $+$ has not been monitored as long as $\times$ occurred in the goal. Thus in the abstract space there is no information available whether $+$ still occurs inside the term to be modified.

5.2 Paths to Common Parts of the Skeleton

As we have seen in the previous section, the abstraction to maximal symbols preserves to little information to plan in advance several deduction steps in the abstract space. The next abstraction [Aut96] suggests equalizing the paths in both terms from the top-level to a specific part of the skeleton. In case of success the symbols on this path will be added to the skeleton in the next step. This heuristic produces a proof sketch which suggests the manipulation of "paths" to specific occurrences of terms. Expanding this proof sketch to a first order proof will create a chain of bridge lemmas to be applied on `s` and `t`.

Consider again the given goal in our previous example:

$$((a \times a) - (a \times b)) + ((b \times a) - (b \times b)) = (a \times a) - (b \times b) \tag{18}$$

Extracting the function symbols occurring on the way from top-level to the occurrences of the common skeleton-part $a \times a$, we obtain the list $\langle +, - \rangle$ for the left-hand and $\langle - \rangle$ for the right-hand side. If we add also the argument positions in which the skeleton part occurs as subscripts to the symbols then we obtain the lists $\langle +_1, -_1 \rangle$ and $\langle -_1 \rangle$ respectively. Analogously, we may abstract the equations given in our database: From (10) we obtain

$$\langle +_1, -_1 \rangle = \langle \rangle \tag{19}$$

and the abstraction of the erased version of (13) results in:

$$\langle +_1 \rangle = \langle -_1, +_1 \rangle, \tag{20}$$

$$\langle +_2, -_1 \rangle = \langle -_1, +_2 \rangle, \tag{21}$$

$$\langle +_2, -_2 \rangle = \langle -_2 \rangle, \tag{22}$$

Thus, the problem of equating the wave-fronts is reduced to the problem of equalizing the strings of indexed symbols with the help of these abstracted equations on strings of indexed symbols. A solution of the above problem is given by the following deduction in the abstract space:

$$\langle +_1, -_1 \rangle =_{(20)} \langle -_1, +_1, -_1 \rangle =_{(19)} \langle -_1 \rangle$$

which suggests the use of a version of equation (13) followed by the use of (10). Refining this abstract solution to the ground space results in the same deduction steps as in the previous section. But in contrast to the previous abstraction we have planned the complete adaption on the abstract space before we refined the obtained proof sketch to the ground space.

In case we choose the lists of function symbols occurring on the path to the second part $b \times b$ of the skeleton we obtain obtain a solution

$$\langle +_2, -_2 \rangle =_{(22)} \langle -_2 \rangle$$

suggesting only the application of equation (13) which would already equalize the paths to this skeleton part $b \times b$.

In contrast to the given example, refining a proof sketch into the ground space will result in several subgoals in order to enable the use of the speculated equations.

5.3 Rewriting

This heuristic will rewrite both wave-fronts wrt. a given set of rewrite-rules. The skeleton parts of the goal equation are regarded as constants and are not subject to any rewriting.

Consider again our example of proving

$$((a \times a) - (a \times b)) + ((b \times a) - (b \times b)) = (a \times a) - (b \times b) \tag{23}$$

This time may use rewrite-rules generated from the equations given in the database in order to rewrite the corresponding wave-fronts. The terms $a \times a$ and $b \times b$ are regarded as constants and will not be rewritten. Depending on the set of rewrite-rules we compute from the existing database we are able to prove or at least to simplify the given goal.

5.4 Inbuilt Theories

Integrating decision procedures for inbuilt theories like the sup-inf method or other methods for linear arithmetic in a first-order setting causes usually very complex interactions between the decision procedure and the first-order calculus in order to handle conditional equations or user-defined functions properly.

In contrast our approach supports the use of these decision procedures as it builds up step by step a common skeleton. This skeleton consists of a set of common subterms of the terms q and r. In case corresponding wave-fronts consist only of symbols of our inbuilt theory we may abstract from the shape of the skeleton and replace each member of the skeleton in both terms by a new constant. Thus, we obtain a problem in the in-built theory which can be tackled by the decision procedure. We then use the obtained solution in the abstract space to equalize the wave-fronts and to enlarge the skeleton by the identical wave-fronts.

In our common example we abstract $a \times a$ to c and $b \times b$ to d and obtain the problem of proving the following equation on integers:

$$(c - (a \times b)) + ((b \times a) - d) = c - d$$

On we have proven this equation we may equalize both wave-fronts by replacing the one wave-front by the other ones which results in:

$$(a \times a) - (b \times b) = (a \times a) - (b \times b).$$

Since both wave-fronts are identical we enlarge the skeleton to

$$(a \times a) - (b \times b) = (a \times a) - (b \times b).$$

6 Rewriting the Skeleton

As already mentioned above, the skeleton guiding the proof has mainly two effect. Given an equality proof of two annotated terms q, and r by modifying $q \to q^1 \to \ldots \to q^n$ and $r \to r^1 \to \ldots \to r^m$ with $q^n = r^m$. Since q and r share the same skeleton and all deduction steps preserve that skeleton, $q^n (= r^m)$ still contain the skeleton of q and r. Thus, the skeleton of q and r is a pattern of the goal term $q^n (= r^m)$.

On the other hand enlarging the skeleton by a common function symbols imposes also a link between the two selected occurrences of the symbol. During the ongoing deduction this link has to be inherited and in case the proof succeeds it will finally link two identical subterms.

There are cases in which we want to impose such a link between different subterms and inherit it until it finally links two identical subterms but in which the also implied first mentioned property that the skeleton stays invariant (or will only be increased) is to restrictive to guide a successful proof search.

As an example suppose we want to prove the equation

$$\mathit{length}(\mathit{app}(\mathit{cons}(u,x),y)) = \mathit{length}(\mathit{app}(x,\mathit{cons}(v,y))) \tag{24}$$

with the help of the given equations

$$\begin{aligned} X+Y &= Y+X && (25)\\ s(X)+Y &= s(X+Y) && (26)\\ \mathit{length}(\mathit{cons}(X,y)) &= s(\mathit{length}(Y)) && (27)\\ \mathit{length}(\mathit{app}(X,Y)) &= \mathit{length}(X)+\mathit{length}(Y) && (28) \end{aligned}$$

Further suppose, due to some heuristics guiding the skeleton-enlargement we end up with the following annotated equation:

$$\mathit{length}(\mathit{app}(\mathit{cons}(u,x),y)) = \mathit{length}(\mathit{app}(x,\mathit{cons}(v,y))) \tag{29}$$

The proof gets stuck since there are no appropriate annotated equations available which would allow to equalize the wave-fronts occurring in the different arguments of *app*.

Reformulating the skeleton in the goal we use (27) twice to obtain:

$$\mathit{length}(\mathit{cons}(u,x)) + \mathit{length}(y) = \mathit{length}(x) + \mathit{length}(\mathit{cons}(v,y)) \tag{30}$$

Notice that the skeletons on both sides of the equation again coincide and that the linking of corresponding subterms is inherited through the change of skeletons. Rippling the wave-fronts out of the arguments of + (using the commutativity law) results in the following deduction steps

$$s(\mathit{length}(x)) + \mathit{length}(y) = \mathit{length}(x) + s(\mathit{length}(y))$$

$$s(\mathit{length}(x) + \mathit{length}(y)) = s(\mathit{length}(x) + \mathit{length}(y))$$

which finally proves the theorem.

Another example in which a rewriting of the initial skeleton is required is the proof the **lim**-example [Ble90]:

$$\mathit{lim}(\lambda x.f(x),a,l) \wedge \mathit{lim}(\lambda x.g(x),a,m) \rightarrow \mathit{lim}(\lambda x.(f(x)+g(x)),a,l+m)$$

Although this example is not an equality problem we can use the same techniques described above to equalize either one or both premises and the conclusion of the implication. Enlargement of the skeleton will result in an annotated implication.

$$\mathit{lim}(\lambda x.f(x),a,l) \wedge \mathit{lim}(\lambda x.g(x),a,m) \rightarrow \mathit{lim}(\lambda x.(f(x)+g(x)),a,l+m)$$

In order to prove this statement we have to open up the definition of *lim*

$$\begin{aligned} &\lim(f,a,l) \equiv \\ &(\forall \epsilon\ (0<\epsilon \rightarrow \exists \delta\ (0<\delta \wedge \forall x\ (x \neq a) \wedge |x-a|<\delta \rightarrow |f(x)-l|<\epsilon))) \end{aligned}$$

After unfolding this definition the occurring wave-fronts can be easily equalized in the reformulated theorem (see [YoBuGrWaBa94] for a complete proof of this theorem).

Reformulating the skeleton involves another branching point in the search space and the question arises how and when do we guide possible rewriting of the skeleton?

The rewriting has to be done on both sides of the goal equation simultaneously and we have to take care that each step on one side can be also performed on the other side. In order to ensure this, we only use equations for rewriting with the property that each function symbol on the left-hand side of the equation matches with some skeleton part. Thus, wave-fronts will only match which the instantiation of some variable occurring on the left-hand side of the equation. Also all variables occurring on the right-hand side of the rewrite-equation have to occur on the left-hand side. Since we apply the same equation on both sides at the same position wrt the skeleton both replacing subterms differ at most in the instantiations of the right-hand side variables. Hence, we add all function symbols of the non-instantiated right-hand side to the skeleton and relate corresponding symbols to each other. The annotation of the instantiated variables is inherited from the previous term. In case each variable occurs only once on the left-hand there is a unique annotation for all instantiated occurrences of the variable on the right-hand side. But in case the variables occurs several times on the left-hand side we have to choose which occurrence we select to annotate the instantiation of the right-hand side appropriately.

Rewriting of the skeleton applies if the rippling of the wave-fronts is blocked as in the examples above. In 29 the wave-front $\mathit{cons}(v, \ldots)$ is blocked since no applicable wave-rule is available to ripple it out. Thus we would like to manipulate the skeleton in such a way that this wave-front is moved into a position in which wave-rules are applicable. Since the wave-front will not be changed during the rewrite of the skeleton we guide the rewrite by computing appropriate function symbols as skeleton parts which allow one to ripple out the corresponding wave-front. In our example of proving (24) we have to ripple out *cons*. The only available wave-front to do this is (27) which restricts the possible goals of rewriting $\mathit{length}(\mathit{app}(\ldots, \ldots))$ to the term pattern $\ldots \mathit{length}(\ldots) \ldots$ One may think of a typical rippling problem and may use rippling techniques to obtain the solution of using (28) to reformulate the skeleton.

7 Human Computer Interface

The techniques illustrated in this paper result in a hierarchical planning of proofs. According to a *divide and conquer* approach we split a given problem into a set of smaller problems. Also, the use of *abstractions* results in a decomposition of existing problems as it maps the problem of solving an equation to the problem of refining a proof sketch obtained from a solution in some abstraction space. This hierarchy may be also used to present the proof (or the proof attempt) to the user in a hierarchical way. Since this approach is highly goal-oriented also the intentions of the system can be shown to be user (e.g. which subterms should be correlated to each over, how to resolve horizontal clashes, which abstractions should be used to adapt the wave-fronts, etc.). Vice versa, with the help of this vocabulary the user is able to influence the behavior of the system on a strategic level as he may specify for example the kind of abstraction to be used in a concrete deduction step or the bridge lemma to be used in order to refine a solution of the abstract space.

8 Extension to General Theorem Proving

In many calculi the proof of a formula is reduced to the problem of manipulating its different parts. Not only when proving an equation two terms have to be transformed until they finally coincide but also with resolution based provers the resolution rule can only be applied if there are two complementary literals with unifiable termlists. Again, using paramodulation, the task of equalizing terms occurs in order to enable the resolution rule. Thus, proving the equality of terms or equivalences of formulas by identifying their syntactical structure is also a main goal during the proof of general formulas.

In order to embed our approach in a general framework of planning proofs in a general first- (or even higher-) order theory we need additional heuristics describing how the proof of arbitrary formula can be decomposed to the sub-problems of equalizing terms or termlists.

In a first-order setting (especially the resolution calculus) the approach of Plaisted [Pla81] can be used to compute a propositional proof sketch of the formula by neglecting all the arguments of the predicates occurring either in the axioms or in the theorem. Then, based on this proof sketch we try to lift this solution to first-order by equalizing the termlists of corresponding literals upon which we want to resolve.

A more general framework to decompose proof problems to problems of equalizing terms can be obtained by using the approach of Negrete-Yankelevich [Neg96] which covers also non-first order logics.

9 Further Work

Although the approach presented in this paper is able to proof some challenging equality problems (cf. the proof of SAM's lemma in the appendix) it still lacks of heuristics which guide the prover in some specific cases and which are subject to further development:

Variables Variables occurring within the terms to be equalized hamper the use of the abstraction of maximal symbols since there are cases in which rewriting of both sides will not result in a common maximal symbol. For instance given the usual recursive definition of *app* consider the equality problem $\exists X, Y \; app(cons(a,b),c) = cons(X,Y)$. Any rewrite to introduce *app* on the right-hand side or to delete *app* on the left-hand side will fail. But the approach works well if we drop the need for maximal symbols and choose *cons* as a common symbol. The question arises how do we guide an appropriate selection of non-maximal symbols; or, how do we correlate maximal symbols to variables or variables to variables?

Permutations `rewrite_2` in Figure 2 is mainly involved with the permutation of arguments of a function symbol f. So the question arises whether there is a suitable meta-theory of permutations which allow one to calculate appropriate proof sketches in order to permute the list of arguments of f in a given specific order?

Blowing up terms The method of difference reduction guides the proof search in order to minimize the syntactical differences. Especially in group theory (or e.g. in Ternary Boolean Algebra) there are theorems the proofs of which require to blow up both terms and thus to increase enormously the syntactical differences before any difference unification between both terms can take place. But the abstractions presented in this paper are of no use to guide this expansion process, so further work will be concerned with the development of appropriate abstractions usable to those kind of problems.

10 Conclusion

In many calculi the proof of a formula is reduced to the problem of manipulating its different parts. In order to prove an equation, for example, both sides of the equation have to be transformed until they finally coincide. Alternatively, with resolution based provers the resolution rule can only be applied if there are two complementary literals with unifiable termlists. Again, using paramodulation, the task of equalising terms occurs in order to enable the resolution rule. Thus, proving the equality of terms or equivalences of formulas by identifying their syntactical structure is a main goal during the proof of formulas. According to the paradigm of difference reduction, the syntactical differences guide the process of modification in our approach.

Planning intermediate steps with the help of these abstractions allows one to construct proofs in a hierarchical way. In a first step we have to compute a common skeleton. Next, the wave-fronts denoting the syntactical differences have to be moved into corresponding positions (wrt. the common skeleton) and finally the "shape" of the wave-front has to be identified in order to enlarge the common skeleton. We use so-called maximal symbols to split existing goals of equalizing formulas into smaller subgoals. In case of maximal symbols occurring only on one side the subgoals describe the deletion of the occurrences of those symbols. In case both sides share the same set of maximal symbols some of their occurrences are related between both sides and the subgoal is created to equalize the related subterms. In case a maximal symbol occurs several times on either side heuristics guide the selection of the "appropriated" once. These heuristics take into account the set of variables and the nested structure of maximal symbols occurring in their arguments. This procedure will be iterated until all wave-fronts have been identified and added to the skeleton.

Refining an abstract step into a more concrete deduction will usually require a sequence of additional deduction steps in order to enable either the predicted rule or to guarantee its preconditions. These additional steps form a subproof which is "only" useful in case the application of the rule can be enabled. Hence, in a natural way this methodology results in a hierarchical structuring of proof search. Each deduction step is done in order to achieve a specific goal which itself may be again part of a superior goal. Also this approach will recursively invoke new equality problems: Heuristics may select appropriate lemmata which should be used in order to achieve the asserted goal. Since in general a lemma may not immediately be applicable some modifications have to be done to the formula (resp. to the lemma) in order to enable its use. A subgoal arises requiring the matching of the appropriate subterm with the corresponding side of the lemma. After solving this subgoal, the lemma is applied and the proof attempt continues with its previous goal.

A Examples

A.1 An integer problem

Consider the following equation problem on INTEGER:

$$(a+b)\times(a-b)=(a\times a)-(b\times b)$$

given the following equations:

$$\begin{aligned}
X+Y &= Y+X & (31)\\
(X+Y)+Z &= X+(Y+Z) & (32)\\
X\times Y &= Y\times X & (33)\\
(X-Y)+Y &= X & (34)\\
X+(Y-Z) &= (X+Y)-Z & (35)\\
(X+Y)\times Z &= (X\times Y)+(X\times Z) & (36)\\
X\times(Y-Z) &= (X\times Y)-(X\times Z) & (37)
\end{aligned}$$

We start with an empty skeleton:

$$(a+b)\times(a-b) \quad \text{and} \quad (a\times a)-(b\times b)$$

1. Since the maximal symbols of both sides $\{\times,-\}$ coincide `rewrite_1` does not perform any modification
2. Enlargement of the skeleton by $\times$ (selecting the second occ of $\times$ would actually result in the same proof; see below):

$$(a+b)\times(a-b) \quad \text{and} \quad (a\times a)-(b\times b)$$

3. `rewrite_2` does not perform any modification
4. Procedure enters recursive call with the first argument of $\times$:
   (a) Since one side of the equation is a subterm of the other the skeleton is immediately enlarged by a (no rippling out is necessary):

$$(a+b) \quad \text{and} \quad a$$

5. Procedure enters recursive call with the second argument of $\times$:
   (a) Since one side of the equation is a subterm of the other the skeleton is immediately enlarged by a (no rippling out is necessary):

$$(a-b) \quad \text{and} \quad a$$

6. Rippling out the wave-front on the left-hand side:

$$\begin{aligned}
& (a+b)\times(a-b) && \text{and} && (a\times a)-(b\times b)\\
\rightarrow_{(36)}\; & (a\times(a-b))+(b\times(a-b)) && \text{and} && (a\times a)-(b\times b)\\
\rightarrow_{(37)}\; & ((a\times a)-(a\times b))+(b\times(a-b)) && \text{and} && (a\times a)-(b\times b)
\end{aligned}$$

7. Process iterates for the second occurrence of $\times$ which finally results in:

$$((a\times a)-(a\times b))+((b\times a)-(b\times b)) \quad \text{and} \quad (a\times a)-(b\times b)$$

8. Now there is a difference of maximal symbols on both sides. `rewrite_1` tries first to eliminate $\times$ on the left hand side by some equation but fail. Then, it $+$ is eliminated since $+$ is a symbol occurring on the left-hand side and also a maximal symbol wrt. right-hand side. This introduces new annotations of the terms. We indicate the original skeleton (which has still to be preserved) by underlining the corresponding bits.

   (a) Difference match with (34) (wrt. commutativity law) results in:

   $$(\underline{(a \times a)} - (a \times b)) + ((b \times a) - \underline{(b \times b)})$$

   (b) Rippling out with (35) results in:

   $$((\underline{(a \times a)} - (a \times b)) + (b \times a)) - \underline{(b \times b)}$$

   (c) Applying the equation (34) results in:

   $$\underline{(a \times a)} - \underline{(b \times b)}$$

9. now things are ready to return to the original problem (also returning to the original annotations):

   $$(a \times a) - (b \times b) \quad \text{and} \quad (a \times a) - (b \times b)$$

   Enlargement of the skeleton yields finally

   $$(a \times a) - (b \times b) \quad \text{and} \quad (a \times a) - (b \times b)$$

which proves the equality of both terms.

A.2 SAM's Lemma

We will illustrate our approach by another example of SAM's lemma [GOBS69] which is still a challenge equality problem although it has also been solved by some resolution provers without paramodulation.

Given two associative, commutative, and idempotent functions f and g and additionally, the following set of axioms:

$$\begin{aligned} f(X, g(X, Y)) &= X && (38) \\ g(X, f(X, Y)) &= X && (39) \\ f(0, X) &= 0 && (40) \\ f(1, X) &= X && (41) \\ g(0, X) &= X && (42) \\ g(1, X) &= 1 && (43) \\ f(X, Z) = X \rightarrow f(g(X, Y), Z) &= g(X, f(Y, Z)) && (44) \end{aligned}$$

the task is to prove the following formula:

$$f(f(y, u), v) = 0 \wedge f(x, g(u, v)) = 0 \rightarrow f(g(x, f(y, u)), g(x, f(y, v))) = x$$

We start with an empty skeleton for the conclusion of the theorem:

$$f(g(x, f(y, u)), g(x, f(y, v))) = x$$

1. We use the abstraction of maximal symbol to determine the next steps of `rewrite_1` and compare the maximal symbols of both sides. Since there are no function hierarchies f and g are considered as maximal symbols. Both symbols occur only on the left hand side. Hence, we are looking for an appropriate equation which would remove occurrences of f or g. From all possible equations only the left hand side of (38) and the left hand side of the first premises difference-match against the left hand side of the conclusion.

   (a) Using a set-of-support heuristic the system prefers the first premise and tries to enable its application within the left hand side of the conclusion. Hence, we obtain the following annotated conclusion in order to solve this subgoal:
   $$f(g(x, f(y, u)), g(x, f(y, v))) = x$$

   (b) We apply standard rippling techniques to move the wave-fronts out of the skeleton and thus, to enable the use of the first premise. Using the conditional annotated equation of (44):
   $$f(X, Z) = X \rightarrow f(g(X, Y), z) = g(X, f(Y, Z))$$
   we are able to manipulate the conclusion to
   $$g(x, f(f(y, u), g(x, f(y, v)))) = x \tag{45}$$
   since the instantiated condition $f(x, g(x, f(y, v))) = x$ of (44) holds trivially due to axiom (38).

   (c) The problem remains to move the second context 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 $f(x, f(y, u)) = x$ which now prevents the use of the equation. Also, the application of the annotated equation (42) fails due to a clash inside the unification. Thus the rippling process gets stuck.

   (d) In order to get the process back on its feet, unblocking strategies for rippling are used to overcome the unification clash. (42) would be applicable if 0 occurs instead of x as the first argument of g. Considering the second premise $f(x, g(u, v)) = 0$, we reformulate this problem as a rippling problem: we have to create an appropriate context around the occurrence of x inside the first argument of g in order to obtain a formula
   $$g(x, f(f(y, u), g(f(x, g(u, v)), f(y, v)))) = x$$
   Alternating backward and forward reasoning this problem is reduced to the task of removing the context within the term
   $$g(x, f(f(y, u), g(f(x, g(u, v)), f(y, v))))$$
   Notice, that we have now reached level four of our proof hierarchy: In order to equalize both sides of the equation we decided to eliminate occurrences of f and g. To enable the use of a suitable equation we have to ripple out some wave-fronts and in order to that we have to enable the use of an appropriate annotated equation. Corresponding to the different levels we have also different skeletons which have *all* to be preserved during the single deduction steps.
   Removing the context within the term
   $$g(x, f(f(y, u), g(f(x, g(u, v)), f(y, v))))$$
   is done with usual rippling techniques.

i. Applying (44) with the help of the commutativity law we obtain:

$$g(x, f(f(y,u), f(g(u,v), g(x, f(y,v))))) \tag{46}$$

ii. Using the associativity law of f results in

$$g(x, f(f(f(y,u), g(u,v)), g(x, f(y,v))))$$

and

$$g(x, f(f(y, f(u, g(u,v))), g(x, f(y,v))))$$

iii. Finally, we are able to delete the wave-front with the help of (38):

$$g(x, f(f(y,u), g(f(y,v), x)))$$

(e) Thus, the left hand side of (45) is reformulated which results in

$$g(x, f(f(y,u), g(f(x, g(u,v)), f(y,v)))) = x$$

(f) Using the intended bridge lemma (the second premises) yields the following result

$$g(x, f(f(y,u), g(0, f(y,v)))) = x$$

(g) which now allows one to remove the wave-front with the help of (42):

$$g(x, f(f(y,u), f(y,v))) = x$$

(h) Applying associativity and commutativity law of f finally yields

$$g(x, f(f(f(y,u), v), y) = x$$

which enables the use of the first premises.

2. We are left with the problem of solving

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

3. Again the abstraction of maximal symbol is used to determine the next steps of `rewrite_1`. (40) is chosen to delete the occurrence of f. The lemma is trivially applicable and we obtain

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

4. The next task is to remove the occurrence of g which is done by (42) and which results in the trivial task of proving:

$$x = x$$

Notice that the complete proof of SAM's lemma is done within the first call of `rewrite_1`. The adaption of maximal symbols actually results in the trivial task of solving $x = x$. On the other hand this example involve tricky rippling heuristics. In order to get rid of the wave-fronts in (46) we had to ripple in the wave-fronts towards the occurrence of u where we were able to apply an annotated version of equation (38) to get rid of the wave-front.

References

[Aut96] S. Autexier Heuristiken zum Beweisen von Gleichungen *MSc Thesis*, University of Saarland, 1996

[Ble90] W. Bledsoe Challenge problems in elementary calculus Journal of Automated Reasoning, Vol. 6(3), pp. 341-359, 1990

[BS88] K. Hans Bläsius and J.H. Siekmann. Partial unification for graph based equational reasoning. In *9th CADE*, Springer, LNCS, 1988

[BSvH+93] A. Bundy, A. Stevens, F. van Harmelen, A. Ireland, and A. Smaill. Rippling: A heuristic for guiding inductive proofs. *Artificial Intelligence*, 62, pp. 185–253, 1993.

[BvHS91] A. Bundy, F. van Harmelen, J. Hesketh, and A. Smaill. Experiments with proof plans for induction. *Journal of Automated Reasoning*, 7, 1991.

[BW93] D. Basin and T. Walsh. Difference unification. In *13th IJCAI*, Morgan Kaufmann, 1993.

[BW95] D. Basin and T. Walsh. A Calculus for and Termination of Rippling. To appear in the *Journal of Automated Reasoning*, 1995.

[CH94] J. Cleve, D. Hutter. A methodology for equational reasoning. In: *Hawaii International Conference on System Sciences 27*, Eds : Jay F. Nunamaker, jr. and Ralph H. Sprague, jr., IEEE Computer Society Press, 1994

[Dig85] V.J. Digricoli. The management of heuristic search in boolean experiments with RUE resolution. In *9th IJCAI*, pp. 1154 – 1161, Los Angeles, California, 1985.

[GOBS69] J. Guard, and F. Oglesby, and J. Bennett, and L. Settle. Semi-automated mathematics. In *Journal of the ACM*, pp. 49-62, Vol. 16, 1969

[GW92a] F. Giunchiglia and T. Walsh. A Theory of Abstraction. *Artificial Intelligence*, 56(2–3):323–390, 1992. Also available as DAI Research Paper No 516, Dept. of Artificial Intelligence, Edinburgh.

[GW92b] Fausto Giunchiglia and Toby Walsh. Tree subsumption: Reasoning with outlines. In B. Neumann, editor, *10th ECAI*, pages 77–81. Wiley & Son, 1992.

[Hut90] D. Hutter. Guiding inductive proofs. In M.E. Stickel, editor, *10th CADE*, Springer, LNAI 449, 1990

[Hut] D. Hutter. Colouring terms to control equational reasoning. Journal of Automated Reasoning, forthcoming

[Hut96] Hutter, D. Using Rippling for Equational Reasoning. 20. Jahrestagung Künstliche Intelligenz, LNAI, Springer, 1996

[HS96] D. Hutter, and C. Sengler INKA - The Next Generation In J. Slaney, M.McRobbie, editor, *13th CADE*, Springer, LNAI, 1996

[Mor69] J. Morris. E-resolution: an extension of resolution to include the equality relation. In *IJCAI-69*, 1969.

[Neg96] S. Negrete-Yankelevich Proof Planning with Logic Presentations PhD Thesis, University of Edinburgh, 1996

[NSS63] A. Newell, J.C. Shaw, and H.A. Simon. The logic theory machine. In Feigenbaum and Feldman, editors, *Computers and Thought*, pp. 61–79. McGraw-Hill, 1963.

[Pla81] David Plaisted. Theorem proving with abstractions. *Artificial Intelligence*, 16:47–108, 1981.

[VSE96] D. Hutter, B. Langenstein, C. Sengler, J. Siekmann, W. Stephan, A. Wolpers Deduction in the Verification Support Environment. In *FME-96*, Oxford, Springer, LNCS, 1996

[YoBuGrWaBa94] T. Yoshida, A. Bundy, I. Green, T. Walsh, D. Basin Coloured Rippling: An extension of a theorem proving heuristic In A.G.Cohn, editor, Proceedings of ECAI-94, pp. 85-89, John Wiley & Sons, 1994