**Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH**

**Research
Report**

RR 97-04

Parameterized Abstractions used for Proof-Planning

Serge Autexier, Dieter Hutter

March 1997

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341

Parameterized Abstractions used for Proof-Planning

Serge Autexier, Dieter Hutter

DFKI-RR 97-04

This work has been supported by a grant from The Federal Ministry of Education, Science, Research and Technology (FKZ ITWM-9600).

ISSN 0946-008X

Parameterized Abstractions used for Proof-Planning

Serge Autexier, Dieter Hutter

{Serge.Autexier, Dieter.Hutter}@dfki.uni-sb.de

Abstract

In order to cope with large case studies arising from the application of formal methods in an industrial setting, this paper presents new techniques to support hierarchical proof planning. Following the paradigm of difference reduction, proofs are obtained by removing syntactical differences between parts of the formula to be proven step by step. To guide this manipulation we introduce dynamic abstractions of terms. These abstractions are parameterized by the individual goals of the manipulation and are especially designed to ease the proof search based on heuristics. The hierarchical approach and thus the decomposition of the original goal into several subgoals enables the use of different abstractions or different parameters of an abstraction within the proof search. In this paper we will present one of these dynamic abstractions together with heuristics to guide the proof search in the abstract space.

1 Introduction

The verification of software components of realistic size results in proof obligations that due to their intrinsic technical size and complexity cannot be treated completely automatically, but require human interaction and guidance. On the other hand we are faced with a considerable number of these proof obligations, many of which may lead to proofs of several thousand steps. Both aspects result in the development of an appropriate proof planning approach in order to deal with large sets of axioms and also to allow the user to interact on a strategic level.

Using proof planning allows one to construct proofs in a hierarchical manner by decomposition of the given goals in a sequel of subgoals. In our setting the decomposition is obtained by using abstractions. According to [Giunchiglia and Walsh, 1992] abstractions are mappings of a representation of a problem, the *ground* problem, into a new representation, the *abstract* problem. Solving the abstract problem results in a proof sketch in the ground space which guides the search. In the past a series of abstractions have been investigated and used for proof planning with little success. Basically, these abstractions map formulas into some simplified structure (e.g., abstractions into sets of involved symbols or abstractions ignoring the termlist of literals). As a result these abstractions either drop too much information and thus, planning in the abstract space is rather unconstrained, or the proof search in the abstract space has more or less the same complexity as on the ground space.

Following the paradigm of difference reduction we plan a proof by removing syntactical differences between parts of the formula to be proven (e.g. left- and right-hand side of an equation) and use a rippling calculus to maintain the differences between these parts. Within this rippling calculus we are able to annotate specific information (colors) to each occurrence of a symbol in the formula indicating whether this occurrence belongs to the common part (the *skeleton*) of both formulas or whether it is part of their differences (*wave-fronts*). Any manipulation of the annotated formula within the calculus will automatically focus on the differences and keep the skeleton parts unchanged.

Starting with an empty skeleton we have to manipulate both parts until they share some common structure. Adding the obtained common parts to the skeleton will keep them from being manipulated and focus the attention of the prover to the remaining differences. Iterating this process will step by step remove the differences until both terms coincide. In order to equalize the differences we use

the knowledge of a common skeleton to compute appropriate abstractions of the problem. Thus, different abstractions may be used in different phases of the proof since they are parameterized by the common skeleton. Once a solution is found in the abstract space it has to be reformulated in the ground space.

In the following we restrict ourselves on equality problems and present an abstraction and heuristics on this abstraction that are tailored to this approach. The abstraction of a formula is parameterized by the given skeleton and the attention of the prover is focused to the sequel of function symbols governing the occurrences of the skeleton.

2 A Commented Example

To illustrate our ideas we will present a small example in the field of lattice-ordered groups. In the following we inspect a proof of the theorem `GRP175-1` of the TPTP library (cf. [Sutcliff and Suttner, 1996]):

$$\begin{aligned} &\forall x, y\ u(1, y) = y \rightarrow \\ &\qquad u(1, i(x) \times (y \times x)) = i(x) \times (y \times x) \end{aligned} \tag{1}$$

Besides others, the following formulas are part of the axiomatization:

$$\forall X\ 1 \times X = X \tag{2}$$

$$\forall X\ i(X) \times X = 1 \tag{3}$$

$$\forall X, Y, Z\ X \times u(Y, Z) = u(X \times Y, X \times Z) \tag{4}$$

$$\forall X, Y, Z\ u(X, Y) \times Z = u(X \times Z, Y \times Z) \tag{5}$$

Proving the given theorem results in proving the equality

$$u(1, i(x) \times (y \times x)) = i(x) \times (y \times x) \tag{6}$$

assuming

$$u(1, y) = y \tag{7}$$

Following the paradigm of difference reduction we first compare the function symbols occurring on both sides of the equation (6). Since u occurs on the left-hand side but not on the right-hand side we have to get rid of the occurrence of u in (6) which suggests the use of (7) as a bridge lemma. Thus, we establish a subgoal to enable the use of (7), which results in a transformation of the left-hand side of (6) into a term, where $u(1, y)$ occurs as a subterm. During this transformation, u occurring on the top level of the left-hand side of (6) has to be "moved inside" toward the occurrence of y. Inspecting our database, the application of both,

(4) and (5) (applying from right to left) would move an occurrence of u inside some argument of $\times$. Thus, applying one of these equations twice would move the occurrence of u close to y. Hence, our proof sketch consists of the successive application of either (4) or (5) and the use of (7). Unfortunately, neither (4) nor (5) are immediately applicable. To enable the use of (4) on the left-hand side of (6) we have to modify the first argument of $\times$, which is done with the help of equation (3):

$$u((i(x) \times x), i(x) \times (y \times x)) = i(x) \times (y \times x) \tag{8}$$

Now, equation (4) is applicable and its use results in

$$i(x) \times u(x, y \times x) = i(x) \times (y \times x) \tag{9}$$

Again, the use of equations (4) or (5) which would move u towards y is blocked, and we have to apply (2) to enable the application of (5):

$$i(x) \times u(1 \times x, y \times x) = i(x) \times (y \times x) \tag{10}$$

Applying (5) yields

$$i(x) \times (u(1, y) \times x) = i(x) \times (y \times x) \tag{11}$$

which allows us to use the governing condition (7)

$$i(x) \times (y \times x) = i(x) \times (y \times x). \tag{12}$$

The central idea of the above proof was to move the occurrence of u towards the occurrence of y in order to apply the condition (7). Thus, the main steps of the proof are the applications of (4) and (5) followed by the use of (7). All the other proof steps — e.g., the use of (3) and (2) — are done to achieve subgoals established by the intended application of the mentioned equations.

In order to automate such a proof we compute a *proof sketch* which *abstracts* from these preparation steps. We focus on the main outline of the proof which is in our example the move of u within the left-hand side of the theorem. Thus, we are interested in the path from top level to the occurrence of y and how close u is located to y. For example in (6) y occurs in position $\langle 2, 2, 1 \rangle$ and is governed by a sequel of functions $u, \times, \times$ while after the application of (4) y still occurs at $\langle 2, 2, 1 \rangle$ but is now governed by the sequel $\times, u, \times$.

3 Abstraction

In the previous example we measured the progress of the proof by comparing the paths from top level to the occurrence of y which denoted the invariant part or

skeleton of our example. In this section we will formalize this idea into a notion of $\mathcal{S}$-terms that are specific abstractions of terms.

In a first step we enrich occurrences π of a term t by the function symbols occurring along the denoted path from the top level to the denoted subterm $t|\pi$. Thus, an *enriched occurrence* $\tilde{\pi}$ is a sequel of function symbols where each symbol is indexed by an argument position. For instance, $\langle u_1, \times_1 \rangle$ is an enriched occurrence of $u(X,Y) \times Z$ corresponding to the standard occurrence $\langle 1, 1 \rangle$. Furthermore, we define that two enriched occurences are *independent*, if and only if one is not a prefix of the other.

Each enriched occurrence $\tilde{\pi}$ of some t denotes a subterm $t|\tilde{\pi}$. Thus, $\xrightarrow{\tilde{\pi}} t|\tilde{\pi}$ describes a specific subterm $t|\tilde{\pi}$ of t and the information about the path $\tilde{\pi}$ from top level to its occurrence. A set $\mathcal{T} = \{\xrightarrow{\tilde{\pi}_1} t|\tilde{\pi}_1, \ldots, \xrightarrow{\tilde{\pi}_n} t|\tilde{\pi}_n\}$ is called an $\mathcal{S}$-*term* if all $\tilde{\pi}_i$ denote independent positions. An $\mathcal{S}$-term abstracts from all parts of t which are not on the path to one of the specified subterms $t|\tilde{\pi}_i$. The interpretation of an $\mathcal{S}$-term $\mathcal{T} = \{\xrightarrow{\tilde{\pi}_1} u_1, \ldots, \xrightarrow{\tilde{\pi}_n} u_n\}$ is the set of terms t for which $\mathcal{T}$ is a legal abstraction, i.e., $\tilde{\pi}_i$ are enriched occurrences of t and $t|\tilde{\pi}_i = u_i$. The empty set is an $\mathcal{S}$-term which denotes *all* terms while $\{\xrightarrow{\langle\rangle} t\}$ characterizes exactly t.

In order to manipulate $\mathcal{S}$-terms we introduce $\mathcal{S}$-*equations*

$$\{\xrightarrow{\tilde{\pi}_1} q_1, \ldots, \xrightarrow{\tilde{\pi}_n} q_n\} = \{\xrightarrow{\tilde{\pi}'_1} r_1, \ldots, \xrightarrow{\tilde{\pi}'_m} r_m\}$$

which are pairs of $\mathcal{S}$-terms such that $\{q_1, \ldots, q_n\} = \{r_1, \ldots, r_m\}$ holds, i.e. the set of selected subterms (of the terms to be abstracted) are identical on both sides.

For example is

$$\{\xrightarrow{\langle \times_1 \rangle} X\} = \{\xrightarrow{\langle u_1, \times_1 \rangle} X, \xrightarrow{\langle u_2, \times_1 \rangle} X\}$$

an $\mathcal{S}$-equation while

$$\{\xrightarrow{\langle \times_1 \rangle} X, \xrightarrow{\langle \times_2, u_1 \rangle} Y\} = \{\xrightarrow{\langle u_1, \times_1 \rangle} X, \xrightarrow{\langle u_2, \times_1 \rangle} X\}$$

is not. The interpretation of an $\mathcal{S}$-equation $\mathcal{Q} = \mathcal{R}$ is defined as the set of equalities $q = r$ where q is part of the interpretation of $\mathcal{Q}$ and r part of the interpretation of $\mathcal{R}$.

$\mathcal{S}$-*substitutions* are finite mappings from variables to $\mathcal{S}$-terms. We extend the scope of an $\mathcal{S}$-substitution σ to a $\mathcal{S}$-term $\mathcal{T}$ by replacing each variable x of $\mathcal{T}$ in the domain of σ by $\sigma(x)$, but we have to take care to obtain an $\mathcal{S}$-term again.

To ease readability we present only the definition of $\mathcal{S}$-substitutions which change only one variable x, but the definition can be easily extended to the general case. Given an $\mathcal{S}$-substitution $\sigma = [x \leftarrow \{\xrightarrow{\tilde{\pi}'_1} s_1, ..., \xrightarrow{\tilde{\pi}'_n} s_n\}]$ and an $\mathcal{S}$-term $\mathcal{T} = \{\mathcal{T}_1, ..., \mathcal{T}_m\}$ with $\mathcal{T}_i = \xrightarrow{\tilde{\pi}_i} t_i$ then $\sigma(\mathcal{T}) = \bigcup_{1 \leq i \leq n} \sigma(\mathcal{T}_i)$ where

- $\sigma(\xrightarrow{\tilde{\pi}} x) = \{\xrightarrow{\tilde{\pi} \circ \tilde{\pi}'_j} s_j \mid 1 \leq j \leq n\}$,
- $\sigma(\xrightarrow{\tilde{\pi}} t) = \{\xrightarrow{\tilde{\pi}} t\}$ if x does not occur in t,
- $\sigma(\xrightarrow{\tilde{\pi}} t) = \{\xrightarrow{\tilde{\pi}} (t[x \leftarrow s_1])\}$ if $\sigma = [x \leftarrow \{\xrightarrow{\langle\rangle, s_1}\}]$ and
- otherwise undefined.

σ is *admissible* for $\mathcal{T}$ if each $\sigma(\mathcal{T}_i)$ is defined. An $\mathcal{S}$-term $\mathcal{Q}$ "*$\mathcal{S}$-matches*" an $\mathcal{S}$-term $\mathcal{T}$ if there is an admissible $\mathcal{S}$-substitution σ for $\mathcal{Q}$ such that $\sigma(\mathcal{Q}) = \mathcal{T}$.

In order to define the application of an $\mathcal{S}$-equation, we call an enriched occurrence $\tilde{\pi}$ *admissible* for an $\mathcal{S}$-term $\mathcal{T}$ if there is a $\xrightarrow{\tilde{\pi}'} t \in \mathcal{T}$ and there is an enriched occurrence $\tilde{\pi}''$ such that $\tilde{\pi} \circ \tilde{\pi}'' = \tilde{\pi}'$. The subterm $\mathcal{T}|\tilde{\pi}$ is defined by $\{\xrightarrow{\tilde{\pi}'} t \mid \xrightarrow{\tilde{\pi} \circ \tilde{\pi}'} t \in \mathcal{T}\}$.

Finally, we define $\mathcal{T}[\tilde{\pi} \leftarrow \mathcal{S}] = \{\xrightarrow{\tilde{\pi}'} t \mid \xrightarrow{\tilde{\pi}'} t \in \mathcal{T}$ and $\tilde{\pi}$ and $\tilde{\pi}'$are independent$\} \cup$ $\{\xrightarrow{\tilde{\pi} \circ \tilde{\pi}'} s \mid \xrightarrow{\tilde{\pi}'} s \in \mathcal{S}\}$ unless $\tilde{\pi}$ is admissible for $\mathcal{T}$.

We complete our definitions by the application of an $\mathcal{S}$-equation: An $\mathcal{S}$-equation $\mathcal{Q} = \mathcal{R}$ is *applicable* to an $\mathcal{S}$-term at an admissible enriched occurrence $\tilde{\pi}$ if and only if there is an admissible $\mathcal{S}$-substitution σ for $\mathcal{Q}$ and $\mathcal{R}$ such that $\sigma(\mathcal{Q}) = \mathcal{S}|\tilde{\pi}$. $\mathcal{S}[\tilde{\pi} \leftarrow \sigma(\mathcal{R})]$ is the result of the application.

We illustrate the usage of our $\mathcal{S}$-logic by the introductory example of section 2. The $\mathcal{S}$-equations are abstractions of the equations (4), (5) and (7), where (4) is abstracted with respect to the occurrences of Z, (5) with respect to the occurrences of Y and (7) with respect to the occurrences of y:

$$\{\xrightarrow{\langle \times_2, u_2 \rangle} Z\} = \{\xrightarrow{\langle u_2, \times_2 \rangle} Z\} \quad (13)$$

$$\{\xrightarrow{\langle \times_1, u_2 \rangle} Y\} = \{\xrightarrow{\langle u_2, \times_1 \rangle} Y\} \quad (14)$$

$$\{\xrightarrow{\langle u_2 \rangle} y\} = \{\xrightarrow{\langle\rangle} y\} \quad (15)$$

In Figure 1 on page 7 the first order proof of the theorem `GRP 175-1` and its corresponding abstract proof are presented. The arrow under the equation number in the abstract proof indicates in which direction the $\mathcal{S}$-equation has been applied.

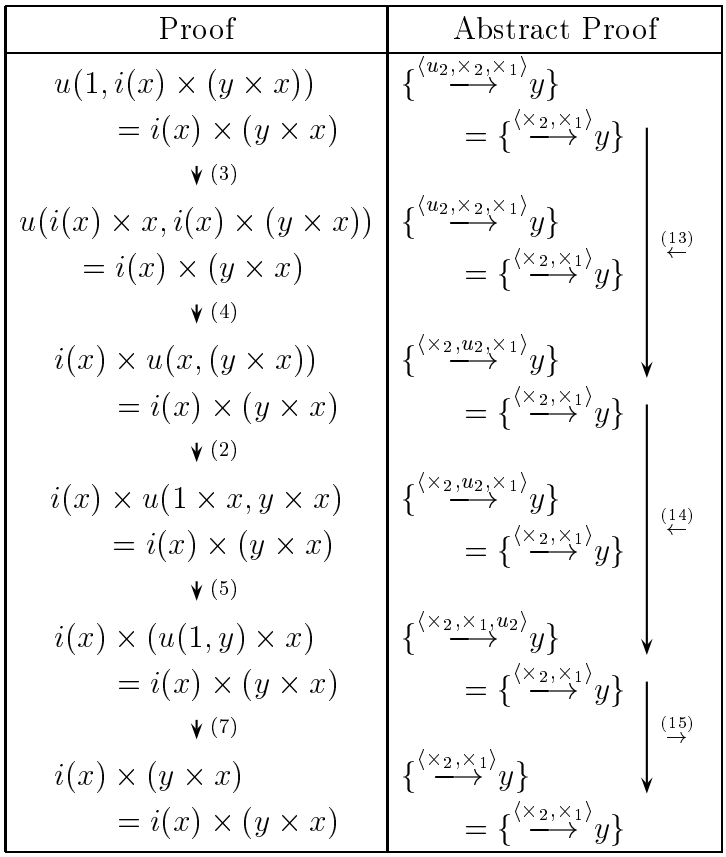

Figure 1: First order proof and abstract proof of theorem (6)

3.1 Refinements

Given a deduction in the abstract space of the $\mathcal{S}$-logic, we use it as a proof sketch in the ground space. Each deduction step in the abstract space corresponds to a sequel of deduction steps in the ground space. In order to obtain a first-order proof, we have to refine each abstract deduction step $\mathcal{S} \rightarrow_{\mathcal{Q}=\mathcal{R}} \mathcal{T}$ to a first order deduction. $s \rightarrow \ldots \rightarrow_{q=r} \rightarrow \ldots t$ where $\mathcal{S}$ is an $\mathcal{S}$-term of s and $\mathcal{T}$ is an $\mathcal{S}$-term of t. In general, each applied $\mathcal{S}$-equation $\mathcal{Q} = \mathcal{R}$ of an abstract deduction step corresponds to a set of possible first order equations. Hence, on the ground space we have to choose one of these equations which may involve backtracking in case we fail to enable the application of a chosen equation.

In our previous example some additional manipulations have to be performed on the term s to make an equation applicable. Namely, parts of s which are "hidden" in the abstract space, have to be manipulated on the ground space as it can be seen in Figure 1. As a first abstract deduction step the $\mathcal{S}$-equation (13) is applied. This step corresponds to two steps in the ground space. Here, the equa-

tions (3) and (4) are applied successively. While the application of equation (4) is suggested (since we used one of its abstractions in the abstract deduction step), the application of equation (3) is only performed as a subtask to enable the application of (4). This illustrates how the given proof sketch constrains the search in the ground space. Deduction steps in the ground space are divided into steps which immediately correspond to steps in the abstract space and preparation steps which enable the use of the selected equations.

4 Heuristics

Given appropriate abstractions for proof planning, we now define heuristics to guide the proof search in the abstract space. For this purpose consider our abstract proof in Figure 1. Inspecting all $\mathcal{S}$-terms occurring during the abstract deduction, we find a common structure $\langle \times_2, \times_1 \rangle$ in all of them. Since we are interested to minimize the differences of terms in the ground space we also would like to minimize the differences in the abstract space. In order to prevent the common structure from being modified, we adopt the notion of rippling (cf. [Hutter, 1990; Bundy *et al.*, 1990]) to enriched occurrences. Thus, each element of an enriched occurrence is annotated by a color-information specifying whether this element belongs to the skeleton or to the wave-front. Considering an enriched occurrence as a list, we obtain its skeleton by removing all elements belonging to the wave-front. Throughout our example we illustrate elements of the wave-front by shading them.

Given an equality problem we compute an abstracted equality problem $\mathcal{S} = \mathcal{T}$ and search for a common skeleton for the enriched occurrences of $\mathcal{S}$ and $\mathcal{T}$. For example, $\langle \times_2, \times_1 \rangle$ is the common *skeleton* of $\langle \times_2, u_2, \times_1 \rangle$ and $\langle \times_2, \times_1 \rangle$. u_2 is a wave-front of the first enriched occurrence. Similarly to the first-order case, there is no unique "maximal" skeleton of two enriched occurrences.

We illustrate the use of coloring of an enriched occurrence by the following example. Consider the first abstract equality problem of Figure 1, and there the two enriched occurrences of y. Using $\langle \times_2, \times_1 \rangle$ as a common skeleton and shading the wave-fronts results in the following colored abstract equality problem:

$$\{\overset{\langle u_2, \times_2, \times_1 \rangle}{\longrightarrow} y\} = \{\overset{\langle \times_2, \times_1 \rangle}{\longrightarrow} y\}$$

Using this color annotation we are able to represent the differences of two $\mathcal{S}$-terms such that we are able to predict how the application of an $\mathcal{S}$-equation changes the

wave-fronts. In order to apply a colored $\mathcal{S}$-equation $\mathcal{Q} = \mathcal{R}$ on a colored $\mathcal{S}$-term $\mathcal{S}$, the wave-fronts (respectively the skeleton) of $\mathcal{Q}$ have to match with the wave-fronts (respectively the skeleton) of $\mathcal{S}$. For example, consider the abstraction of axiom (4). The enriched occurrences of the subterm Z can be colored in the following manner:

$$\{\overset{\langle \times_2, u_2 \rangle}{\longrightarrow} Z\} = \{\overset{\langle u_2, \times_2 \rangle}{\longrightarrow} Z\} \tag{16}$$

If the above equation is applied from left to right on a colored $\mathcal{S}$-term $\mathcal{S}$, then we can predict that the *wave-front* belonging to the enriched occurrence of Z will be moved toward top level in $\mathcal{S}$, and the *skeletons* will remain unchanged. Similarly, an abstract equation

$$\{\overset{\langle u_2 \rangle}{\longrightarrow} y\} = \{\overset{\langle\rangle}{\longrightarrow} y\} \tag{17}$$

will remove the wave-front u_2 in the enriched occurrence of y.

Thus, we classify the $\mathcal{S}$-equations obtained by abstraction of the axioms according to their behavior in case of application. We search for a "maximal" common skeleton of the left- and right-hand sides, add the annotations to the $\mathcal{S}$-equations, and characterize them whether they will remove a wave-front, or move a wave-front up or inside. For example the colored $\mathcal{S}$-equation (16) is classified as a "moving up" $\mathcal{S}$-equation, if applied from left to right. The $\mathcal{S}$-equation (17) is characterized as a "removing" $\mathcal{S}$-equation.

Summing up, given an equality problem $s = t$ we compute an abstract equality problem $\mathcal{S} = \mathcal{T}$ and annotate the enriched occurrences of both to obtain a common skeleton. Then appropriate colored abstract equations are applied which will manipulate the wave-fronts until all differences are eliminated.

4.1 Introductory Example Revisited

We illustrate our technique with the help of our introductory example. The abstractions (13), (14) and (15) of the axioms (4), (5) and (7) have already been presented in the previous section.

They will be used for the abstract proof of theorem (6). Annotating them with colors results in the following colored abstract equations: The first colored $\mathcal{S}$-equation results from (13)

$$\{\overset{\langle \times_2, u_2 \rangle}{\longrightarrow} Z\} = \{\overset{\langle u_2, \times_2 \rangle}{\longrightarrow} Z\}, \tag{18}$$

and is characterized as moving a wave-front u_2 "inside" if applied from right to left. The second $\mathcal{S}$-equation

$$\{\overset{\langle \times_1, u_2 \rangle}{\longrightarrow} Y\} = \{\overset{\langle u_2, \times_1 \rangle}{\longrightarrow} Y\}, \tag{19}$$

is obtained from (14) and is also characterized as moving a wave-front u_2 "inside" if applied from right to left. The last colored $\mathcal{S}$-equation

$$\{\overset{\langle u_2 \rangle}{\longrightarrow} y\} = \{\overset{\langle\rangle}{\longrightarrow} y\}, \tag{20}$$

is obtained from the condition (15) of the theorem. This one removes a wave-front u_2 if applied from left to right. The colored abstracted theorem is

$$\{\overset{\langle u_2, \times_2, \times_1 \rangle}{\longrightarrow} y\} = \{\overset{\langle \times_2, \times_1 \rangle}{\longrightarrow} y\} \tag{21}$$

According to the presented heuristics, the wave-front occurring in the enriched occurrence of y on the left-hand side is moved inside by using the colored $\mathcal{S}$-equation (18) from right to left:

$$\{\overset{\langle \times_2, u_2, \times_1 \rangle}{\longrightarrow} y\} = \{\overset{\langle \times_2, \times_1 \rangle}{\longrightarrow} y\} \tag{22}$$

In a next step this wave-front is moved further inside by applying the colored $\mathcal{S}$-equation (19) from right to left on the left-hand side of the theorem which yields

$$\{\overset{\langle \times_2, \times_1, u_2 \rangle}{\longrightarrow} y\} = \{\overset{\langle \times_2, \times_1 \rangle}{\longrightarrow} y\} \tag{23}$$

Finally, the wave-front is removed using the colored $\mathcal{S}$-equation (20) which results in the trivial problem

$$\{\overset{\langle \times_2, \times_1 \rangle}{\longrightarrow} y\} = \{\overset{\langle \times_2, \times_1 \rangle}{\longrightarrow} y\}, \tag{24}$$

The abstract deduction solving the abstract equality problem (6) is just the one we used in our informal approach in section 2, which is what we were looking for. However, note that there are some choice points in the abstract deduction above, leading also to an abstract solution and thus to possible abstract plans. Altogether there are five possible abstract deductions according to the heuristic and the color restrictions, all solving the abstract equality problem. Thus, there are five proof plans, but only one of this proof plans is executable, the one above.

5 Implementation

The presented abstraction as well as the heuristics have been implemented in the InKa-system (cf. [Biundo *et al.*, 1986; Hutter and Sengler, 1996]). For that, the

$\mathcal{S}$-terms, $\mathcal{S}$-equations and the $\mathcal{S}$-substitution defined in the $\mathcal{S}$-logic have been implemented. In each $\mathcal{S}$-equation a list of the first order equations is stored of which it is a legal abstraction. Then, this information is used to plan the refinement from the abstract space to the ground space. The plans are presently totally ordered, but attempts are made in order to extend it to partially ordered plans. A plan is simply represented as a list of directed $\mathcal{S}$-equations and occurrences at which they have been applied.

The presented abstraction and heuristic has been successfully tested on several examples together with other abstractions and heuristics as it has been described in the introduction. The examples were taken from several domains other than group theory and performed very well (cf. [Autexier, 1997] for various examples). E.g., in the example above both the planning and its refinement took less than a second on a SPARC 20.

6 Comparisons

In the history of AI research a wide variety of abstractions have been proposed. Further, a theory of abstraction has been developed by Giunchiglia and Walsh (cf. [Giunchiglia and Walsh, 1992]), which led to the development of ABSFOL (cf. [Giunchiglia and Villafiorita, 1996]). However, our abstraction can not be encoded in ABSFOL, since the language to describe abstractions is not powerful enough. Indeed, it is not possible to define parameterized abstractions, and our abstraction is parameterized by occurrences of subterms.

Among all kinds of abstractions, there are especially two abstractions developed for proof planning, namely *gazing*, and an extension of it to deal with functions. The idea of *Gazing* (cf. [Plummer, 1987]) is roughly speaking to map first order formulas onto propositional formulas and then to use propositional decision procedures to find a plan. Therefore, *gazing* is not comparable to our technique, since we are only dealing with equality problems. An extension of *gazing* is to map literals into a set of its predicate *and* function symbols. This approach results in inconsistent abstraction spaces which prohibits a complete proof search in the abstract space. Adding more information in the abstraction — as it is done in the extended *gazing* — hampers a powerful proof planning. Therefore our abstraction is more adequate for the purposes of equality proof planning, especially because of its flexibility. However, this additional flexibility leads to a larger branching factor in the plan search space. Thus, some powerful constraints, like coloring, have been developed to compensate this effect. Nevertheless, the

additional flexibility in the abstraction leads to planning techniques dealing much better with equality problems than the extended *gazing* technique.

As mentioned above, our heuristics are strongly related to the rippling techniques [Hutter, 1990; Bundy *et al.*, 1990]. Essentially we use a kind of rippling on strings to guide the search process in the abstract space, but unlike the original rippling approach we are able to abstract terms from unimportant argument positions.

7 Conclusion

We presented parameterized abstractions of terms which are used to compute proof sketches in the setting of hierarchical proof planning. Besides the heuristics given in section 4 we developed other techniques to equalize enriched occurrences with the help of $\mathcal{S}$-equations. These heuristics make use of the fact that enriched occurrences are basically strings and search algorithms based on strings can be used (cf. [Autexier, 1997]).

Although $\mathcal{S}$-deductions are only defined in an equational setting, the idea can be lifted to general first-order formulas. Then our approach can be used to equalize specific subformulas of a theorem in order to enable e.g. a resolution step.

Classical theorem provers may have a better performance on some problems, but the main advantage of our planning approach is, that we can allow user interaction on a strategic level (i.e. on the level of the abstractions). This is essential when dealing with proof obligations occuring in the verification of realistic software components. Furthermore, the hierarchical proof planning procedure supports a good proof presentation, which is rather difficult with classical theorem provers. Actually, the abstract planning steps provide a simple mechanism in order to divide a proof into different parts, which can be explained independently.

References

[Autexier, 1997] Serge Autexier. The Logic of an Abstraction: The $\mathcal{S}$-Logic. MSc Thesis, Technical Report, Department of Computer Science, University of Saarland, 1997.

[Biundo *et al.*, 1986] S. Biundo, B. Hummel, D. Hutter, and C. Walther. The Karlsruhe Induction Theorem Proving System, In *Proceedings of the 8th International Conference on Automated Deduction (CADE).*

[Bundy *et al.*, 1990] Alan Bundy, Frank van Harmelen, Alan Smaill, and Andrew Ireland. Extension to the rippling-out tactic for guiding inductive proofs, In *Proceedings 10th International Conference on Automated Deduction (CADE)*, Springer, LNAI 449, 1990.

[Giunchiglia and Villafiorita, 1996] Fausto Giunchiglia and Adolfo Villafiorita. ABSFOL: A proof checker with abstraction, In *Proceedings of the 13th International Conference on Automated Deduction (CADE)*, Springer, LNAI 1104, 1996.

[Giunchiglia and Walsh, 1992] Fausto Giunchiglia and Toby Walsh. A Theory of Abstraction. *Journal of Artificial Intelligence.*

[Hutter and Sengler, 1996] Dieter Hutter and Claus Sengler. INKA - The Next Generation, In *Proceedings of the 13th International Conference on Automated Deduction (CADE).*

[Hutter, 1990] Dieter Hutter. *Guiding Induction Proofs.* In *Proceedings 10th International Conference on Automated Deduction (CADE)*, Springer, LNAI 449, 1990.

[Hutter, 1996] Dieter Hutter. *Using Rippling for Equational Reasoning.* In *Proceedings 20th German Annual Conference on Artificial Intelligence KI-96*, Ed. S. Hölldobler, Springer, LNAI, Dresden, Germany, 1996.

[McRobbie and Slaney, 1996] *Proceedings of the 13th International Conference on Automated Deduction (CADE)*, Springer, LNAI 1104, 1996.

[Plummer, 1987] D. Plummer. *Gazing: Controlling the Use of Rewrite Rules.* PhD thesis, Department of Artificial Intelligence, University of Edinburgh.

[Sutcliff and Suttner, 1996] Geoff Sutcliff and Christian B. Suttner. The TPTP Problem Library.

Parameterized Abstractions used for Proof-Planning

Serge Autexier, Dieter Hutter

RR 97-04

Research Repor