Equational Proof-Planning by Dynamic Abstraction

Serge Autexier*
CS Department
University of Saarland

Dieter Hutter†
German Research Centre
for Artificial Intelligence

Abstract

We present a new abstraction designed for the purpose of equational proof-planning. The abstraction maps a given equation into some partial representation of its structure wrt. some given subterms common to both sides of the equation. Thus, the abstraction of an equation is dynamic, since it dependents on the selected common subterms. Furthermore we sketch a heuristics based on the difference reduction paradigm to ease the abstract proof plan construction.

1 Introduction

Proof planning allows to construct proofs in a hierarchical manner by decomposition of the given goals in a sequel of subgoals. Our purpose is to use abstractions to obtain the decomposition. According to [4] 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, but none had a real success in equational proof-planning. 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.

In the following we restrict ourselves on equality problems and present an abstraction and heuristics to guide the abstract proof search. The abstraction of an equation is parameterized by some common subterms of both sides of the equation and the attention of the prover is focused to the sequel of function symbols governing the occurrences of these subterms.

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. [11]):

$$\forall x, y.\ u(1, y) = y \rightarrow u(1, i(x) \times (y \times x)) = i(x) \times (y \times x) \tag{1}$$

*Fachbereich 14, Informatik, Universität des Saarlandes, Postfach 15 11 50, 66041 Saarbrücken, Germany, autexier@cs.uni-sb.de

†DFKI GmbH, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany, hutter@dfki.de

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

$$\forall X.\ 1 \times X = X \quad (2) \qquad \forall X, Y, Z.\ X \times u(Y, Z) = u(X \times Y, X \times Z) \quad (4)$$

$$\forall X.\ i(X) \times X = 1 \quad (3) \qquad \forall X, Y, Z.\ u(X, Y) \times Z = u(X \times Z, Y \times Z) \quad (5)$$

Proving the given theorem results in proving the equality

$$u(1, i(x) \times (y \times x)) = i(x) \times (y \times x) \quad (6) \qquad \text{assuming } u(1, y) = y \quad (7)$$

Following the paradigm of difference reduction we first compare the function symbols occurring on both sides of 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. More formally, 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 the left term of (6) y occurs in position $\langle 2, 2, 1\rangle$ and is governed by a sequel of functions $u, \times, \times$. Thus, the idea of the abstraction is to use a representation combining both the position of a term and the sequel of functions down to it: thus, we describe the structure of the left term wrt. the occurrence of y by $\langle u_2, \times_2, \times_1\rangle_y$. Since we are interested in moving u "down to" y, we inspect our database to find equations "moving down" functions u. For this purpose we proceed the same way for the axioms as we did for the theorem. For example we consider the occurrences of Z in equation (4): the structures of the left and right hand sides of (4) wrt. Z are represented by $\langle \times_2, u_2\rangle_Z$ and $\langle u_2, \times_2\rangle_Z$. Thus, if we could apply (4) from right to left on the left hand side of the theorem, we could "move" u towards y. Analogously, the next step suggests the application of (5) and finally the application of the condition (7) to remove the function u. Thus, the suggested proof plan found when using this abstraction is to enable the applications of (4), (5) and (7) (in this order, cf. Figure 1). The refinement of this proof plan succeeds in the proof viewed in Figure 1. Thereby the applications of (3) and (2) are only performed to enable the applications of (4) and (5) respectively, as suggested by the proof plan.

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. Thereby did the occurrences of y denote the invariant parts or *skeleton* of our example. In this section we will formalize this idea into a notion of $\mathcal{S}$-terms that are specific structural 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 $\langle 1, 1\rangle$. Each enriched occurrence $\tilde{\pi}$ of t denotes a subterm $t|\tilde{\pi}$. Thus, $\tilde{\pi}_{t|\tilde{\pi}}$ 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 $\mathcal{T} = \{\tilde{\pi}^1_{t|\tilde{\pi}^1}, \ldots, \tilde{\pi}^n_{t|\tilde{\pi}^n}\}$ is called an $\mathcal{S}$-*term* if all $\tilde{\pi}^i$ denote independent positions. A $\mathcal{S}$-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 a $\mathcal{S}$-term $\mathcal{T} = \{\tilde{\pi}^1_{u_1}, \ldots, \tilde{\pi}^n_{u_n}\}$ is the set of terms t for which $\mathcal{T}$ is a legal abstraction, i.e., $\tilde{\pi}^i$ are enriched occurrences of t

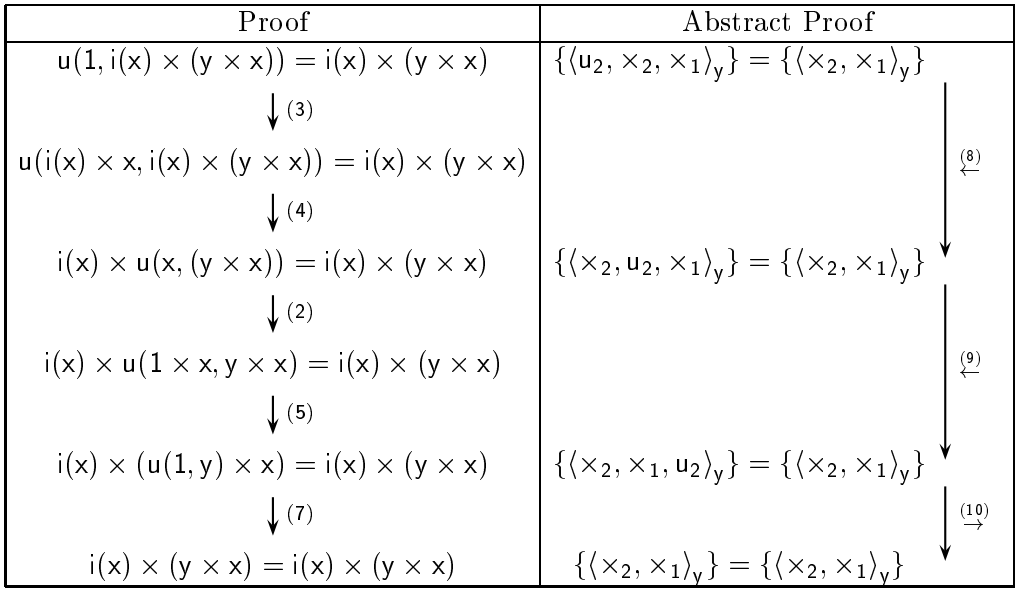

Figure 1: First-order proof and abstract proof of theorem (6)

and $t|\tilde{\pi}^i = u_i$. The empty set is a $\mathcal{S}$-term which denotes *all* terms while $\{\langle\rangle_t\}$ characterizes exactly t.

In order to manipulate $\mathcal{S}$-terms we introduce $\mathcal{S}$-*equations* $\{\tilde{\pi}^1_{q_1}, \ldots, \tilde{\pi}^n_{q_n}\} = \{\tilde{\pi}'^1_{r_1}, \ldots, \tilde{\pi}'^m_{r_m}\}$ which are equations over $\mathcal{S}$-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 $\{\langle \times_1 \rangle_X\} = \{\langle u_1, \times_1 \rangle_X, \langle u_2, \times_1 \rangle_X\}$ is an $\mathcal{S}$-equation while $\{\langle \times_1 \rangle_X, \langle \times_2, u_1 \rangle_Y\} = \{\langle u_1, \times_1 \rangle_X, \langle u_2, \times_1 \rangle_X\}$ is not.

In order to formalize the abstract deduction process (the so-called $\mathcal{S}$-deduction), we introduce a notion of substitution, namely a $\mathcal{S}$-*substitution* and define the application of a $\mathcal{S}$-equation onto a $\mathcal{S}$-term (see [1] for the details).

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

$$\{\langle \times_2, u_2 \rangle_Z\} = \{\langle u_2, \times_2 \rangle_Z\} \quad (8) \qquad \{\langle u_2 \rangle_y\} = \{\langle\rangle_y\} \quad (10)$$
$$\{\langle \times_1, u_2 \rangle_Y\} = \{\langle u_2, \times_1 \rangle_Y\} \quad (9)$$

In Figure 1 the first-order proof of theorem `GRP175-1` 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.

3.1 Refinements

Given a deduction in the abstract space of the $\mathcal{S}$-abstraction, we use it as a proof sketch in the ground space. Indeed does an abstract proof provide a set of partially ordered proof-plans for the original first-order logic theorem. Thereby does each deduction step in the abstract space correspond 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 a $\mathcal{S}$-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.

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. In order to prevent the common structure from being modified we adapt the notion of Rippling (cf. [5, 2]) to enriched occurrences. Thus, each element of an enriched occurrence is annotated by a colour-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 which belong 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 $\langle \times_2, \mathsf{u}_2, \times_1 \rangle$ 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 colouring 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 coloured abstract equality problem: $\{\langle \mathsf{u}_2, \times_2, \times_1 \rangle_{\mathsf{y}}\} = \{\langle \times_2, \times_1 \rangle_{\mathsf{y}}\}$. Using this colour 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 coloured $\mathcal{S}$-equation $\mathcal{Q} = \mathcal{R}$ on a coloured $\mathcal{S}$-term $\mathcal{S}$, the wave-fronts resp. the skeleton of $\mathcal{Q}$ have to match with the wave-fronts resp. the skeleton of $\mathcal{S}$. For example, consider the abstraction of axiom (4). The enriched occurrences of the subterm Z can be coloured in the following manner: $\{\langle \mathsf{u}_2, \times_2 \rangle_{\mathsf{Z}}\} = \{\langle \times_2, \mathsf{u}_2 \rangle_{\mathsf{Z}}\}$. If the above equation is applied from left to right on a coloured $\mathcal{S}$-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 $\{\langle \mathsf{u}_2 \rangle_{\mathsf{y}}\} = \{\langle\rangle_{\mathsf{y}}\}$ 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 behaviour 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 is the first coloured $\mathcal{S}$-equation classified as a "moving up" $\mathcal{S}$-equation and the second as a "removing" $\mathcal{S}$-equation.

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

The presented abstraction as well as the heuristics have been implemented in the InKa-system (cf. [6]) and successfully tested on several examples. E.g., does the plan search of

the above example and its refinement take less than a second on a SPARC 20. Thereby theoretical properties (cf. [1] for more details) allowed an efficient implementation of the $\mathcal{S}$-abstraction.

In the above example the proof plan search is a simple task, when using the presented Rippling-like heuristic; but even in more challenging examples simple proof plan search heuristics result in good hints about how the proof can be performed. E.g., this is the case for theorem `BOOOO2` from the TPTP-library. A simple exhaustive proof plan search on the abstraction of the theorem results in a five-steps abstract proof plan; while refining these steps, some complex problems occur, but these are easy to solve by using difference reduction techniques based on classical Rippling over first-order terms.

5 Related Works

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. [4]), which led to the development of ABSFOL (cf. [3]). However, it is difficult to encode our abstraction in ABSFOL, since the language to describe abstractions does not allow to define directly parameterized abstractions.

Among all kinds of abstractions, there are especially two abstractions comparable to our abstraction, namely *Gazing*, and *Grazing* an extension of it, although the first has not been developed for the purpose of proof-planning. The idea of *Gazing* (cf. [8]) is to map in a first step first-order formulas onto propositional formulas and thus is not useful for equational proof-planning. In a second step *Gazing* additionally takes all the occurring function symbols into account together with some polarity information. Since the relationship between the occurring function symbols is omitted, the proof search is rather unconstrained. To fix these problems, *Grazing* (cf. [9]) has been developed, which preserves the term-structure by abstracting only bounded variables. This, however, preserves too much structure and thus hampers a powerful equational 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 colouring, 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 *gazing* technique.

6 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. [1]).

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 stra-

tegic level (i.e. on the level of the abstractions), which is essential when dealing e.g. with proof obligations occurring in the verification of realistic software components. Furthermore, the hierarchical proof planning procedure supports a good proof presentation, which is rather difficult with saturation based 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. Another advantage of the described heuristics is that, like the rippling heuristics for first-order logic proofs, the proof-plan search is more or less independent of the number of axioms in a given database, which is a not the case for saturation based theorem provers.

References

[1] S. Autexier. An Abstraction for Proof-Planning: The $\mathcal{S}$-Abstraction. SEKI Report SR-97-05, Universität des Saarlandes, Fachbereich Informatik, June 1997.

[2] A. Bundy, F. van Harmelen, A. Smaill, and A. Ireland. Extension to the rippling-out tactic for guiding inductive proofs. In Stickel [10], pages 132–146.

[3] F. Giunchiglia and A. Villafiorita. ABSFOL: A proof checker with abstraction. In McRobbie and Slaney [7], pages 136–140.

[4] F. Giunchiglia and T. Walsh. A Theory of Abstraction. *Journal of Artificial Intelligence*, 56(2-3):323–390, 1992.

[5] D. Hutter. Guiding Induction Proofs. In Stickel [10].

[6] D. Hutter and C. Sengler. INKA - The Next Generation. In McRobbie/Slaney [7].

[7] M. A. McRobbie and J. K. Slaney, editors. *Proceedings of the 13$^{\text{th}}$ International Conference on Automated Deduction (CADE)*, volume 1104 of *LNCS* , 1996, Springer.

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

[9] A. Simpson. *Grazing: A stand alone Tactic for Theoretical Inference.* MSc Thesis, Department of Artificial Intelligence, University of Edinburgh, 1988.

[10] M. E. Stickel, editor. *Proceedings 10$^{\text{th}}$ International Conference on Automated Deduction (CADE)*, volume 449 of *LNAI* . Springer, July 1990.

[11] G. Sutcliff and Ch. B. Suttner. The TPTP Problem Library. AR-96-02, Institut für Informatik, TU-München, June 1996.