A New Calculus for Semantic Matching*

Bernd Bütow, Robert Giegerich, Enno Ohlebusch, Stephan Thesing

Technische Fakultät der Universität Bielefeld
Postfach 100131, 33501 Bielefeld, Germany
{berndb|robert|enno|isthesin}@techfak.uni-bielefeld.de

Abstract. In this paper, we present Reverse Restructuring, a new calculus for solving the semantic matching problem. For narrowing, advanced selection rules are commonly seen as an appropriate method to reduce the search space. Our approach to design a special calculus for special goals is another way of reducing the efficiency defects of narrowing. Reverse Restructuring constructs derivations in the reverse direction by guessing terms from which an already known term might be derived. To this end, the rules of the underlying term rewriting system are also applied in the reverse direction, i.e. from right to left. We show the soundness and completeness of this calculus and demonstrate its efficiency for an important class of problems.

1 Introduction

Narrowing is commonly seen as the basis for the amalgamation of logic and functional programming. Narrowing today refers to a whole family of procedures for solving equational goals $s =_R t$ in the equational theory defined by a term rewriting system (TRS) $R$. The development of different narrowing strategies is driven by the need to improve efficiency by cutting down the search space. Most approaches require certain restrictions on the TRS in order to ensure completeness. A certain endpoint in this line of development is currently marked by the needed narrowing strategy of Antoy et al. [AEH94], where a relatively complex selection rule ensures that no "unnecessary" narrowing step is done. The price paid by this strategy is a severe restriction on the underlying TRSs.

Another line of attacking the efficiency defects of narrowing is the idea to design special calculi for special goals. Of course, this idea makes only sense if the subgoals arising in a computation have the same restricted form. Such an approach was first suggested by Dershowitz et al. [DMS92], introducing the problem of semantic matching: Given an arbitrary term $s$ and a term $N$ in ground normal form, find a substitution $\sigma$ such that $s\sigma =_R N$.

At first glance, this does not seem to be a problem worth individual study. It is a common technical device to transform general narrowing goals $s =_R t$ into the form $eq(s,t) =_R true$ by means of the extra rule $eq(X,X) \rightarrow true$.

* This work is supported by the DFG-project: "Abstrakte Inferenzmaschine" under Az. Gi 178/1-2
In this sense, all semantic unification problems can be turned into semantic matching problems, and we cannot expect a semantic matching calculus to offer any advantages.

However, the situation changes if we restrict the underlying TRS $\mathcal{R}$ to left-linear or variable-preserving rules. Note that the rule $eq(X, X) \rightarrow true$ violates either restriction. In this setting, Dershowitz et al. [DMS92] gave a calculus for semantic matching which is sound and complete.

The class of problems which can be solved by semantic matching is still quite interesting. In particular, function inversion belongs to this class, i.e. goals of the form $f(X) \rightleftharpoons N$. It has been observed in [Gie90] that compiler code selection can be seen as solving an equation $\delta(X) \rightleftharpoons ilprog$, where $ilprog$ is an intermediate language program, and $\delta$ is a derivor mapping target machine programs into intermediate language programs. (We shall elaborate this example in section 4.2.) Current compiler technology [ESL89, Gie90, FHP92, Emm94] solves this problem with an efficiency which narrowing procedures can only dream of.

The overall goal of our work is to improve the efficiency of semantic matching while at the same time improving the flexibility of code selection mechanisms. The present paper is a first step in this direction. Our basic idea is as follows: In solving $f(X) \rightleftharpoons N$, we try to exploit the fact that $N$ is in ground normal form. Rather than guessing instantiations of $X$ that successively construct a rewrite derivation $f(X\sigma) \rightarrow^{*}_{\mathcal{R}} N$, we construct this derivation in the reverse direction by guessing terms from which $N$ can be derived. In this paper, we introduce the Reverse Restructuring calculus, show its soundness and completeness for the semantic matching problem, and outline some optimizations. Furthermore, we will provide the Reverse Restructuring computation with some knowledge about the forward computation paths resulting in a further considerable reduction of the search space. More details and full proofs can be found in [BT94].

2 Preliminaries

2.1 Notations

In this section we briefly recall basic notions of term rewriting (see for example [DJ90, Klo92] for more details).

A signature is a set $\Sigma$ of operators. Every $f \in \Sigma$ is combined with an arity $n$, $n \geq 0$. Let $X, X \cap \Sigma = \emptyset$, be a countable infinite set of variables. We define the set of terms $T(\Sigma, X)$ with variables as the smallest set containing $X$ such that $f(t_1, \ldots, t_n) \in T(\Sigma, X)$ if $f$ has arity $n$ and $t_1, \ldots, t_n \in T(\Sigma, X)$. By $\mathcal{V}(t)$, we denote the set of variables occurring in the term $t$. If $\mathcal{V}(t) = \emptyset$, $t$ is said to be ground. An occurrence $p$ in a term $t$ is defined as a sequence of natural numbers identifying a subterm in $t$. The empty sequence, denoted by $\epsilon$, identifies for every term $t$ the whole term itself. For every term $f(t_1, \ldots, t_n)$, the sequence $i.p$ where $i \leq n$ and $p$ is an occurrence in $t_i$, identifies the subterm of $t_i$ at occurrence $p$. $t/p$ denotes the subterm of $t$ at occurrence $p$; $t[p \leftarrow t']$ denotes the term obtained from $t$ by replacing $t/p$ by $t'$. We define the prefix ordering