

Termination and Decidability Results

for

String Unification

Mike Livesey

Jörg Siekmann

August 1975

University of Essex

Abstract

This paper investigates the unification problem for strings, i.e. for 'in-built' associativity.

The first section deals with substitutions and their normal form. The second section presents an improvement to the basic string-unification algorithm and the third section establishes classes of strings for which termination or decidability results can be derived.

Introduction

For almost as long as attempts at proving theorems by machines have existed, it has been well known [e.g. (2)], that certain axioms if left without precautions in the database of an automatic theorem proving (ATP) system will force the ATP to go astray. So far, we have concentrated on the following axioms:

Associativity	(A)
Commutativity	(C)
Idempotence	(I)

and on their respective combinations (with and without an identity element). We have unification algorithms, which have these axioms (and combinations of these axioms) "built-in" [11].

The pure associativity is the theoretically hardest case. A (semi)-algorithm for it was presented in [18] and although this paper is virtually selfcontained, [18] may provide additional motivation and a more explicit discussion in terms of ATP-systems.

Unification algorithms are however fundamentally more important and significant beyond ATP: each of the above axioms and each of their combinations defines a particular data-structure; e.g.:

A	:	strings
A+C	:	bags
A+C+I	:	sets etc.

and certain AI-languages (like QA4 [14], PLANNER [8] etc.) have matching algorithms for these cases built into their deductive machinery.

Apart from the fact that these matching algorithms have without exception been designed ad hoc, i.e. without respect to completeness, minimality [see (18) for definitions] or sometimes even correctness; the basic question of whether a particular matching problem for a particular data structure is decidable (i.e. of whether a matching algorithm exists) has not been answered.

A little reflection will show that for very rich matching structures, as e.g. MATCHLESS in PLANNER [8], the matching problem is undecidable. This presents a problem for the designer of such languages: on the one hand, very rich and expressive matching structures are desirable, since they form the basis for the invocation- and deduction mechanism. On the other hand, drastic restrictions will be necessary if matching algorithms are to be found. The question is: just how severe do these restrictions have to be. To answer this question, more precise definitions of matching structures than are customary up to now, will be required.

In this paper, the structures under consideration are strings, composed of constants and variables. A more detailed investigation into 'languages' of this kind, from the formal language theory point of view, will be presented in [12].

Then the next step may be to consider basic functions over strings, like e.g. REVERSE, POWER, KLEENE-STAR etc. and to

- a) find matching algorithms for these cases
- b) find a basic, minimal set of such functions.

However it is possible to show that already for slightly more general 'string'-languages no general matching algorithm exists and thus the problem becomes to find suitable subclasses of such languages.

Apart from these two 'practical' motivations (i.e. ATP and matching algorithms for PLANNER-type languages), there is a - much older - field in which 'matching' problems arise. It can easily be shown that the problem of unification with in-built associativity (the matching problem for strings) is equivalent to solving simultaneous equations over a free semigroup. This problem, i.e. the decision problem of whether a set of word equations have a solution apparently originated with Markov⁽ⁱ⁾ and has been investigated by inter alia Hmelevskij [5,6,7] and [19].

(i) and is called Markov's problem in Eastern Europe and Löb's problem in America.

Briefly the problem is; given word equations:

$$\psi_{11} = \psi_{12}$$

$$\psi_{21} = \psi_{22}$$

.

.

.

$$\psi_{n1} = \psi_{n2}$$

with $\psi_{ik} \in \{V \cup C\}^*$, where V is an alphabet (of variables) and C an alphabet (of constants), do there exist substitutions of expressions of C^* into the variables of V such that

$$\psi_{i1} \equiv \psi_{i2}, \quad 1 < i < n; \quad \text{i.e. such that}$$

the two sides of the equations become identical.

No general solution to this problem is known; and it has been an open question now for over 25 years. A more exact formulation of the problem and solutions to some special cases can be found in [5].

This paper is divided into three sections. The first section gives basic definitions and terminology. We have spent some time however on the substitution part, to find definitions which emphasise the functional character of substitutions. The usual definition of a substitution as 'a set of pairs' confuses the old issue of "the name of a function" and the function itself and has led to - in our view unnecessary - complications in what is meant by the application of a substitution.

The second section gives an improvement of the basic algorithm, which if speed should ever be of any concern, could bring a considerable advantage. More importantly however: the improved algorithm can solve (i.e. terminate) cases, which the original one can not.

The last section shows various ways of classifying strings and gives termination or decidability results for such classes.

Das Volumen dieser Arbeit (40 Seiten) schließt einen vollständigen Abdruck im hier gegebenen Rahmen leider aus. Interessenten wenden sich bitte an die Autoren.