A Transformation System for Definite Programs Based on Termination Analysis

J. Cook and J.P. Gallagher*

Department of Computer Science, University of Bristol, Bristol BS8 1TR, U.K.

Abstract. We present a goal replacement rule whose main applicability condition is based on termination properties of the resulting transformed program. The goal replacement rule together with a multi-step unfolding rule forms a powerful and elegant transformation system for definite programs. It also sheds new light on the relationship between folding and goal replacement, and between different folding rules. Our explicit termination condition contrasts with other transformation systems in the literature, which contain conditions on folding and goal replacement, often rather complex, in order to avoid "introducing a loop" into a program. We prove that the goal replacement rule preserves the success set of a definite program. We define an extended version of goal replacement that also preserves the finite failure set. A powerful folding rule can be constructed as a special case of goal replacement, allowing folding with recursive rules, with no distinction between old and new predicates. A proof that Seki's transformation system preserves recurrence, an important termination property, is outlined.

1 Introduction

In this paper we define a goal replacement rule (for definite programs) based on termination analysis. Under the conditions of the rule, a replacement can only be made if the resulting program terminates for all ground calls to the head of the transformed clause. We show that application of this rule to a program will preserve its success set. The finite failure set can also be preserved by adding a similar termination condition on the program being transformed. We also give a definition of a more general unfolding rule which is based on a partial evaluation rule discussed in [LS91] and [GS91].

On reviewing the literature regarding folding and goal replacement, it was apparent that the underlying issue (when considering the correctness of such rules) was the termination of the resulting program. This was particularly evident in [Sek89] and [BCE92].

In the first of these papers, Seki, discussing the unfold/fold transformations of Tamaki and Sato [TS84], noted that a goal may not terminate in a program resulting from a Tamaki-Sato folding rule, even though it finitely failed in the original program. To combat this, Seki proposed a reformulated folding rule.

* Correspondence to jpg@compae.bristol.ac.uk
Under the conditions of [BCE92], a goal replacement can be made provided that 'a loop is not introduced' into the program. This is clearly a reference to some termination properties of the transformed program.

It should be noted that the original unfold/fold transformations for functional programs [BD77] preserves only partial correctness. Total correctness, that is, termination, has to be established separately. In logic programming, starting with the work of Tamaki and Sato [TS84], the tradition has been to devise conditions on unfolding and folding that implicitly guarantee total correctness results. Recently, conditions on goal replacement requiring 'reversibility' of goal replacement have been proposed by Proietti and Pettorossi [PP93], stressing the preservation of partial and total correctness as separate concerns. This work is in the same spirit as ours though the approach differs. Our goal replacement is not reversible, but still preserves total correctness (in the sense of the program's success set).

The link between transformation rules and termination analysis provided us with the motivation for the system we present. The main difference of approach between our work and the works just cited (and also [GS91], [Mah87], [Sek89] and others) is that in our goal replacement rule termination of the transformed program is a condition of application of the rule, and we leave open the means by which termination is checked. Although the problem of checking goal replacement conditions is thereby shifted to checking termination properties, the advantage gained is that the link between folding and goal replacement is clarified. More flexible folding rules can be invented, as we show below. Also, research on termination analysis can be brought to bear on the problem.

We note that Boulanger and Bruynooghe [BB93] also developed an approach to goal replacement that is based on generation of replacement lemmata during an abstract interpretation of the program. This approach also seems capable of performing fold-like operations that are less restrictive than usual, allowing folding with recursive rules, for instance.

There are two alternative strategies for implementing our transformation rules. One possibility is to use termination analysis techniques, to check termination of a transformed program directly. Secondly, special cases of the goal replacement can be derived, in which syntactic conditions ensuring termination are checked. This suggest a reconstruction of 'fold' transformations. In general the first approach seem the more promising, since it seems very difficult to find syntactic conditions that guarantee useful termination properties (such as acceptability [AP90]).

Section 2 contains a review of logic program transformation systems. In Section 3 we introduce our transformation rules and prove the correctness results. In Section 4 termination analysis is reviewed. An example of our system is given in Section 5. We show that our replacement rule allows 'foldings' to be based on

---

2 We are grateful to A. Pettorossi for this pertinent comment

3 We have been informed that folding using recursive rules was also considered in unpublished technical reports by Tamaki and Sato (1986), and by Kanamori and Fujita (1986).