Abstract. A rewrite sequence is said to be outermost-fair if every outermost redex occurrence is eventually eliminated. O'Donnell has shown that outermost-fair rewriting is normalising for almost orthogonal first-order term rewriting systems. In this paper we extend this result to the higher-order case.

1 Introduction

It may occur that a term can be rewritten to normal form but is also the starting point of an infinite rewrite sequence. In that case it is important to know how to rewrite the term such that eventually a normal form is obtained. The question of how to rewrite a term can be answered by a strategy, which selects one or more redex occurrences in every term that is not in normal form. If repeatedly contracting the redex occurrences that are selected by the strategy yields a normal form whenever the initial term has one, the strategy is said to be normalising.

A classical result for λ-calculus with β-reduction is that the strategy selecting the leftmost redex occurrence is normalising. This is proved in [CFC58]. For orthogonal first-order term rewriting systems, O'Donnell has shown in [O'D77] that the parallel-outermost strategy, which selects all redex occurrences that are outermost to be contracted simultaneously, is normalising. This result is a consequence of a stronger result which is also proved in [O'D77], namely that every outermost-fair rewrite sequence eventually ends in a normal form whenever the initial term has one. A rewrite sequence is said to be outermost-fair if every outermost redex occurrence is eventually eliminated.

This paper is concerned with the question of how to find a normal form in a higher-order rewriting system, in which rewriting is defined modulo simply typed λ-calculus. We extend the result by O'Donnell to the higher-order case: we show that outermost-fair rewriting is normalising for almost orthogonal higher-order rewriting systems, that satisfy some condition on the bound variables. This condition is called full extendedness. As in the first-order case, an immediate corollary of the main result is that the parallel-outermost strategy is normalising for orthogonal higher-order rewriting systems that are fully extended.

Our result extends and corrects a result by Bergstra and Klop, proved in the appendix of [BK86], which states that outermost-fair rewriting is normalising for orthogonal Combinatory Reduction Systems. Unfortunately, the proof presented in [BK86] is not entirely correct.

The remainder of this paper is organised as follows. The next section is concerned with the preliminaries. In Section 3 the notion of outermost-fair rewriting
is explained. In Section 4 the main result of this paper is proved, namely that outermost-fair rewriting is normalising for the class of almost orthogonal and fully extended higher-order rewriting systems. The present paper is rather concise in nature; for a detailed account the interested reader is referred to [Raa96].

2 Preliminaries

In this section we recall the definition of higher-order rewriting systems [Nip91, MN94], following the presentation in [Oos94, Raa96]. We further give the definitions of almost orthogonality and full extendedness. The reader is supposed to be familiar with simply typed λ-calculus with β-reduction (denoted by →β) and restricted η-expansion (denoted by →η); see for instance [Bar92, Aka93].

Simple types, written as A, B, C,..., are built from base types and the binary type constructor *. We suppose that for every type A there are infinitely many variables of type A, written as xA, yA, zA, etc.

Higher-Order Rewriting Systems. The meta-language of higher-order rewriting systems, which we call the substitution calculus as in [Oos94, OR94, Raa96], is simply typed λ-calculus.

A higher-order rewriting system is specified by a pair (A, R) consisting of a rewrite alphabet and a set of rewrite rules over A.

A rewrite alphabet is a set A consisting of simply typed function symbols. A preterm of type A over A is a simply typed λ-term of type A over A. Preterms are denoted by s, t, ... Instead of λxA.s we will write xA.s. We will often omit the superscript denoting the type of a variable. Preterms are considered modulo the equivalence relation generated by α, β and η. Every αβη-equivalence class contains a βη-normal form that is unique up to α-conversion. Such a representative is called a term. Terms are the objects that are rewritten in a higher-order rewriting system.

In the following, all preterms are supposed to be in η-normal form. Note that the set of η-normal forms is closed under β-reduction.

For the definition of a rewrite rule we first need to introduce the notion of rule-pattern, which is an adaptation of the notion of pattern due to Miller [Mil91]. A rule-pattern is a closed term of the form x1...xm,fs1...sn such that every y ∈ {x1,...,xm} occurs in f s1...sn, and it occurs only in subterms of the form yz1...zp with z1,...,zp the η-normal forms of different bound variables not among x1,...,xm. The function symbol f is called the head-symbol of the rule-pattern.

A rewrite rule over a rewrite alphabet A is defined as a pair of closed terms over A of the same type of the form x1...xm → x1...xm' t, with x1...xm A.As a rule-pattern. The head-symbol of a rewrite rule is the head-symbol of its left-hand side. Rewrite rules are denoted by R, R', ...