Skew and ω-Skew Confluence and Abstract Böhm Semantics

Zena M. Ariola[1] and Stefan Blom[2]

[1] University of Oregon
[2] University of Innsbruck

Abstract. Skew confluence was introduced as a characterization of non-confluent term rewriting systems that had unique infinite normal forms or Böhm like trees. This notion however is not expressive enough to deal with all possible sources of non-confluence in the context of infinite terms or terms extended with letrec. We present a new notion called ω-skew confluence which constitutes a sufficient and necessary condition for uniqueness. We also present a theory that can lift uniqueness results from term rewriting systems to rewriting systems on terms with letrec. We present our results in the setting of Abstract Böhm Semantics, which is a generalization of Böhm like trees to abstract reduction systems.

1 Introduction

For term rewriting systems, it is well-known that uniqueness of normal forms follows from confluence and that given termination, confluence and uniqueness of normal forms are equivalent [27, 32]. Because of this, the normal form of a term is a natural candidate for the semantics of a term. However, there are many term rewriting systems in which there are interesting terms that do not have a normal form. For example, according to the rewriting rule given below:

$$\mathsf{F}\ x \to \mathsf{Cons}(x, \mathsf{F}\ x)$$

the term $\mathsf{F}\ 1$ does not have a normal form. More precisely, it doesn't have a *finite* normal form. The term does have an *infinite* normal form: the infinite list of ones.

There are several ways to define infinite normal forms on terms. An obvious way to define them is by means of infinitary rewriting [23, 24, 32, 16, 26]. Another way is to use a definition similar to that of the Böhm Tree in the lambda calculus [13] or Böhm like trees for term rewriting systems [25]. These infinite normal forms are closely related to the notion of observational equivalence. A detailed study of the relation between different notions of infinite normal form and contextual or observational equivalence is given in [19].

An advantage of the Böhm Tree approach or Böhm semantics over infinitary rewriting is that it allows one to deal in a simple manner with rewrite rules that remove unused definitions (garbage collection). For example, consider the rewrite rule

$$\mathsf{let}\ x = M\ \mathsf{in}\ N \to N, \text{ if } x \text{ does not occur free in } N\ .$$

A. Middeldorp et al. (Eds.): Processes... (Klop Festschrift), LNCS 3838, pp. 368–403, 2005.

To define Böhm semantics for a rewrite system containing this rule, it suffices to rewrite to normal form with respect to this rule. It should also be possible to deal with this in the setting of infinitary rewriting. The rule can be seen as a rewrite rule in a combinatory reduction system and although the latest results ([26]) cover fully extended CRS's only, it is known how to deal with non fully extended rules such as the η-rule ([30]), so the only thing which is needed is a combination of these two results.

We also prefer the Böhm Tree approach because in programming languages it is very important if a result can be reached in finitely many steps or not. This is immediately clear in the Böhm Tree approach and needs a compression lemma in the infinitary rewriting case.

The notions of infinite normal form and Böhm semantics are related to the notion of information content, also called instant semantics [36] or direct approximation [28, 34]. These notions determine a prefix of the term, which can never be changed by reduction. If we rewrite term graphs represented as terms with letrec then we can follow the same intuition, but we must be careful by how we interpret the prefix. In the style of calculus used by Ariola and Klop, the letrec bindings will remain at the top of the term and keep changing during the entire reduction. Strictly speaking, the only stable prefix is Ω, our constant for undefined. However, if we forget about the syntax and look at the picture of the graph then we will see stable prefixes as usual. Effectively, we have to ignore the letrec's when we determine the stable prefix. The same situation occurs when we consider a term rewriting system in which a strategy has been encoded as a symbol which keeps traveling up and down the term. In that case the administrative symbol(s) have to be ignored. If we consider abstract rewriting systems rather than term rewriting then the information content of a term is simply an observation about that object. The combination of the information contents of all reachable objects is what we refer to as the *Abstract Böhm Semantics*.

An important property is *uniqueness* (also referred to as soundness) of Böhm semantics. This property is similar to uniqueness of normal forms and states that every two convertible terms have the same Böhm semantics. Confluence implies uniqueness of Böhm semantics. However, confluence is not necessary for guaranteeing uniqueness. Skew confluence[1], as introduced in [4, 14, 6], characterizes rewrite systems that have unique Böhm semantics with respect to a notion of direct approximant or notion of finite information content. The idea behind skew confluence is that if there exists a computation that develops a certain information content, then any other computation can be extended to develop more detailed information content. The theory of skew confluence works well for term rewriting and certain forms of term graph rewriting. However, there are problems in applying the notion to other forms of term graph rewriting and infinitary rewriting. These problems are due to the fact that the information content is not really a single observation. It actually is a set of observations. For example, when we observe a term we actually observe the finite prefixes of that term. If the term is finite, the set of observations is finite and skew confluence works.

[1] The name was suggested to us by Jan Willem Klop.