Completion for Multiple Reduction Orderings

Masahito Kurihara¹, Hisashi Kondo² and Azuma Ohuchi¹

¹ Department of Information Engineering
Hokkaido University, Sapporo, 060 Japan
[kurihara|ohuchi]@hui.ehokudai.ac.jp

² Department of Systems Engineering
Ibaraki University, Hitachi, 316 Japan
kondo@lily.dse.ibaraki.ac.jp

Abstract. We present a completion procedure (called MKB) which works with multiple reduction orderings. Given equations and a set of reduction orderings, the procedure simulates a computation performed by the parallel processes each of which executes the standard Knuth-Bendix completion procedure (KB) with one of the given orderings. To gain efficiency, however, we develop new inference rules working on objects called nodes, which are data structure consisting of a pair $s : t$ of terms associated with the information to show which processes contain the rule $s \rightarrow t$ (or $t \rightarrow s$) and which processes contain the equation $s \leftrightarrow t$. The idea is based on the observation that some of the inferences made in the processes are closely related, so we can design inference rules that simulate multiple KB inferences in several processes all in a single operation. Our experiments show that MKB is significantly more efficient than the naive simulation of parallel execution of KB procedures, when the number of reduction orderings is large enough.

1 Introduction

Given equations and a reduction ordering, the Knuth-Bendix completion procedure (KB) [9] tries to compute a complete (convergent) set of rewrite rules. As a result, it may either succeed (with a finite, convergent set of rules), or fail (because of unorientable equations), or loop (in a diverging process trying to generate an infinite set of rules). The practical interest of the completion processes is limited by the possibility of the failure and divergence. The success of the procedure heavily depends on the choice of the reduction ordering. Thus the simplest (but often effective) way of trying to hopefully avoid the failure or divergence is to change the orderings. Actually, in many existing implementations the user can interactively change (or extend) the orderings. Note, however, that this kind of implementation necessarily requires that the users have knowledge of appropriate class of reduction orderings and intuition which is hopefully correct. From the viewpoint of interface with software designers and/or AI researchers who are not familiar with termination proof techniques, automatic change of (or search for) the orderings is desired. However, the problem is that since the completion process can diverge (and we can never decide the divergence in general),
it is inappropriate to search for a correct ordering by just sequentially scanning possible orderings. This means that we have to consider, more or less, parallel execution of the completion procedures each working with one of possible orderings. However, naive implementation would result in serious inefficiency.

In this paper, we present a single completion procedure (called MKB) which works with multiple reduction orderings. Basically, given equations and a set of reduction orderings, the procedure simulates a computation performed by the parallel processes each of which executes KB with one of the given orderings. To gain efficiency, however, we develop new inference rules working on objects called nodes, which are data structure consisting of a pair \( s : t \) of terms associated with the information to show which processes contain the rule \( s \rightarrow t \) (or \( t \rightarrow s \)) and which processes contain the equation \( s \leftrightarrow t \). The idea is based on the observation that some of the inferences made in the processes are closely related, so we can design inference rules that simulate multiple KB inferences in several processes all in a single operation. Our experiments show that MKB is significantly more efficient than the naive simulation of parallel execution of KB procedures, when the number of reduction orderings is large enough. In Section 2 we review the standard completion very briefly. Then we present MKB as an inference system in Section 3. A possible MKB completion procedure is presented in Section 4. Section 5 summarizes our work.

2 Standard Completion

We assume that the reader is familiar with the general idea of term rewriting systems. The reader may consult the surveys by Dershowitz and Jouannaud[5], Klop[8], Huet and Oppen[6], Avenhaus and Madlener[1], and Plaisted[11]. In this section we briefly review the standard completion techniques, based on [2, 3, 4].

A set \( R \) of rewrite rules is convergent (or complete) if it is terminating and confluent. The system is (inter)reduced if for all \( l \rightarrow r \) in \( R \), \( r \) is irreducible with \( R \) and \( l \) is irreducible with \( R - \{ l \rightarrow r \} \). A convergent, reduced system is called canonical. Let \( \succ \) be a reduction ordering (i.e., a well-founded, strict partial ordering on terms such that \( s \succ t \) implies \( C[s\sigma] \succ C[t\sigma] \) for all contexts \( C[] \) and substitutions \( \sigma \)). Given a set \( E \) of equations and a reduction ordering \( \succ \), the standard completion procedure tries to generate a convergent (canonical) set \( R \) of rewrite rules which is contained in \( \succ \) and which induces the same equational theory as \( E \) (if rules of \( R \) are regarded as equations). The standard completion is defined in terms of the inference system KB that consists of the following six inference rules:

\[
\begin{align*}
\text{DELETE: } & \quad (E \cup \{ s \leftarrow s \}; R) \vdash (E; R) \\
\text{COMPOSE: } & \quad (E; R \cup \{ s \leftarrow t \}) \vdash (E; R \cup \{ s \leftarrow u \}) \text{ if } t \rightarrow_R u \\
\text{SIMPLIFY: } & \quad (E \cup \{ s \leftarrow t \}; R) \vdash (E \cup \{ s \leftarrow u \}; R) \text{ if } t \rightarrow_R u \\
\text{ORIENT: } & \quad (E \cup \{ s \leftarrow t \}; R) \vdash (E; R \cup \{ s \leftarrow t \}) \text{ if } s \succ t \\
\text{COLLAPSE: } & \quad (E; R \cup \{ t \leftarrow s \}) \vdash (E \cup \{ u \leftarrow s \}; R) \text{ if } l \rightarrow_R r \in R, t \rightarrow \{l \leftarrow_r \} u, \\
& \quad \text{ and } t \triangleright l \\
\text{DEDUCE: } & \quad (E; R) \vdash (E \cup \{ s \leftarrow t \}; R) \quad \text{ if } s \leftarrow_R u \rightarrow_R t
\end{align*}
\]