SLOTHROP: Knuth-Bendix Completion with a Modern Termination Checker

Ian Wehrman, Aaron Stump, and Edwin Westbrook*

Dept. of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO USA

Abstract. A Knuth-Bendix completion procedure is parametrized by a reduction ordering used to ensure termination of intermediate and resulting rewriting systems. While in principle any reduction ordering can be used, modern completion tools typically implement only Knuth-Bendix and path orderings. Consequently, the theories for which completion can possibly yield a decision procedure are limited to those that can be oriented with a single path order.

In this paper, we present a variant on the Knuth-Bendix completion procedure in which no ordering is assumed. Instead we rely on a modern termination checker to verify termination of rewriting systems. The new method is correct if it terminates; the resulting rewrite system is convergent and equivalent to the input theory. Completions are also not just ground-convergent, but fully convergent. We present an implementation of the new procedure, SLOTHROP, which automatically obtains such completions for theories that do not admit path orderings.

1 Introduction

A Knuth-Bendix completion procedure is a technique for solving the word problem for a finite set of identities. In this procedure, the user provides the set of identities as well as a reduction order on terms.

Using *unfailing completion* [3], ground-convergent completions can be discovered even when no compatible reduction order exists. However, successful completions may contain unoriented equations along with oriented rewrite rules. The resulting system is convergent only for ground terms and often contains more rules than a fully convergent completion.

Nonetheless, many theories are easily oriented by a few useful classes of reduction orderings, such as Knuth-Bendix and recursive path orderings (KBO and RPO, respectively). The wide applicability of KBO and RPO has led to the success of completion procedures. While in principle any reduction ordering can be used, modern completion tools like WALDMEISTER [9] typically implement only these two classes of orderings. (However, the tool CiME also implements polynomial orderings [4].) Consequently, the theories for which completion can

* This work was partially supported by NSF grant CCF-0448275.

F. Pfenning (Ed.): RTA 2006, LNCS 4098, pp. 287–296, 2006.

possibly yield a decision procedure in the form of a convergent rewrite system are limited to those with completions that admit such a path order.

In this paper, we present a new variant on the standard Knuth-Bendix completion procedure in which no ordering is explicitly provided. Instead a constraint rewriting system is constructed during execution, and its reduction relation is used for an ordering. Termination of the constraint system then implies termination of the intermediate rewrite systems. In the implementation, we rely on modern termination-checking methods to verify the termination of the constraint systems.

The new method is correct if it terminates; the resulting rewrite system is convergent and equivalent to the input theory. In addition, the completions are not just ground-convergent, but fully convergent. The method is also complete for finite executions in that if there exists a successful, finite execution of a standard Knuth-Bendix completion procedure with some reduction ordering, then an equivalent execution in the modified system exists.

We begin in Sect. 2 with a presentation of standard Knuth-Bendix completion and statements of correctness. In Sect. 3 we introduce our new completion variant in which the reduction order is left implicit and prove its correctness. In Sect. 4, we present an implementation of the new procedure called SLOTHROP, and in Sect. 5 we discuss its performance and results, including convergent completions automatically obtained for the first time. Finally in Sect. 6, we discuss areas of future interest with respect to the new technique.

2 Knuth-Bendix Completion

In this section, we review the basic definition and properties of completion procedures. We use standard notation for terms and term rewriting systems, as presented in [1].

A *Knuth-Bendix completion procedure* [8] is an algorithm that takes as input a reduction ordering $>$ and a finite set of equations E and attempts to produce a decision procedure for the word problem for E in the form of a rewriting system. Completion algorithms attempt to construct a convergent rewriting system R that is equivalent to E (i.e., with the same equational theory, $\overset{*}{\leftrightarrow}_E = \overset{*}{\leftrightarrow}_R$) by generating a possibly infinite sequence of intermediate rewriting systems which yield approximations of the equational theory of E.

Bachmair formulated Knuth-Bendix completion as an equational inference system [2]. We refer to this standard system as $\mathcal{C}$ because it serves as the basis of a correctness condition for our refinement of the procedure. The rules of the inference system $\mathcal{C}$ are shown in Fig. 1. A *deduction* of $\mathcal{C}$, written $(E, R) \vdash_{\mathcal{C}} (E', R')$, consists of finite sets of identities E, E' and rewrite systems R, R'. A *finite execution* γ of the system $\mathcal{C}$ is the pair $(E_0, \emptyset)$ followed by a finite sequence of deductions

$$(E_0, \emptyset) \vdash_{\mathcal{C}} (E_1, R_1) \cdots \vdash_{\mathcal{C}} (E_n, R_n),$$

where E_0 is the input theory provided by the user, and each deduction results from an application of one of the inference rules of $\mathcal{C}$. (We consider only finite

ORIENT:	$\dfrac{(E \cup \{s \doteq t\}, R)}{(E, R \cup \{s \to t\})}$	if $s > t$
DEDUCE:	$\dfrac{(E, R)}{(E \cup \{s = t\}, R)}$	if $s \leftarrow_R u \rightarrow_R t$
DELETE:	$\dfrac{(E \cup \{s = s\}, R)}{(E, R)}$	
SIMPLIFY:	$\dfrac{(E \cup \{s \doteq t\}, R)}{(E \cup \{u \doteq t\}, R)}$	if $s \rightarrow_R u$
COMPOSE:	$\dfrac{(E, R \cup \{s \to t\})}{(E, R \cup \{s \to u\})}$	if $t \rightarrow_R u$
COLLAPSE:	$\dfrac{(E, R \cup \{s \to t\})}{(E \cup \{v = t\}, R)}$	if $s \xrightarrow{\sqsupset}_R v$

Fig. 1. Standard Knuth-Bendix Completion ($\mathcal{C}$)

executions of $\mathcal{C}$ in this paper; the infinite case is discussed briefly in Sect. 6.) The *length* of γ, written $|\gamma|$, is the number of deductions in γ. A finite execution γ of $\mathcal{C}$ *succeeds* if $E_{|\gamma|} = \emptyset$ and $R_{|\gamma|}$ is a convergent rewrite system equivalent to E as described above; otherwise it *fails*. Elsewhere ([2], [1]), $\mathcal{C}$ is proved correct in that any successful, finite execution γ results in a convergent rewrite system $R_{|\gamma|}$ equivalent to the input identities E.

The main difficulty with the standard completion procedure is in finding an appropriate reduction order. Choosing a suitable RPO, KBO or polynomial interpretation (the only options available in known tools) is difficult even for experienced users, and for many theories no such path ordering exists. In the next section, we solve this problem with a variant on the standard completion procedure which discovers a suitable reduction ordering without input from the user.

3 Completion with Termination Checking

We now present a modification of the standard Knuth-Bendix completion procedure. The primary difference is that no reduction order is explicitly provided as input, only a finite set of identities. Lacking any specific reduction order to guide the search, we preserve termination of each intermediate rewrite system R_i by ensuring that some reduction order $\succ_i$ compatible with R_i exists. The orders $\succ_i$ are constructed using terminating rewrite systems C_i, specifically as the transitive closure of the reduction relation on C_i, written $\xrightarrow{+}_{C_i}$. This relation is a well-founded order exactly when the system C_i is terminating. While in the standard system $\mathcal{C}$ a rule $s \to t$ is added by ORIENT to R_i only if $s > t$ with the user-specified reduction order, in the modified system the rule is added only if the addition of $s \to t$ to C_i preserves termination. Of course, deciding termination is not possible in general. In Sect. 4, we discuss how this test is accomplished in practice.

Figure 2 provides the inference rules for a modification of the standard completion procedure, which we refer to as system $\mathcal{A}$. A deduction of $\mathcal{A}$, written $(E, R, C) \vdash_{\mathcal{A}} (E', R', C')$, consists of identities E, E' and rewrite systems R, R' as in standard completion, and finite *constraint* rewriting systems C, C' new to $\mathcal{A}$. A finite execution α of the system $\mathcal{A}$ is the triple $(E_0, \emptyset, \emptyset)$ followed by a finite sequence of deductions

$$(E_0, \emptyset, \emptyset) \vdash_{\mathcal{A}} (E_1, R_1, C_1) \cdots \vdash_{\mathcal{A}} (E_n, R_n, C_n),$$

with E_0 the set of input identities and where each deduction results from an application of one inference rule from $\mathcal{A}$. We consider only finite executions of $\mathcal{A}$; infinite executions are discussed in Sect. 6. We write $|\alpha|$ to denote the length of the sequence. An execution α of $\mathcal{A}$ is *equivalent* to an execution γ of $\mathcal{C}$ when the intermediate equations and rewrite systems are the same at each step. A finite execution α of system $\mathcal{A}$ succeeds when $E_{|\alpha|} = \emptyset$ and $R_{|\alpha|}$ is a convergent rewrite system equivalent to E. Because we only consider finite executions, every execution that does not succeed fails.

ORIENT:	$\dfrac{(E \cup \{s \doteq t\}, R, C)}{(E, R \cup \{s \to t\}, C \cup \{s \to t\})}$	if $C \cup \{s \to t\}$ terminates
DEDUCE:	$\dfrac{(E, R, C)}{(E \cup \{s = t\}, R, C)}$	if $s \leftarrow_R u \rightarrow_R t$
DELETE:	$\dfrac{(E \cup \{s = s\}, R, C)}{(E, R, C)}$	
SIMPLIFY:	$\dfrac{(E \cup \{s \doteq t\}, R, C)}{(E \cup \{u \doteq t\}, R, C)}$	if $s \rightarrow_R u$
COMPOSE:	$\dfrac{(E, R \cup \{s \to t\}, C)}{(E, R \cup \{s \to u\}, C)}$	if $t \rightarrow_R u$
COLLAPSE:	$\dfrac{(E, R \cup \{s \to t\}, C)}{(E \cup \{v = t\}, R, C)}$	if $s \overset{\sqsupset}{\rightarrow}_R v$

Fig. 2. Modified Knuth-Bendix Completion ($\mathcal{A}$)

The rules DEDUCE, DELETE, SIMPLIFY, COMPOSE and COLLAPSE of $\mathcal{A}$ are identical to those of $\mathcal{C}$, except for the presence of the constraint system C which is carried unmodified from antecedent to consequent. The critical difference between $\mathcal{A}$ and $\mathcal{C}$ is in the definition of the ORIENT rule. In the standard system $\mathcal{C}$, an identity $s \doteq t$ of E is added to R as rule $s \to t$ only when $s > t$ for the given reduction order. In the modified system $\mathcal{A}$, we add the rule $s \to t$ to R only when the augmented constraint system $C \cup \{s \to t\}$ is terminating. The system $\mathcal{A}$ accepts as input only the finite set of identities E; no reduction order is explicitly provided.

We now state correctness of $\mathcal{A}$ for finite executions (partial correctness). The proof proceeds by showing that $\mathcal{A}$ simulates a standard Knuth-Bendix completion procedure $\mathcal{C}$. For each finite execution α of $\mathcal{A}$, we construct an execution

γ of $\mathcal{C}$ with an equivalent sequence of deductions as α. The constraint systems are used to show that any finite execution is equivalent to one which uses the single order induced by the final constraint system. This is important because completion is not generally correct when reduction orders are changed during execution, even if each is compatible with the immediate intermediate rewrite system [10]. The induced order is $\xrightarrow{+}_{C_{|\alpha|}}$, the transitive closure of the reduction relation of the final constraint system $C_{|\alpha|}$.

Theorem 1 (Partial Correctness of $\mathcal{A}$). *Let α be a finite execution of the system $\mathcal{A}$. Then there exists an equivalent execution γ of $\mathcal{C}$ using reduction order $\xrightarrow{+}_{C_{|\alpha|}}$.*

3.1 Partial Completeness

We now show a limited form of completeness for our procedure with respect to standard Knuth-Bendix completion. Namely, for any successful execution of the standard completion procedure $\mathcal{C}$ there exists a corresponding execution of the modified procedure $\mathcal{A}$ with the same deductions. This shows that our method can at least construct decision procedures for those theories that are decidable by the standard method. In Sect. 5, we give an example of a theory for which our method constructs a convergent completion that, to the authors' knowledge, cannot be automatically constructed by any tool that implements $\mathcal{C}$ due to inability to specify an appropriate reduction order.

Theorem 2 (Partial Completeness of $\mathcal{A}$). *For any finite execution γ of $\mathcal{C}$ with reduction order $>$, there exists an equivalent execution α of $\mathcal{A}$. Furthermore, $\xrightarrow{+}_{C_{|\gamma|}} \subseteq >$.*

Proof. By induction on γ. The beginning execution is $\gamma = (E_0, \emptyset)$, which translates to $\alpha = (E_0, \emptyset, \emptyset)$. Otherwise, $\gamma = \gamma' \vdash_{\mathcal{C}} (E_k, R_k)$ and by IH there exists α' that satisfies the claim for γ', and also $\rightarrow_{C_{k-1}} \subseteq >$. Let $C_k = C_{k-1}$ if the final deduction is the result of any rule except ORIENT, and $C_k = C_{k-1} \cup \{s \rightarrow t\}$ otherwise, with $\{s \rightarrow t\} = R_k - R_{k-1}$. We claim $\alpha = \alpha' \vdash_{\mathcal{A}} (E_k, R_k, C_k)$ and show this is a correct execution of $\mathcal{A}$. This is trivial for rules other than ORIENT, since their side conditions do not mention the constraint systems. Otherwise, $s > t$ and $\xrightarrow{+}_{C_{k-1}} \subseteq >$ which implies $\xrightarrow{+}_{C_k} \subseteq >$. This in turn implies that C_k is terminating because its rules are compatible with the reduction order $>$.

This theorem demonstrates the existence of a successful execution of $\mathcal{A}$ for every successful finite execution of $\mathcal{C}$. But note that that the rule ORIENT in $\mathcal{A}$ can orient an equation $s = t$ in either direction when both $C \cup \{s \rightarrow t\}$ and $C \cup \{t \rightarrow s\}$ are terminating systems. Consequently, an execution of $\mathcal{A}$ as defined above will fail if a poor decision is made during orientation. The ability to construct a successful execution relies on a non-deterministic orientation choice. Deterministically, an execution of $\mathcal{A}$ becomes a binary tree in which each node is an instance of the rule ORIENT. In practice, we must *search* for a successful execution. We ensure discovery of such an execution (corresponding to a path from the root $(E_0, \emptyset)$) by fairly advancing each of the individual executions.

4 Implementation

We have implemented our modified Knuth-Bendix completion procedure in a 7000-line Ocaml program called SLOTHROP.[1] The implementation is based on a particular completion strategy developed and proved correct by Huet [7], and later by Bachmair using the inference system $\mathcal{C}$ [2]. The implementation itself is originally based on an ML implementation of Huet's algorithm by Baader and Nipkow [1] and makes use of data structures programmed by Filliâtre [5].

The main technical challenge in the implementation is with the ORIENT rule. As is well known, determining whether or not a term rewriting system terminates is undecidable in general. However, modern termination-checking tools, such as APROVE [6], succeed in proving many systems terminating or nonterminating with almost alarming success. In our implementation, we take advantage of this success and use APROVE as an oracle to answer queries about the termination of constraint rewriting systems in each orientation step. If APROVE fails to prove a system terminating or nonterminating, we treat it as a nonterminating system and delay its treatment. However, the array of techniques used by APROVE to show termination includes recursive path orders among many others, so there is little difficulty recognizing the termination of systems compatible with such an order. Furthermore, since APROVE is able to prove termination of systems that are not compatible with a path order, SLOTHROP can find convergent completions of theories other modern completion tools (e.g., WALDMEISTER) cannot. One example of such a theory is given in Sect. 5.

Integrating a separate termination checker also provides separation-of-concerns benefits for theorem proving. As the power and speed of the APROVE tool, so does SLOTHROP. This also provides the opportunity to leverage other termination checkers with different properties (e.g., one which is faster but less powerful might be useful for simple theories).

Another important aspect of our implementation is the manner in which different branches of executions are explored. When APROVE determines that some equation $s = t$ can be oriented either way, both branches are explored. Implementation of this exploration is critical to performance. The binary tree of executions is potentially infinite, and branches whenever orderings exist that are compatible with both orientations of an equation.

A breadth-first search of the branches is sufficient for partial completeness; if there is some successful finite execution corresponding to a branch on the tree, it will eventually be expanded. In practice, however, this strategy spends too much time in uninteresting areas of the search space, and prevents SLOTHROP from finding completions for any but the most modest theories in a reasonable amount of time. A more effective strategy is a best-first search in which the next execution to advance is chosen based on a cost function defined by

$$\mathrm{cost}(E, R, C) = \mathrm{size}(C) + \mathrm{size}(E) + \mathrm{size}(\Gamma(R)),$$

[1] SLOTHROP is available online at `http://cl.cse.wustl.edu/` on the software page.

where $\Gamma(R)$ denotes the set of all nontrivial critical pairs of R. With this strategy, size(C) can be thought of as the cost to reach the current intermediate step in the execution and size(E) + size($\Gamma(R)$) as a heuristic estimate for the cost to find a convergent completion.

In some cases, APROVE is unable to prove termination of a system with either orientation of a particular equation. Here, we do not discard the system entirely, but attempt to orient other equations in hope that the previously unorientable equation will simplify into an orientable (or trivial) one. In the current implementation of SLOTHROP, treatment of such systems is delayed until others are explored which can be proved terminating or nonterminating. This heuristic is suitable for simple systems, but better ones are needed for more difficult theories.

5 Performance and New Results

SLOTHROP is capable of completing a variety of theories fully automatically in a modest amount of time. For example, the standard 10-rule completion of the group axioms is discovered in under 3 seconds on a modern desktop PC. On the way to this completion, it encounters 27 orientations, roughly half of which are not trivially nonterminating and must be verified with APROVE. On the execution branch that leads to a completion, however, only two orientation steps are required. SLOTHROP automatically completes the theory of groups plus a single endomorphism (GE_1) in under 10 seconds, requiring about 100 calls to APROVE. A large theory with 21 equations corresponding to propositional proof simplification rules [12] is considerably more difficult to complete because of the number of orientations. Nonetheless, SLOTHROP does find a completion without user intervention after about 7 hours and 3000 calls to APROVE.

The majority of SLOTHROP's running time is spent waiting for calls to APROVE. Although we have encountered many examples of rewriting systems which APROVE can show terminating after a prohibitively long amount of time, in practice we have found that it is uncommon for such difficult systems to appear on the branch of a successful execution. Most calls to APROVE that occur on successful branches return in under 2 seconds. Completeness of SLOTHROP can be exchanged for performance enhancements by calling APROVE with a short timeout. The above completions were obtained with a 5-second timeout.

Since SLOTHROP is not restricted to a given reduction ordering, it can also search for multiple completions of a given theory. For example, it finds two

$1 * x = x$	$x^{-1} * x = 1$	$(x * y) * z = x * (y * z)$
$f(x * y) = f(x) * f(y)$	$g(x * y) = g(x) * g(y)$	$f(x) * g(y) = g(y) * f(x)$

Fig. 3. The Theory of Two Commuting Group Endomorphisms (CGE_2)

completions of the basic group axioms corresponding to both orientations of the associativity rule. It also finds four completions of GE_1 corresponding to the orientations of the associativity endomorphism rules. It also discovers a number of other larger completions of the same theory in which endomorphism are oriented differently depending on the context.

Additionally, a convergent completion can be obtained by Slothrop for the theory of two commuting group endomorphisms (CGE_2), shown in Fig. 5. The reader may verify that no RPO or KBO is compatible with the theory (in particular, the final commutativity rule). A completion was recently obtained for the first time by hand [11] — rules derived from critical pairs were manually oriented, local confluence checked, and termination of the resulting system verified by AProVE.

Using unfailing completion [3], Waldmeister is able to complete CGE_2 as well, but constructs a larger system which is ground-confluent only — i.e, it contains equations as well as rewrite rules. This system is often less helpful than a small convergent completion, for example, in characterizing the normal forms of the system for algebraic proof mining [12]. Furthermore, Waldmeister does not appear to be able to find this ground-convergent completion fully automatically; a carefully selected Knuth-Bendix ordering (given in [11]) must be provided. Slothrop is able to find the convergent completion with no input from the user other than the theory itself. (This still takes more than an hour, however, even using the heuristic described in Sect. 4.)

6 Conclusion and Future Work

We have presented a new variant on Knuth-Bendix completion which does not require the user to provide a reduction ordering to orient identities. The procedure is correct and complete, but only for finite executions. An implementation of the procedure, called Slothrop, can find convergent completions for a number of interesting theories without any input from the user, including one (CGE_2) which cannot be obtained by any existing tool.

A primary goal of future work is to increase the efficiency of Slothrop. Basic heuristic search techniques have made the algorithm feasible for many theories, but it is still prohibitively slow for large theories — completion of the CGE_3 has not yet been achieved. The performance of Slothrop also does not approach that of well-tuned equational theorem provers such as Waldmeister for most tasks. Modern search and learning techniques, e.g. as developed for SAT, may be applicable to the search for a convergent completion. Finally, we would like to explore extensions to termination checking techniques to allow proofs to be constructed incrementally. This may significantly decrease the amortized time to prove a series of term rewriting systems terminating, since Slothrop tends to make a number of successive calls on rewrite systems whose rules form increasing chains.

Infinite Executions. While the provided argument for completeness carries to infinite executions essentially unmodified, it is not the case that all non-

failing runs are successful. In particular, termination of the infinite union of the intermediate constraint systems does not follow from termination of the individual systems; this is because in general the union of an infinite number of finite, terminating rewrite systems is not itself terminating. For example, consider the family of (string) rewriting systems $R_j = \cup_{0 \leq i \leq j} \{fg^i f \to fg^{i+1}\}$. For any $k \in \mathbb{N}$ it is easy to see that $\cup_{0 \leq j \leq k} R_j$ is terminating. But it is not the case that $\cup_{j \in \mathbb{N}} R_j$ is terminating, for it contains the infinite derivation $ff \to fgf \to fggf \to \cdots$.

Instead, it must be shown in a proof of correctness for the infinite case that *some* successful branch of execution always exists, and that it will always be found in the search for a completion. The authors believe the modified procedure to be correct in the infinite case, making it usable as a semidecision procedure for theories. We have a proof sketch of correctness for the infinite case of the system $\mathcal{A}$, and a complete proof is in progress.

Acknowledgements. The authors are especially appreciative of Peter Schneider-Kamp and the AProVE team for their willingness to make changes to their system necessary for integration with SLOTHROP. We thank Stephan Falke for testing SLOTHROP and providing a theory that requires delayed treatment of unorientable equations. We also thank Li-Yang Tan and the referees for helpful comments on drafts.

References

1. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.
2. L. Bachmair. *Canonical Equational Proofs.* Progress in Theoretical Computer Science. Birkhäuser, 1991.
3. L. Bachmair, N. Dershowitz, and D.A. Plaisted. Completion Without Failure. In *Resolution of Equations in Algebraic Structures*, volume 2, Rewriting Techniques, pages 1–30. Academic Press, 1989.
4. Evelyne Contejean, Claude Marché, and Xavier Urbain. CiME3, 2004. Available at `http://cime.lri.fr/`.
5. Jean-Christophe Filliâtre. Ocaml data structures. Available at `http://www.lri.fr/~filliatr/software.en.html`.
6. J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Automated Termination Proofs with AProVE. In V. van Oostrom, editor, *the 15th International Conference on Rewriting Techniques and Applications*, pages 210–220. Springer, 2004.
7. G. Huet. A Complete Proof of Correctness of the Knuth-Bendix Completion Algorithm. *Journal of Computer and System Science*, 23(1):11–21, 1981.
8. D. Knuth and P. Bendix. Simple Word Problems in Universal Algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.
9. B. Löchner and T. Hillenbrand. The Next Waldmeister Loop. In A. Voronkov, editor, *18th International Conference on Automated Deduction*, pages 486–500, 2002.

10. A. Sattler-Klein. About Changing the Ordering During Knuth-Bendix Completion. In P. Enjalbert E. W. Mayr and K. W. Wagner, editors, *Symposium on Theoretical Aspects of Computer Science*, volume 775 of *LNCS*, pages 176–186. Springer, 1994.
11. A. Stump and B. Löchner. Knuth-Bendix Completion of Theories of Commuting Group Endomorphisms. *Information Processing Letters*, 2006. To appear.
12. I. Wehrman and A. Stump. Mining Propositional Simplification Proofs for Small Validating Clauses. In A. Armando and A. Cimatti, editors, *3rd International Workshop on Pragmatics of Decision Procedures in Automated Reasoning*, 2005.