Maximal Completion*

Dominik Klein[1] and Nao Hirokawa[1]

1 **School of Information Science**
Japan Advanced Institute of Science and Technology
Nomi, Japan
`{dominik.klein,hirokawa}@jaist.ac.jp`

Abstract

Given an equational system, completion procedures compute an equivalent and complete (terminating and confluent) term rewrite system. We present a very simple and efficient completion procedure, which is based on MaxSAT solving. Experiments show that the procedure is comparable to recent powerful completion tools.

1998 ACM Subject Classification F.4.2 Grammars and Other Rewriting Systems

Keywords and phrases Term Rewriting, Knuth-Bendix Completion, Multi-completion

Digital Object Identifier 10.4230/LIPIcs.RTA.2011.71

Category System Description

1 Introduction

Completion tries to construct from an equational system (ES) an equivalent complete (confluent and terminating) term rewrite system (TRS). The standard completion procedure by Knuth and Bendix [14] and Huet [12] takes not only a target ES but also a reduction order as a parameter. This reduction order is used to ensure termination of a resulting complete TRS. Because the choice of a reduction order is critical for getting a successful run of the procedure, several attempts have been made to automatically find such an order. Here we mention the pioneering work of Kurihara and Kondo [16] on running completion using multiple orders in parallel and the approach by Wehrman et al. [23] to automatically construct a reduction order using a termination tool on the fly. Very recently Sato et al. [21] showed how both approaches can be combined.

We present a new completion procedure, dubbed maximal completion. This procedure induces a set of (exponentially) many TRSs to find a desired complete TRS from the set. Via a natural encoding into maximal satisfiability problems, the procedure can be easily implemented by a MaxSAT (or MaxSMT) solver. Experiments by our completion tool Maxcomp show that this approach performs comparable with the above approaches. The tool Maxcomp is available at:

`http://www.jaist.ac.jp/project/maxcomp/`

This paper is concerned with constructing *complete* term rewriting systems only. But we anticipate that our approach can be adapted for unfailing completion [6], which gives up the aim of trying to construct a complete system. Instead it only aims to construct a *ground*

* The research described in this paper is supported by the Grant-in-Aid for Young Scientists (B) 22700009 of the Japan Society for the Promotion of Science.

complete (ground terminating and ground confluent) system, which is effectively used in first-order theorem proving.

The paper is structured as follows: In Section 2 we introduce maximal completion, and its automation techniques are described in Section 3. Section 4 relates the procedure to existing ones. Empirical results are reported in Section 5, where our tool Maxcomp is compared with the state-of-art completion tools Slothrop [23] and mkbTT [25]. Finally, we conclude the presentation in Section 6 by mentioning potential future work. Throughout the paper, we assume familiarity with term rewriting in general, and most notions and notations are borrowed from [3, 22].

2 Maximal Completion

A TRS is *complete* if it is terminating and confluent. We say that $\mathcal{R}$ is a complete TRS *for* an ES $\mathcal{E}$ if $\mathcal{R}$ is a complete TRS with $\leftrightarrow_{\mathcal{R}}^{*} = \leftrightarrow_{\mathcal{E}}^{*}$. The completion problem is to find a complete TRS for a given $\mathcal{E}$.

To derive a procedure for completion, we recall the definition of *critical pairs*. An *overlap* $(\ell_1 \to r_1, p, \ell_2 \to r_2)_\mu$ of a TRS $\mathcal{R}$ consists of variants $\ell_1 \to r_1$ and $\ell_2 \to r_2$ of rewrite rules of $\mathcal{R}$ without common variables, a non-variable position $p \in \mathcal{P}\mathsf{os}(\ell_2)$, and a most general unifier μ of ℓ_1 and $\ell_2|_p$. If $p = \epsilon$ then we require that $\ell_1 \to r_1$ and $\ell_2 \to r_2$ are not variants of the same rewrite rule. The induced *critical pair* is $\ell_2\mu[r_1\mu]_p \approx r_2\mu$, and the set of all such pairs of $\mathcal{R}$ is written as $CP(\mathcal{R})$. Note that pairs (s,t) of terms are denoted by $s \approx t$ or $s \to t$ depending on the contexts. Below, we write $\downarrow_{\mathcal{R}}$ for the joinability relation $\to_{\mathcal{R}}^{*} \cdot {}_{\mathcal{R}}^{*}\!\leftarrow$.

▶ **Lemma 1.** *$\mathcal{R}$ is a complete TRS for an ES $\mathcal{E}$ if and only if $\mathcal{R}$ is terminating, $\mathcal{R} \subseteq \leftrightarrow_{\mathcal{E}}^{*}$, and $\mathcal{E} \cup CP(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}}$.*

Proof. For the "if"-direction, by Knuth and Bendix' confluence criterion [14, 11], confluence of $\mathcal{R}$ follows from $CP(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}}$ and termination of $\mathcal{R}$. Moreover, $\mathcal{E} \subseteq \downarrow_{\mathcal{R}}$ and $\mathcal{R} \subseteq \leftrightarrow_{\mathcal{E}}^{*}$ yield $\leftrightarrow_{\mathcal{R}}^{*} = \leftrightarrow_{\mathcal{E}}^{*}$. The "only if"-direction is immediate from $\leftrightarrow_{\mathcal{E}}^{*} \subseteq \leftrightarrow_{\mathcal{R}}^{*} \subseteq \downarrow_{\mathcal{R}}$. ◀

Lemma 1 yields a simple completion procedure. Let $\mathcal{E}$ be an ES. We assume that two parameter functions $\mathfrak{R}$ and S are given and the next two conditions hold for every ES $\mathcal{C}$: $S(\mathcal{C})$ is a set of equalities in $\leftrightarrow_{\mathcal{E}}^{*}$, and $\mathfrak{R}(\mathcal{C})$ is a set of terminating TRSs $\mathcal{R}$ with $\mathcal{R} \subseteq \leftrightarrow_{\mathcal{E}}^{*}$.

▶ **Definition 2.** Given ESs $\mathcal{E}$ and $\mathcal{C}$, the procedure φ is defined as

$$\varphi(\mathcal{C}) = \begin{cases} \mathcal{R} & \text{if } \mathcal{E} \cup CP(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}} \text{ for some } \mathcal{R} \in \mathfrak{R}(\mathcal{C}) \\ \varphi(\mathcal{C} \cup S(\mathcal{C})) & \text{otherwise} \end{cases}$$

Note that $\varphi(\mathcal{C})$ is neither unique nor defined in general.

▶ **Theorem 3.** *$\varphi(\mathcal{E})$ is a complete TRS for an ES $\mathcal{E}$, if it is defined.*

The procedure φ repeatedly expands $\mathcal{C}$ (initially $\mathcal{E}$) by $S(\mathcal{C})$ until $\mathfrak{R}(\mathcal{C})$ contains a complete TRS for $\mathcal{E}$. For its success the choice of $\mathfrak{R}(\mathcal{C})$ and $S(\mathcal{C})$ is crucial. Let $t\downarrow_{\mathcal{R}}$ denote a fixed normal form of t with respect to $\mathcal{R}$. We say that a TRS $\mathcal{R}$ is *over* an ES $\mathcal{C}$ if $\mathcal{R} \subseteq \mathcal{C} \cup \mathcal{C}^{-1}$. The set of all terminating TRSs over $\mathcal{C}$ is denoted by $\mathfrak{T}(\mathcal{C})$. We propose to use

$$\mathfrak{R}(\mathcal{C}) = \mathit{Max}\, \mathfrak{T}(\mathcal{C})$$

$$S(\mathcal{C}) = \bigcup_{\mathcal{R} \in \mathfrak{R}(\mathcal{C})} \{ s\downarrow_{\mathcal{R}} \approx t\downarrow_{\mathcal{R}} \mid s \approx t \in \mathcal{E} \cup CP(\mathcal{R}) \text{ and } s\downarrow_{\mathcal{R}} \neq t\downarrow_{\mathcal{R}} \}$$

Here *Max* computes all maximal sets of rewrite rules (called *maximal* TRSs) in its given family of TRSs, and this is the reason that we call our method *maximal* completion. Part (b) in the next lemma explains why *non*-maximal TRSs in $\mathfrak{T}(\mathcal{R})$ can be ignored safely.

▶ **Lemma 4.** *The following claims hold:*

(a) Let $\mathcal{R} \subseteq \mathcal{R}' \subseteq \leftrightarrow^*_{\mathcal{E}}$ and $\mathcal{R}'$ terminating. $\mathcal{R}'$ is complete for $\mathcal{E}$ if $\mathcal{R}$ is complete for $\mathcal{E}$.

(b) $\mathcal{E} \cup CP(\mathcal{R}) \subseteq {\downarrow_{\mathcal{R}}}$ for some $\mathcal{R} \in \mathfrak{T}(\mathcal{C})$ iff $\mathcal{E} \cup CP(\mathcal{R}) \subseteq {\downarrow_{\mathcal{R}}}$ for some $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$.

Proof. (a) Due to completeness of $\mathcal{R}$, we have $\mathcal{E} \cup CP(\mathcal{R}') \subseteq {\downarrow_{\mathcal{R}}}$. The claim follows together with $\rightarrow_{\mathcal{R}} \subseteq \rightarrow_{\mathcal{R}'}$. (b) The 'only if'-direction is straightforward from the first claim, and the converse is trivial. ◀

We illustrate maximal completion with an example. In examples the inverse $t \approx s$ of an indexed rule $i\colon s \approx t$ is denoted as i'.

▶ **Example 5.** Consider the ES $\mathcal{E}$ consisting of the equalities:

$$1\colon\ \mathsf{s}(\mathsf{p}(x)) \approx x \qquad 2\colon\ \mathsf{p}(\mathsf{s}(x)) \approx x \qquad 3\colon\ \mathsf{s}(x) + y \approx \mathsf{s}(x+y)$$

We compute $\varphi(\mathcal{E})$ with the above $S(\mathcal{C})$ and $\mathfrak{R}(\mathcal{C})$.

(i) $\mathfrak{R}(\mathcal{E})$ consists of two TRSs $\{1,2,3\}$ and $\{1,2,3'\}$. Since the join-condition of φ does not hold, we have $\varphi(\mathcal{E}) = \varphi(\mathcal{E} \cup S(\mathcal{E}))$. Here $S(\mathcal{E})$ consists of two equalities:

$$4\colon\ x + y \approx \mathsf{s}(\mathsf{p}(x) + y) \qquad 5\colon\ \mathsf{p}(\mathsf{s}(x) + y) \approx x + y$$

(ii) $\mathfrak{R}(\{1,\dots,5\})$ consists of the two TRSs $\{1,2,3,4',5\}$ and $\{1,2,3',4',5\}$. Again the join-condition does not hold. Thus, $\varphi(\{1,\dots,5\}) = \varphi(\{1,\dots,9\})$, where

$$\begin{array}{ll} 6\colon\ (x+y)+z \approx \mathsf{s}((\mathsf{p}(x)+y)+z) & 7\colon\ \mathsf{p}((\mathsf{s}(x)+y)+z) \approx (x+y)+z \\ 8\colon\ \mathsf{p}(x)+y \approx \mathsf{p}(x+y) & 9\colon\ \mathsf{p}((x+y)+z) \approx (\mathsf{p}(x)+y)+z \end{array}$$

(iii) $\mathfrak{R}(\{1,\dots,9\})$ consists of four TRSs including the TRS $\mathcal{R}$ of

$$\{1,2,3,4',5,6',7,8,9'\}$$

which satisfies the join-condition. Thus, $\varphi(\{1,\dots,9\}) = \mathcal{R}$.

Hence, $\varphi(\mathcal{E}) = \mathcal{R}$ and it is a complete TRS for $\mathcal{E}$.

Very often a complete TRS resulting from maximal completion contains many superfluous rules. It is known that this problem is resolved by computing reduced TRSs (cf. [12]). A TRS $\mathcal{R}$ is *reduced* if $\ell \in NF(\mathcal{R} \setminus \{\ell \to r\})$ and $r \in NF(\mathcal{R})$ for all rules $\ell \to r \in \mathcal{R}$. We write $\widehat{\mathcal{R}}$ for the reduced TRS

$$\{\ell \to r \in \widetilde{\mathcal{R}} \mid \ell \in NF(\widetilde{\mathcal{R}} \setminus \{\ell \to r\})\}$$

where $\widetilde{\mathcal{R}} = \{\ell \to r{\downarrow_{\mathcal{R}}} \mid \ell \to r \in \mathcal{R}\}$. The TRS $\widehat{\mathcal{R}}$ fulfills the desired property:

▶ **Lemma 6.** *If a TRS $\mathcal{R}$ is complete for $\mathcal{E}$, then $\widehat{\mathcal{R}}$ is complete for $\mathcal{E}$.*

Proof. Using the fact that $\widehat{\mathcal{R}}$ is complete and $\leftrightarrow^*_{\mathcal{R}} = \leftrightarrow^*_{\widehat{\mathcal{R}}}$ (see [19]). ◀

▶ **Example 7** (continued from Example 5)**.** The reduced version $\widehat{\mathcal{R}}$ is $\{1,2,3,8\}$.

As Example 5 illustrates, maximality dismisses undesirable complete TRSs like empty or singletons in $\mathfrak{T}(\mathcal{C})$. This is one major source of efficiency in maximal completion. We refer to the subsequent two sections for further discussion on $\mathfrak{R}(\mathcal{C})$ and $S(\mathcal{C})$.

3 Automation

We describe how to automate the approach of Section 2.

3.1 Computing $\mathfrak{R}(\mathcal{C})$

Since termination is undecidable, for automation we compute maximal elements in the set of TRSs over a given $\mathcal{C}$, for which we can show termination with reduction orders automatically. However, since there are exponentially many TRSs over $\mathcal{C}$ in general, it is impractical to check termination of each of them to compute maximal elements. We present a solution using MaxSAT solving.

In last years, SAT/SMT-encodings of termination conditions based on existing subclasses of reduction orders have been extensively investigated, and today they are well-established. Here we mention recursive path orders [17, 7], Knuth-Bendix orders [26] and orders based on matrix interpretations [9]. Importantly, all of them can test the existence of a reduction order $>$ that satisfies arbitrary Boolean combinations of order constraints:

$$C ::= s > t \mid \top \mid \bot \mid \neg\, C \mid C \vee C \mid C \wedge C$$

We exploit this fact to encode a maximal termination problem into a maximal satisfiability problem. Even though NP-hard in general, nowadays solving can be efficiently done by SMT solvers.

Computing maximal terminating TRSs is done iteratively: Given a set of equalities $\mathcal{C}$, assume we already found k maximal terminating TRSs $\mathcal{R}_1, \ldots, \mathcal{R}_k$ over $\mathcal{C}$. We construct the following optimization problem ψ:

$$\text{Maximize} \quad \bigvee_{s \approx t \in \mathcal{C}} (s > t) \oplus (t > s) \qquad \text{subject to} \quad \bigwedge_{i=1}^{k} \; \bigvee_{\ell \to r \in (\mathcal{C} \cup \mathcal{C}^{-1}) \setminus \mathcal{R}_i} \ell > r$$

where $C_1 \oplus C_2$ stands for the exclusive-or $(C_1 \wedge (\neg C_2)) \vee ((\neg C_1) \wedge C_2)$. Since each $\ell > r$ can be encoded w.r.t. a particular class of reduction orders to Boolean constraints, ψ can be treated as an instance of MaxSAT/MaxSMT. A solution yields a maximal subset of oriented equalities from $\mathcal{C}$, that forms a terminating TRS $\mathcal{R}_{k+1}$ and is different from all $\mathcal{R}_1, \ldots, \mathcal{R}_k$. If ψ is unsatisfiable, we found all maximal terminating TRSs over $\mathcal{C}$ (w.r.t. the considered reduction order) and return $\{\mathcal{R}_1, \ldots, \mathcal{R}_k\}$. Otherwise, we re-encode ψ w.r.t. $\mathcal{R}_{k+1}$ for another MaxSAT/MaxSMT-instance.

Finally, in our implementation we do not compute all maximal terminating TRSs. This is because there still may be exponentially many maximal terminating TRSs. Instead, we fix a number K to stop the enumeration of maximal terminating TRSs whenever the number reaches K. This is motivated by the following observation: Assume that there exists a complete TRS $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$, but we fail to select it. Since $\mathcal{R}$ is a terminating TRS over $\mathcal{C} \cup S(\mathcal{C})$, by Lemma 4 (a) there exists a maximal terminating, complete TRS $\mathcal{R}' \in \mathfrak{R}(\mathcal{C} \cup S(\mathcal{C}))$ with $\mathcal{R} \subseteq \mathcal{R}'$. Thus when missing the complete TRS $\mathcal{R}$ in one iteration, there is still a chance to select $\mathcal{R}'$ in the next one.

3.2 Filtering $S(\mathcal{C})$

Our implementation of the parameter function $S(\mathcal{C})$ follows closely the proposed one of Section 2 but adds a few small operations as described below.

When orienting equalities to rules, some equalities tend to generate a lot of critical pairs. This is why Knuth-Bendix completion employs selection heuristics (cf. [3, 25, 23]) that select only certain kinds of equalities. We also heuristically select equalities, since otherwise the number of critical pairs grows too fast and our implementation fails to handle it. In order to address it, we first normalize the equalities to filter out all those whose size exceeds a bound d. Then, we select n smallest equalities. We formulate this filtering. For a set of equalities $\mathcal{C}$, we write $\mathcal{C}^{<d}$ to denote all equalities $s \approx t$ of $\mathcal{C}$ with $|s| + |t| < d$. Moreover we write $\mathcal{C}{\restriction_n}$ for the set of the n smallest equalities in $\mathcal{C}$. With these notations, $S(\mathcal{C})$ can be described as follows:

$$S(\mathcal{C}) = \bigcup_{\mathcal{R} \in \mathfrak{R}(\mathcal{C})} \left(\{ s{\downarrow_\mathcal{R}} \approx t{\downarrow_\mathcal{R}} \mid s \approx t \in \mathcal{E} \cup CP(\mathcal{R}) \text{ and } s{\downarrow_\mathcal{R}} \neq t{\downarrow_\mathcal{R}} \}^{<d}{\restriction_n} \right)$$

4 Related Work and Comparison

We relate our procedure φ to existing completion methods. Due to their algorithmic nature precise simulations are difficult, but we capture their main features. We say that $\mathcal{S}$ is an *inter-reduced* version of a terminating TRS $\mathcal{R}$, if $\mathcal{S}$ is a terminating reduced TRS whose rules are obtained by rewriting rules in $\mathcal{R}$ by $\mathcal{R}$ itself.

- **Knuth-Bendix Completion** [14, 12]. Let $>$ be a reduction order for the Knuth-Bendix completion procedure and for the orientable part $\{\ell \to r \in \mathcal{C} \cup \mathcal{C}^{-1} \mid \ell > r\}$ we write $\mathcal{C}^{>}$. This procedure can be simulated by φ if one uses

  $$\mathfrak{R}(\mathcal{C}) = \{\mathcal{C}^{>}\} \qquad \text{and} \qquad S(\mathcal{C}) = \{s{\downarrow_{\mathcal{R}'}} \approx t{\downarrow_{\mathcal{R}'}}\}$$

  where, $\mathcal{R}'$ is an inter-reduced version of $\mathcal{C}^{>}$ and $s \approx t \in \mathcal{C} \cup CP(\mathcal{R}')$.

- **Multi-completion** [16]. Multi-completion uses a class of reduction orders $>_1, \ldots, >_n$ to run Knuth-Bendix completion in parallel. Typically, the class is the set of all possible recursive path orders. Its run can be simulated in our method as follows:

  $$\mathfrak{R}(\mathcal{C}) = \{\mathcal{C}^{>_1}, \ldots, \mathcal{C}^{>_n}\} \qquad \text{and} \qquad S(\mathcal{C}) = \{s{\downarrow_{\mathcal{R}'}} \approx t{\downarrow_{\mathcal{R}'}}\}$$

  where, $\mathcal{R}'$ is an inter-reduced version of $\mathcal{C}^{>_i}$ and $s \approx t \in \mathcal{C} \cup CP(\mathcal{R}')$. A naive implementation of this approach fails due to the large number of compatibility checks as well as computations of normal forms. In order to gain efficiency Kondo and Kurihara [16] provided a specialised data structure, so-called *node* for sharing these computations among the orders.

- **Completion with termination tools** [23]. This procedure does not require a reduction order as an input parameter, because during its process a necessary reduction order is constructed on the fly:

  $$\mathfrak{R}(\mathcal{C}) = \{\mathcal{R}\} \qquad \text{and} \qquad S(\mathcal{C}) = \{s{\downarrow_{\mathcal{R}'}} \approx t{\downarrow_{\mathcal{R}'}}\}$$

  where, $\mathcal{R}$ is a TRS over $\mathcal{C}$ whose termination is shown by a *termination tool*, $\mathcal{R}'$ is an inter-reduced version of $\mathcal{R}$, and $s \approx t \in \mathcal{C} \cup CP(\mathcal{R}')$. Unlike a fixed single reduction order, a termination tool can find a number of terminating TRSs over $\mathcal{C}$, which avoids failure of Knuth-Bendix completion. But its drawback is a similar problem as with multi-completion. In the paper [23] a heuristic for the *best search strategy* is suggested to select one of the terminating TRSs. This approach significantly extends the power of Knuth-Bendix completion, and has been adopted in their completion tool Slothrop.

- **Multi-completion with termination tools** [21, 25]. The method replaces reduction orders in multi-completion by a termination tool:

  $$\mathfrak{R}(\mathcal{C}) = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\} \qquad \text{and} \qquad S(\mathcal{C}) = \{s{\downarrow_{\mathcal{R}'}} \approx t{\downarrow_{\mathcal{R}'}}\}$$

  where, $\mathcal{R}_1, \ldots, \mathcal{R}_n$ are all TRS over $\mathcal{C}$ whose termination is shown by a termination tool, $\mathcal{R}'$ is an inter-reduced version of some $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$, and $s \approx t \in \mathcal{C} \cup CP(\mathcal{R}')$. A variant of the node data structure in multi-completion provides a compact representation of $\mathfrak{R}(\mathcal{C})$ as well as an efficient algorithm to compute it. This approach has been implemented in the very effective completion tool mkbTT.

As described in the earlier sections, maximal completion only computes maximal terminating TRSs, which are often much fewer than all terminating TRSs, but it does not miss a complete TRS. This is the main idea of our approach. One drawback is the current limited power of maximal termination provers. Theoretically, Brute force search allows using a termination tool to compute maximal terminating TRSs. However, it is practically infeasible due to exponentially many calls of the termination tool.

Another difference is the definition of $S(\mathcal{C})$. Except for maximal completion, all procedures use a singleton of an equality for $S(\mathcal{C})$ and its *selection* is critical for successful runs. The next example illustrates this.

▶ **Example 8.** Recall the ES $\mathcal{E}$ in Example 5:

$$1:\ \ \mathsf{s}(\mathsf{p}(x)) \approx x \qquad\qquad 2:\ \ \mathsf{p}(\mathsf{s}(x)) \approx x \qquad\qquad 3:\ \ \mathsf{s}(x) + y \approx \mathsf{s}(x + y)$$

We perform $\varphi(\mathcal{E})$ as the simulated run of multi-completion, where the class of reduction orders is all LPOs with total precedence. Assume that our selection strategy for $S(\mathcal{C})$ prefers an equality derived from the critical pair of rule 3 and the rule of the biggest possible index for some TRS in $\mathfrak{R}(\mathcal{C})$.

(i) $\mathfrak{R}(\{1,2,3\}) = \{\{1,2,3\},\{1,2,3'\}\}$, which both do not satisfy the join-condition of φ. Thus, $\varphi(\{1,2,3\}) = \varphi(\{1,\ldots,4\})$, where 4 is the single equality in $S(\{1,2,3\})$:

$$4:\ \ x + y \approx \mathsf{s}(\mathsf{p}(x) + y)$$

(ii) $\mathfrak{R}(\{1,\ldots,4\}) = \{\{1,2,3,4'\},\{1,2,3',4'\}\}$ and the join-condition does not hold again. We continue the run with $\varphi(\{1,\ldots,4\}) = \varphi(\{1,\ldots,5\})$, where 5 in $S(\{1,\ldots,4\})$ is

$$5:\ \ (x + y) + z \approx \mathsf{s}((\mathsf{p}(x) + y) + z)$$

(iii) Generally, $\mathfrak{R}(\{1,\ldots,n\}) = \{\{1,2,3,4',\ldots,n'\},\{1,2,3',4',\ldots,n'\}\}$ and $S(\{1,\ldots,n\})$ is the singleton of

$$n{+}1:\ \ ((x_1 + x_2) + \cdots) + x_{n-1} \approx \mathsf{s}(((\mathsf{p}(x_1) + x_2) + \cdots) + x_{n-1})$$

for $n \geq 3$. Thus, the join-condition never holds and the procedure does not terminate.

Admittedly, for the above example it is easy to choose an appropriate selection strategy that succeeds. In general however, it is difficult to know a suitable selection strategy a priori. This is why mkbTT provides several selection strategies as a user parameter. Maximal completion does not use a singleton but a *set* of equalities for $S(\mathcal{C})$, which reduces the risk to get stuck.

To conclude, we like to stress the simplicity of maximal completion, due to avoiding a dedicated search algorithm like one in Slothrop, and a sophisticated but complex data structure like that of multi-completion.

Table 1 Summary for all 115 equational systems

	LPO		KBO		termination tool	
	mkbTT	Maxcomp	mkbTT	Maxcomp	mkbTT	Slothrop
completed	70	**86**	67	**69**	81	71
failure	6	**6**	3	**3**	3	4
timeout	39	**23**	45	**43**	31	40

5 Experiments

We implemented maximal completion in the tool Maxcomp. The tool employs the SMT solver Yices [8] to support LPO and KBO. Concerning parameter K for $\mathfrak{R}(\mathcal{C})$ in Section 3, at the beginning we use $K = 2$ to compute two maximal terminating TRSs. During the recursion of φ, we increase K whenever $S(\mathcal{C}) \subseteq \mathcal{C}$. Moreover, we fix $n = 7$ and $d = 20$ for $S(\mathcal{C})$, simply motivated by the fact that our tool cannot process larger n and d due to the number of equalities. If, at some point, no new equalities are generated and all maximal terminating TRSs are computed (i.e. parameter K cannot be increased anymore), the tool stops with failure.

We compare Maxcomp with the two existing completion tools Slothrop and mkbTT. Since the latter two tools require termination provers, we used AProVE [10] for Slothrop, and for mkbTT its internally supplied prover $\mathsf{T_TT_2}$ [15]. The test-bed consists of 115 equational systems from the distribution of mkbTT.[1] The tests were single-threaded run on a system equipped with an Intel Core Duo L7500 with 1.6 GHz and 2 GB of RAM using a timeout of 300 seconds.

Table 1 gives a summary of the overall results. Here we also included results, where mkbTT's termination proving power was limited to LPO and KBO[2] (this is not possible for Slothrop). Aside from this, all parameters of all tools were left as default. More detailed results for a selected number of systems[3] are depicted in Table 2, where we omitted systems that could be solved by all tools with every termination method, or that could not be solved by any tool. Moreover for the two scalable systems `BGK.Dxx` and `BGK.Mxx`, that are constructed by using a natural number as an input parameter, only the largest ones considered, `BGK.D16` and `BGK.M14`, are shown here. Numbers denote execution time in seconds, $\times$ denotes the tool stopped with failure, and ∞ denotes timeout.

It should be noted that the complete systems found by using a termination tool are mostly different from those found with LPO or KBO, since the reduction order constructed using a termination tool is usually different from them. Also, for fairness it should be noted that by choosing specific, suitable selection strategies for each equational system individually, mkbTT can complete more systems than with its default selection strategy. To name one

[1] `http://cl-informatik.uibk.ac.at/software/mkbtt/`
[2] `mkbtt -s lpo` for LPO, and `mkbtt -s kbo` for KBO.
[3] The complete list is available at `http://www.jaist.ac.jp/project/maxcomp/`.

Table 2 Experimental results

	LPO		KBO		termination tool	
problem	mkbTT	Maxcomp	mkbTT	Maxcomp	mkbTT	Slothrop
BGK94.D16	∞	37.48	∞	∞	74.91	∞
BGK94.M14	2.73	∞	5.73	∞	2.06	25.88
Chr89.A3	40.06	3.17	20.85	∞	∞	∞
fib	0.87	100.32	∞	∞	0.85	7.18
Les83.fib	0.10	0.01	∞	∞	0.23	2.82
Les83.subset	0.22	0.01	∞	∞	0.09	3.27
OKW95.dt1.theory	0.78	46.15	∞	∞	0.91	6.22
rl.theory	2.36	0.17	∞	1.21	1.76	8.07
SK90.3.04	6.72	1.75	∞	∞	99.98	∞
SK90.3.05	1.58	0.35	1.07	0.33	3.46	∞
SK90.3.06	4.22	0.90	∞	∞	∞	∞
SK90.3.07	∞	3.03	∞	∞	∞	∞
SK90.3.15	∞	∞	0.12	0.02	0.08	1.39
SK90.3.18	0.15	0.01	∞	∞	0.30	3.50
SK90.3.22	∞	4.03	∞	8.03	∞	∞
SK90.3.27	13.31	21.09	36.47	0.53	72.31	∞
SK90.3.28	∞	20.23	53.58	∞	∞	110.76
slothrop.ackermann	0.02	0.01	∞	∞	0.03	0.68
slothrop.cge	∞	∞	∞	∞	173.03	∞
slothrop.endo	∞	0.62	∞	0.38	3.88	7.60
slothrop.equiv.proofs	×	×	∞	∞	2.52	262.32
slothrop.equiv.proofs.or	×	×	∞	∞	2.81	∞
slothrop.groups	0.45	0.08	∞	0.08	0.54	2.22
TPDB.zantema.z115	∞	0.73	5.76	42.63	15.27	203.69
TPTP.COL060.1.theory	×	×	0.01	0.01	0.02	1.14
TPTP.GRP454.1.theory	17.13	2.65	49.63	0.14	109.87	∞
TPTP.GRP457.1.theory	17.18	2.66	49.98	1.62	112.73	∞
TPTP.GRP460.1.theory	∞	0.89	17.74	2.91	16.99	∞
TPTP.GRP463.1.theory	∞	1.76	17.80	3.26	16.91	∞
TPTP.GRP481.1.theory	∞	2.40	∞	57.20	∞	69.56
TPTP.GRP484.1.theory	∞	0.52	∞	13.52	∞	∞
TPTP.GRP487.1.theory	∞	1.00	∞	7.05	∞	×
TPTP.GRP490.1.theory	∞	0.84	168.93	13.21	∞	∞
TPTP.GRP493.1.theory	∞	5.13	40.54	2.83	∞	∞
TPTP.GRP496.1.theory	∞	0.90	∞	11.13	∞	241.10
WS06.proofreduction	∞	∞	∞	∞	162.66	∞

example, mkbTT completes SK90.3.22 with LPO and selection strategy `old` [25] within roughly 40 seconds, however times out after 300 seconds with all other predefined strategies (`max`, `slothrop`, `sum`). While the chosen selection strategy vastly affects the outcome of mkbTT, it is in general non-trivial to decide which selection strategy to choose in advance. Lastly, concerning the timeout, with very few exceptions, a higher timeout seems not to affect the results. For overall performance, whenever all tools succeeded, they usually (with four exceptions) did so in less than 35 seconds. For the rest of systems, timing values do not show a clear trend. The parameter K mostly remains unchanged at 2, and for the vast majority of successful runs does not exceed 5, the maximum being 14.

6 Conclusion and future work

We have illustrated a very compact framework for (Knuth-Bendix) completion and demonstrated how to effectively automate it by employing modern MaxSAT solvers. Despite relying on (not very powerful) simplification orders to show termination in our implementation, experimental results indicate practical viability and, within the test-scenario as indicated here, competitiveness compared to all other known approaches. To conclude, we mention potential future work:

- Recapitulating termination techniques into maximal termination is crucial to extend the capability of maximal completion. We plan to implement matrix interpretations [9], which are a recently emerged very powerful class of reduction orders: For instance, a lexicographic combination of matrix interpretations can prove termination of a complete TRS for the system CGE2 [23]. Moreover, adaptation of the dependency pair method [2] and semantic labeling [27] for maximal termination is an important challenge.
- In order to achieve better scalability the growth of $S(\mathcal{C})$ in the iteration of φ needs to be limited. In Knuth-Bendix completion and its variants, inter-reduction is employed as well as the application of various critical pair criteria ([4, 13]). We expect that these techniques can be adapted in our framework.
- Last but not least, application to theorem proving is an important direction. There are variations of Knuth-Bendix completion including unfailing completion [5] and rewriting induction [20, 1], which are very successfully employed in powerful theorem provers like Waldmeister [18]. However, these variations require a fixed reduction order. Very recently Winkler and Middeldorp [24] adapted multi-completion with termination tools for unfailing completion to overcome this restriction. We anticipate that our approach can be integrated in these settings.

Acknowledgements We thank Sarah Winkler and Aart Middeldorp for their valuable comments.

References

1 T. Aoto. Dealing with non-orientable equations in rewriting induction. In *RTA 2006*, volume 4098 of *LNCS*, pages 242–256, 2006.
2 T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1-2):133–178, 2000.
3 F. Baader and T. Nipkow. *Term rewriting and all that.* Cambridge University Press, 1998.
4 L. Bachmair and N. Dershowitz. Critical pair criteria for completion. *Journal of Symbolic Computation*, 6(1):1–18, 1988.

5 L. Bachmair, N. Dershowitz, and J. Hsiang. Orderings for equational proofs. In *LICS*, pages 346–357. IEEE Computer Society, 1986.

6 L. Bachmair, N. Dershowitz, and D. A. Plaisted. *Resolution of Equations in Algebraic Structures: Completion without Failure*, volume 2, pages 1–30. Academic Press, 1989.

7 M. Codish, V. Lagoon, and P. Stuckey. Solving partial order constraints for LPO termination. In *RTA 2006*, volume 4098 of *LNCS*, pages 4–18, 2006.

8 B. Dutertre and L. D. Moura. A fast linear-arithmetic solver for dpll(t). In *CAV*, pages 81–94, 2006.

9 J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. *Journal of Automated Reasoning*, 40(2-3):195–220, 2008.

10 J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In *IJCAR 2006*, volume 4130 of *LNAI*, pages 281–286, 2006.

11 G. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, 1980.

12 G. Huet. A complete proof of correctness of the Knuth-Bendix completion algorithm. *Journal of Computer and System Sciences*, 21(1):11–21, 1981.

13 D. Kapur, D. R. Musser, and P. Narendran. Only prime superpositions need be considered in the Knuth-Bendix completion procedure. *Journal of Symbolic Computation*, 6(1):19–36, 1988.

14 D. Knuth and P. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. 1970.

15 M. Korp, C. Sternagel, H. Zankl, and A. Middeldorp. Tyrolean termination tool 2. In *RTA 2009*, volume 5595 of *LNCS*, pages 295–304, 2009.

16 M. Kurihara and H. Kondo. Completion for multiple reduction orderings. *Journal of Automated Reasoning*, 23(1):25–42, 1999.

17 M. Kurihara and H. Kondo. Efficient BDD encodings for partial order constraints with application to expert systems in software verification. In *IEA/AEI*, volume 3029 of *LNAI*, pages 827–837, 2004.

18 B. Löchner and T. Hillenbrand. A phytography of Waldmeister. *AI Communications*, 15(2–3):127–133, 2002.

19 Y. Métivier. About the rewriting systems produced by the Knuth-Bendix completion algorithm. *Information Processing Letters*, 16(1):31–34, 1983.

20 U. S. Reddy. Term rewriting induction. In *CADE 1990*, volume 449 of *LNCS*, pages 162–177, 1990.

21 H. Sato, S. Winkler, M. Kurihara, and A. Middeldorp. Multi-completion with termination tools (system description). In *IJCAR 2008*, LNCS, pages 306–312, 2008.

22 TeReSe. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

23 I. Wehrman, A. Stump, and E. M. Westbrook. Slothrop: Knuth-Bendix completion with a modern termination checker. In *RTA 2006*, LNCS, pages 287–296, 2006.

24 S. Winkler and A. Middeldorp. Termination tools in ordered completion. In *IJCAR 2010*, volume 6173 of *LNAI*, pages 518–532, 2010.

25 S. Winkler, H. Sato, A. Middeldorp, and M. Kurihara. Optimizing mkbtt (system description). In *RTA 2010*, LIPIcs, 2010.

26 H. Zankl, N. Hirokawa, and A. Middeldorp. KBO orientability. *Journal of Automated Reasoning*, 43(2):173–201, 2009.

27 H. Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24:89–105, 1995.