Termination Tools in Ordered Completion*

Sarah Winkler and Aart Middeldorp

Institute of Computer Science, University of Innsbruck, Austria

Abstract. Ordered completion is one of the most frequently used calculi in equational theorem proving. The performance of an ordered completion run strongly depends on the reduction order supplied as input. This paper describes how termination tools can replace fixed reduction orders in ordered completion procedures, thus allowing for a novel degree of automation. Our method can be combined with the multi-completion approach proposed by Kondo and Kurihara. We present experimental results obtained with our ordered completion tool $\mathsf{omkb}_{\mathsf{TT}}$ for both ordered completion and equational theorem proving.

1 Introduction

Unfailing completion introduced by Bachmair, Dershowitz and Plaisted [2] aims to transform a set of equations into a ground-confluent and terminating system. Underlying many completion-based theorem proving systems, it has become a well-known calculus in automated reasoning. In contrast to standard completion [7], *ordered completion*, as it is called nowadays, always succeeds (in theory). The reduction order supplied as input is nevertheless a critical parameter when it comes to performance issues.

With *multi-completion*, Kondo and Kurihara [9] proposed a completion calculus that employs multiple reduction orders in parallel. It is applicable to both standard and ordered completion, and more efficient than a parallel execution of the respective processes. Wehrman, Stump and Westbrook [16] introduced a variant of standard completion that utilizes a termination prover instead of a fixed reduction order. The tool Slothrop demonstrates the potential of this approach by completing systems that cannot be handled by traditional completion procedures. In [11] it was shown how multi-completion and the use of termination tools can be combined. When implemented in the tool $\mathsf{mkb}_{\mathsf{TT}}$, this approach could cope with input systems that were not completed by Slothrop.

The current paper describes how termination tools can replace reduction orders in ordered completion procedures. In contrast to standard completion using termination provers, two challenges have to be faced. First of all, ordered completion procedures require the termination order to be totalizable for the theory. When using termination tools, the order which is synthesized during the

* This research is supported by FWF (Austrian Science Fund) project P18763. The first author is supported by a DOC-fFORTE fellowship of the Austrian Academy of Sciences.

termination proving process need not have this property. Second, the standard notion of fairness, which determines which (extended) critical pairs need to be computed to ensure correctness, depends on the (final) reduction order, which is not known in advance. We explain how to overcome these challenges, also in a multi-completion setting. We further show how ordered multi-completion with termination tools can be used for equational theorem proving.

The remainder of the paper is organized as follows. Section 2 summarizes definitions, inference systems and main results in the context of (ordered) completion which will be needed in the sequel. Section 3 describes the calculus oKBtt for ordered completion with termination tools. The results obtained in Section 3 are extended to oMKBtt, a calculus for ordered multi-completion with termination tools, in Section 4. More application-specific, we outline in Section 5 how oMKBtt can be used for refutational theorem proving. In Section 6 we briefly describe our tool $\mathsf{omkb}_{\mathsf{TT}}$ that implements the calculus oMKBtt. Experimental results are given in Section 7 before we add some concluding remarks in Section 8. Further information and detailled experimental results can be obtained from the accompanying website.[1]

2 Preliminaries

We consider terms $\mathcal{T}(\mathcal{F}, \mathcal{V})$ over a finite signature $\mathcal{F}$ and a set of variables $\mathcal{V}$. Terms without variables are *ground*. Sets of equations between terms will be denoted by $\mathcal{E}$ and are assumed to be symmetric. The associated *equational theory* is denoted by $\approx_{\mathcal{E}}$. As usual a set of directed equations $l \to r$ is called a rewrite system and denoted by $\mathcal{R}$, and $\to_{\mathcal{R}}$ is the associated *rewrite relation*. We write $s \xrightarrow{l \to r}_p t$ to express that $s \to_{\mathcal{R}} t$ was achieved by applying the rule $l \to r \in \mathcal{R}$ at position p. The relations $\to_{\mathcal{R}}^+$, $\to_{\mathcal{R}}^*$ and $\leftrightarrow_{\mathcal{R}}$ denote the transitive, transitive-reflexive and symmetric closure of $\to_{\mathcal{R}}$. The smallest equivalence relation containing $\to_{\mathcal{R}}$, which coincides with the equational theory $\approx_{\mathcal{R}}$ if $\mathcal{R}$ is considered as a set of equations, is denoted by $\leftrightarrow_{\mathcal{R}}^*$. Subscripts are omitted if the rewrite system or the set of equations is clear from the context.

A rewrite system $\mathcal{R}$ is *terminating* if it does not admit infinite rewrite sequences. It is *confluent* if for every peak $t \,{}^*\!\leftarrow s \to^* u$ there exists a term v such that $t \to^* v \,{}^*\!\leftarrow u$. $\mathcal{R}$ is *ground-confluent* if this property holds for all ground terms s. A rewrite system $\mathcal{R}$ with the property that for every rewrite rule $l \to r$ the right-hand side r is in normal form and the left-hand side l is in normal form with respect to $\mathcal{R} \setminus \{l \to r\}$ is called *reduced*. A rewrite system which is both terminating and (ground-)confluent is called (ground-)*complete*. We call $\mathcal{R}$ *complete for* a set of equations $\mathcal{E}$ if $\mathcal{R}$ is complete and $\leftrightarrow_{\mathcal{R}}^*$ coincides with $\approx_{\mathcal{E}}$.

A proper order $\succ$ on terms is a *rewrite order* if it is closed under contexts and substitutions. A well-founded rewrite order is called a *reduction order*. The relation $\to_{\mathcal{R}}^+$ is a reduction order for every terminating rewrite system $\mathcal{R}$. A reduction order $\succ$ is *complete for* a set of equations $\mathcal{E}$ if $s \succ t$ or $t \succ s$ holds

[1] See `http://cl-informatik.uibk.ac.at/users/swinkler/omkbtt`.

deduce_2	$\dfrac{\mathcal{E},\mathcal{R}}{\mathcal{E}\cup\{s\approx t\},\mathcal{R}}$	if $s \xleftarrow{r_1\leftarrow l_1} u \xrightarrow{l_2\rightarrow r_2} t$ where $l_1\approx r_1, l_2\approx r_2 \in \mathcal{R}\cup\mathcal{E}$ and $r_i \not\succ l_i$
$\mathsf{simplify}_2$	$\dfrac{\mathcal{E}\cup\{s\approx t\},\mathcal{R}}{\mathcal{E}\cup\{s\approx u\},\mathcal{R}}$	if $t \xrightarrow{l\sigma\rightarrow r\sigma} u$ using $l\approx r\in\mathcal{E}$ where $t \rhd l^a$ and $l\sigma\succ r\sigma$
$\mathsf{compose}_2$	$\dfrac{\mathcal{E},\mathcal{R}\cup\{s\rightarrow t\}}{\mathcal{E},\mathcal{R}\cup\{s\rightarrow u\}}$	if $t \xrightarrow{l\sigma\rightarrow r\sigma} u$ using $l\approx r\in\mathcal{E}$ and $l\sigma\succ r\sigma$
$\mathsf{collapse}_2$	$\dfrac{\mathcal{E},\mathcal{R}\cup\{t\rightarrow s\}}{\mathcal{E}\cup\{u\approx s\},\mathcal{R}}$	if $t \xrightarrow{l\sigma\rightarrow r\sigma} u$ using $l\approx r\in\mathcal{E}$ where $t \rhd l$ and $l\sigma\succ r\sigma$

[a] $\rhd$ denotes the strict encompassment relation

Fig. 1. Additional inference rules for ordered completion (oKB).

for all ground terms s and t that satisfy $s \approx_{\mathcal{E}} t$. In the sequel we will consider lexicographic path orders (LPO [6]), Knuth-Bendix orders (KBO [7]), multiset path orders (MPO [4]) and orders induced by polynomial interpretations [10]. The first two are total on ground terms if the associated precedence is total. Orders induced by MPOs and polynomial interpretations can always be extended to an order with that property. Reduction orders that are total on ground terms are of course complete for any theory.

2.1 Ordered Completion

We assume that the reader is familiar with standard completion, originally proposed by Knuth and Bendix [7] and later on formulated as an inference system [1]. This inference system will in the sequel be referred to as KB.

For ordered completion (oKB) [2] the inference system of standard completion is extended with the rules depicted in Fig. 1, where $\succ$ denotes the reduction order used.

An inference sequence $(\mathcal{E}_0,\mathcal{R}_0)\vdash(\mathcal{E}_1,\mathcal{R}_1)\vdash(\mathcal{E}_2,\mathcal{R}_2)\ldots$ is called a *deduction* with *persistent* equalities $\mathcal{E}_\omega = \bigcup_i\bigcap_{j>i}\mathcal{E}_j$ and rules $\mathcal{R}_\omega = \bigcup_i\bigcap_{j>i}\mathcal{R}_j$.

Definition 1. *An equation $s\approx t$ is an* extended critical pair *with respect to a set of equations $\mathcal{E}$ and a reduction order $\succ$ if there are a term u and rewrite steps $u \xrightarrow{l_1\sigma\rightarrow r_1\sigma}_\epsilon s$ and $u \xrightarrow{l_2\sigma\rightarrow r_2\sigma}_p t$ such that $l_1\approx r_1, l_2\approx r_2\in\mathcal{E}$ and $r_i\sigma\not\succ l_i\sigma$. The set of extended critical pairs among equations in $\mathcal{E}$ is denoted by $CP_\succ(\mathcal{E})$.*

An oKB deduction is *fair* if $\mathrm{CP}_\succ(\mathcal{E}_\omega\cup\mathcal{R}_\omega)\subseteq\bigcup_i\mathcal{E}_i$. The following theorems from [2] state the correctness and completeness of oKB.

Theorem 2. *Let $\mathcal{E}$ be a set of equations and $\succ$ a reduction order that can be extended to a reduction order $>$ which is complete for $\mathcal{E}$. Any fair oKB run will*

$$\textsf{orient}\quad \frac{\mathcal{E} \cup \{s \approx t\}, \mathcal{R}, \mathcal{C}}{\mathcal{E}, \mathcal{R} \cup \{s \to t\}, \mathcal{C} \cup \{s \to t\}} \quad \text{if } \mathcal{C} \cup \{s \to t\} \text{ terminates}$$

Fig. 2. The orient inference rule in KBtt.

on inputs $(\mathcal{E}, \varnothing)$ *and* $\succ$ *generate a system* $\mathcal{E}_\omega \cup \mathcal{R}_\omega$ *that is ground-complete with respect to* $>$.

An oKB completion procedure is *simplifying* if for all inputs $\mathcal{E}_0$ and $\succ$ the rewrite system $\mathcal{R}_\omega$ is reduced and all equations $u \approx v$ in $\mathcal{E}_\omega$ are both unorientable with respect to $\succ$ and irreducible in $\mathcal{R}_\omega$.

Theorem 3. *Assume* $\mathcal{R}$ *is a reduced and complete rewrite system for* $\mathcal{E}$ *that is contained in a reduction order* $\succ$ *which can be extended to a complete reduction order for* $\mathcal{E}$. *Any fair and simplifying oKB run that starts from* $(\mathcal{E}, \varnothing)$ *using* $\succ$ *yields* $\mathcal{E}_\omega = \varnothing$ *and* $\mathcal{R}_\omega = \mathcal{R}$.

In the requirement for a reduction order that is totalizable for the theory, ordered completion differs from standard completion. The more recent approach of Bofill *et al.* [3] lacks this restriction, but the obtained completion procedure is only of theoretical interest as it relies on enumerating all ground equational consequences of the theory $\mathcal{E}$.

2.2 Completion with Termination Tools

The inference system KBtt [16] for standard completion with termination tools operates on tuples $(\mathcal{E}, \mathcal{R}, \mathcal{C})$ consisting of a set of equations $\mathcal{E}$, and rewrite systems $\mathcal{R}$ and $\mathcal{C}$. The latter is called the *constraint system*. KBtt consists of the orient rule depicted in Fig. 2 together with the remaining KB rules where the constraint component is not modified.

Correctness and completeness of KBtt follow from the fact that any run of standard completion can be simulated by KB and vice versa [16].

2.3 Completion with Multiple Reduction Orders

Multi-completion (MKB), introduced by Kurihara and Kondo [9] considers a set of reduction orders $\mathcal{O} = \{\succ_1, \ldots, \succ_n\}$. To share inferences for different orders, a special data structure is used.

Definition 4. *A* node *is a tuple* $\langle s : t, R_0, R_1, E\rangle$ *where the* data s, t *are terms and the* labels R_0, R_1, E *are subsets of* $\mathcal{O}$ *such that* R_0, R_1 *and* E *are mutually disjoint,* $s \succ_i t$ *for all* $\succ_i \in R_0$, *and* $t \succ_i s$ *for all* $\succ_i \in R_1$.

The intuition is that given a node $\langle s : t, R_0, R_1, E\rangle$, all orders in the *equation label* E consider the data as an equation $s \approx t$ while orders in the *rewrite labels*

$$\textsf{orient}\quad \frac{\mathcal{N} \cup \{\langle s : t, R_0, R_1, E \uplus R\rangle\}}{\mathcal{N} \cup \{\langle s : t, R_0 \cup R, R_1, E\rangle\}} \quad \text{if } R \neq \varnothing \text{ and } s \succ_i t \text{ for all } \succ_i \in R$$

Fig. 3. orient in MKB.

R_0 and R_1 regard it as rewrite rules $s \to t$ and $t \to s$, respectively. Hence $\langle s : t, R_0, R_1, E\rangle$ is identified with $\langle t : s, R_1, R_0, E\rangle$.

MKB is described by an inference system consisting of five rules. Fig. 3 shows the orient inference rule. As shown in [9], slight modifications to the rewrite inference rules allow to perform ordered multi-completion (oMKB).

3 Ordered Completion with Termination Tools

This section describes how the ideas of KBtt can be incorporated into ordered completion procedures. The derived method will in the sequel be referred to as oKBtt. It is described by an inference system consisting of the rules depicted in Fig. 4 together with orient, delete, simplify, compose and collapse from KBtt.

$$\begin{array}{lll}
\textsf{deduce}_2 & \dfrac{\mathcal{E}, \mathcal{R}, \mathcal{C}}{\mathcal{E} \cup \{s \approx t\}, \mathcal{R}, \mathcal{C}} & \text{if } s \leftarrow_{\mathcal{E}\cup\mathcal{R}} u \rightarrow_{\mathcal{E}\cup\mathcal{R}} t \\[2ex]
\textsf{simplify}_2 & \dfrac{\mathcal{E} \cup \{s \approx t\}, \mathcal{R}, \mathcal{C}}{\mathcal{E} \cup \{s \approx u\}, \mathcal{R}, \mathcal{C} \cup \{l\sigma \to r\sigma\}} & \text{if } t \xrightarrow{l\sigma\to r\sigma} u \text{ using } l \approx r \in \mathcal{E} \text{ where} \\
 & & t \rhd l \text{ and } \mathcal{C} \cup \{l\sigma \to r\sigma\} \text{ terminates} \\[2ex]
\textsf{compose}_2 & \dfrac{\mathcal{E}, \mathcal{R} \cup \{s \to t\}, \mathcal{C}}{\mathcal{E}, \mathcal{R} \cup \{s \to u\}, \mathcal{C} \cup \{l\sigma \to r\sigma\}} & \text{if } t \xrightarrow{l\sigma\to r\sigma} u \text{ using } l \approx r \in \mathcal{E} \text{ and} \\
 & & \mathcal{C} \cup \{l\sigma \to r\sigma\} \text{ terminates} \\[2ex]
\textsf{collapse}_2 & \dfrac{\mathcal{E}, \mathcal{R} \cup \{t \to s\}, \mathcal{C}}{\mathcal{E} \cup \{u \approx s\}, \mathcal{R}, \mathcal{C} \cup \{l\sigma \to r\sigma\}} & \text{if } t \xrightarrow{l\sigma\to r\sigma} u \text{ using } l \approx r \in \mathcal{E} \text{ where} \\
 & & t \rhd l \text{ and } \mathcal{C} \cup \{l\sigma \to r\sigma\} \text{ terminates}
\end{array}$$

Fig. 4. Ordered completion with termination tools (oKBtt).

An inference sequence $(\mathcal{E}_0, \mathcal{R}_0, \mathcal{C}_0) \vdash (\mathcal{E}_1, \mathcal{R}_1, \mathcal{C}_1) \vdash (\mathcal{E}_2, \mathcal{R}_2, \mathcal{C}_2) \vdash \cdots$ with respect to oKBtt is called an oKBtt run and denoted by γ. Persistent equations $\mathcal{E}_\omega$ and rules $\mathcal{R}_\omega$ are defined as for oKB. The set $\mathcal{C}_\omega = \bigcup_i \mathcal{C}_i$ collects persistent constraint rules. We write $(\mathcal{E}_0, \mathcal{R}_0) \vdash^* (\mathcal{E}_\alpha, \mathcal{R}_\alpha)$ to express that the run has length α, where $\alpha = \omega$ if it is not finite.

Example 5. If oKBtt is run on the input equations $\mathsf{g}(\mathsf{f}(x, \mathsf{b})) \approx \mathsf{a}$ and $\mathsf{f}(\mathsf{g}(x), y) \approx \mathsf{f}(x, \mathsf{g}(y))$ and all termination checks are performed with respect to the polynomial interpretation $[\mathsf{f}](x, y) = x + 2y + 1$, $[\mathsf{g}](x) = x + 1$ and $[\mathsf{a}] = [\mathsf{b}] = [\mathsf{c}] = 0$,

the following system is derived:

$$\mathcal{E} = \left\{ \begin{array}{c} \mathsf{f}(\mathsf{f}(x,\mathsf{b}),\mathsf{a}) \approx \mathsf{f}(\mathsf{c},\mathsf{f}(y,\mathsf{b})) \\ \mathsf{f}(\mathsf{f}(x,\mathsf{b}),\mathsf{a}) \approx \mathsf{f}(\mathsf{f}(y,\mathsf{b}),\mathsf{a}) \\ \mathsf{f}(\mathsf{c},\mathsf{f}(x,\mathsf{b})) \approx \mathsf{f}(\mathsf{c},\mathsf{f}(y,\mathsf{b})) \end{array} \right\} \quad \mathcal{R} = \left\{ \begin{array}{c} \mathsf{g}(\mathsf{f}(x,\mathsf{b})) \to \mathsf{a} \\ \mathsf{f}(x,\mathsf{g}(y)) \to \mathsf{f}(\mathsf{g}(x),y) \\ \mathsf{f}(\mathsf{g}(x),\mathsf{f}(y,\mathsf{b})) \to \mathsf{f}(x,\mathsf{c}) \end{array} \right\}$$

However, if the second equation would be oriented from left to right, the oKBtt run diverges. Since $\mathsf{f}(x,\mathsf{g}(y)) \to \mathsf{f}(\mathsf{g}(x),y)$ cannot be oriented by any KBO or LPO which compares lists of subterms only from left to right, ordered completion tools that do not support other termination methods (e.g. Waldmeister) cannot derive a ground-complete system.

Before showing that oKBtt runs can be simulated by ordered completion runs, and vice versa, we note that oKBtt is sound in that it does not change the equational theory.

Lemma 6. *For every oKBtt step $(\mathcal{E},\mathcal{R},\mathcal{C}) \vdash (\mathcal{E}',\mathcal{R}',\mathcal{C}')$ the relations $\leftrightarrow^*_{\mathcal{E}\cup\mathcal{R}}$ and $\leftrightarrow^*_{\mathcal{E}'\cup\mathcal{R}'}$ coincide.*

Lemma 7. *For every finite oKBtt run $(\mathcal{E}_0,\mathcal{R}_0,\mathcal{C}_0) \vdash^* (\mathcal{E}_n,\mathcal{R}_n,\mathcal{C}_n)$ such that $\mathcal{R}_0 \subseteq \to^+_{\mathcal{C}_0}$, there is a corresponding oKB run $(\mathcal{E}_0,\mathcal{R}_0) \vdash^* (\mathcal{E}_n,\mathcal{R}_n)$ using the reduction order $\to^+_{\mathcal{C}_n}$.*

Proof. Let $\succ_n$ denote $\to^+_{\mathcal{C}_n}$. We use induction on n. The claim is trivially true for $n = 0$. For a run of the form $(\mathcal{E}_0,\mathcal{R}_0,\mathcal{C}_0) \vdash^* (\mathcal{E}_n,\mathcal{R}_n,\mathcal{C}_n) \vdash (\mathcal{E}_{n+1},\mathcal{R}_{n+1},\mathcal{C}_{n+1})$, the induction hypothesis yields a corresponding oKB run $(\mathcal{E}_0,\mathcal{R}_0) \vdash^* (\mathcal{E}_n,\mathcal{R}_n)$ using the reduction order $\succ_n$. Since constraint rules are never removed we have $\mathcal{C}_k \subseteq \mathcal{C}_{n+1}$ for all $k \leq n$, so the same run can be obtained with $\succ_{n+1}$. Case distinction on the applied oKBtt rule shows that a step $(\mathcal{E}_n,\mathcal{R}_n) \vdash (\mathcal{E}_{n+1},\mathcal{R}_{n+1})$ using $\succ_{n+1}$ is possible.

If orient added the rule $s \to t$ then $s \succ_{n+1} t$ holds by definition, so oKB can apply orient as well. In case $\mathsf{simplify}_2$, $\mathsf{compose}_2$ or $\mathsf{collapse}_2$ was applied using an instance $l\sigma \to r\sigma$ of an equation in $\mathcal{E}_n$, we have $l\sigma \succ_{n+1} r\sigma$ by definition of the inference rules, hence the respective oKB step can be applied. Clearly, in the remaining cases the inference step can be simulated by the corresponding oKB rule since no conditions on the order are involved. □

Lemma 7 does not generalize to infinite runs; as also remarked in [16], $\to^+_{\mathcal{C}_\omega}$ is not necessarily a reduction order since an infinite union of terminating rewrite systems need not be terminating.

Simulating oKB by oKBtt is also complete as stated below.

Lemma 8. *For every oKB run $(\mathcal{E}_0,\mathcal{R}_0) \vdash^* (\mathcal{E}_\alpha,\mathcal{R}_\alpha)$ of length $\alpha \leq \omega$ using a reduction order $\succ$, there exists an oKBtt run $(\mathcal{E}_0,\mathcal{R}_0,\mathcal{C}_0) \vdash^* (\mathcal{E}_\alpha,\mathcal{R}_\alpha,\mathcal{C}_\alpha)$ such that $\mathcal{C}_\alpha \subseteq \succ$ holds.*

Proof. By induction on α. For $\alpha = 0$ the claim is trivially satisfied by setting $\mathcal{C}_0 = \mathcal{R}_0$. If $\alpha = n+1$ then the induction hypothesis yields a oKBtt run $(\mathcal{E}_0,\mathcal{R}_0,\mathcal{C}_0) \vdash^*$

$(\mathcal{E}_n, \mathcal{R}_n, \mathcal{C}_n)$ such that $\mathcal{C}_n \subseteq \succ$ An easy case distinction on the last inference step $(\mathcal{E}_n, \mathcal{R}_n) \vdash (\mathcal{E}_{n+1}, \mathcal{R}_{n+1})$ shows that using $\succ$ for termination checks allows for a corresponding oKBtt step.

If the applied inference rule is orient then $\mathcal{E}_n = \mathcal{E}_{n+1} \cup \{s \approx t\}$, $\mathcal{R}_{n+1} = \mathcal{E}_n \cup \{s \to t\}$ and $s \succ t$. Thus also $\mathcal{C}_n \cup \{s \to t\} \subseteq \succ$, ensuring termination of the extended constraint system. Hence the oKBtt inference orient can be applied to obtain $(\mathcal{E}_n, \mathcal{R}_n, \mathcal{C}_n) \vdash (\mathcal{E}_n \setminus \{s \approx t\}, \mathcal{R}_n \cup \{s \to t\}, \mathcal{C}_n \cup \{s \to t\})$. If the inference step $(\mathcal{E}_n, \mathcal{R}_n) \vdash (\mathcal{E}_{n+1}, \mathcal{R}_{n+1})$ applies $\mathsf{compose}_2$, $\mathsf{simplify}_2$ or $\mathsf{collapse}_2$ then an equation $l \approx r$ is used with a substitution σ such that $l\sigma \succ r\sigma$. Thus also $\mathcal{C}_n \cup \{l\sigma \to r\sigma\} \subseteq \succ$, ensuring termination of the extended constraint system such that the respective oKBtt rule is applicable with $\mathcal{C}_{n+1} = \mathcal{C}_n \cup \{l\sigma \to r\sigma\}$. In the remaining cases one can set $\mathcal{C}_{n+1} = \mathcal{C}_n$ and replace the applied rule by the respective oKBtt counterpart since no conditions on the order are involved.

If $\alpha = \omega$ then, by the induction hypothesis, for all runs $(\mathcal{E}_0, \mathcal{R}_0) \vdash^* (\mathcal{E}_n, \mathcal{R}_n)$ with $n < \omega$ we have $(\mathcal{E}_0, \mathcal{R}_0, \mathcal{C}_0) \vdash^* (\mathcal{E}_n, \mathcal{R}_n, \mathcal{C}_n)$ and $\mathcal{C}_n \subseteq \succ$. Since the definitions of $\mathcal{E}_\omega$ and $\mathcal{R}_\omega$ in oKB and oKBtt coincide, for $\mathcal{C}_\omega = \bigcup_i \bigcap_{j>i} \mathcal{C}_j$ the sequence $(\mathcal{E}_0, \mathcal{R}_0, \mathcal{C}_0) \vdash^* (\mathcal{E}_\omega, \mathcal{R}_\omega, \mathcal{C}_\omega)$ is a valid oKBtt run and $\mathcal{C}_\omega \subseteq \succ$. □

Totalizability

Lemma 7 shows that an oKBtt run resulting in the final constraint system $\mathcal{C}$ can be simulated by ordered completion using the reduction order $\to_{\mathcal{C}}^+$. If this order should play the role of $\succ$ in Theorem 2 then it has to be contained in a reduction order $>$ which is complete for the theory. Unfortunately, such an order does not always exist. In the proof of the extended critical pair lemma [2], totalizability of the reduction order is needed to guarantee joinability of variable overlaps. Thus, if an oKBtt procedure outputs $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{C}$ such that $\to_{\mathcal{C}}^+$ cannot be extended to a complete order for the theory, ground-confluence of $(\mathcal{E}, \mathcal{R})$ is not guaranteed.

Example 9. A fair oKBtt run starting from

$$\mathcal{E}_0 = \left\{ \begin{array}{cc} \mathsf{f}(\mathsf{a}+\mathsf{c}) \approx \mathsf{f}(\mathsf{c}+\mathsf{a}) & \mathsf{a} \approx \mathsf{b} \\ \mathsf{g}(\mathsf{c}+\mathsf{b}) \approx \mathsf{g}(\mathsf{b}+\mathsf{c}) & x+y \approx y+x \end{array} \right\}$$

might produce the following result:

$$\mathcal{E} = \{x+y \approx y+x\} \qquad \mathcal{R} = \left\{ \begin{array}{c} \mathsf{a} \to \mathsf{b} \\ \mathsf{f}(\mathsf{b}+\mathsf{c}) \to \mathsf{f}(\mathsf{c}+\mathsf{b}) \\ \mathsf{g}(\mathsf{c}+\mathsf{b}) \to \mathsf{g}(\mathsf{b}+\mathsf{c}) \end{array} \right\}$$

with $\mathcal{C} = \mathcal{R} \cup \{\mathsf{f}(\mathsf{a}+\mathsf{c}) \to \mathsf{f}(\mathsf{c}+\mathsf{a})\}$. No reduction order $>$ extending $\to_{\mathcal{C}}^+$ can orient the ground instance $\mathsf{c}+\mathsf{a} \approx \mathsf{a}+\mathsf{c}$ from left to right. So $\mathsf{a}+\mathsf{c} > \mathsf{c}+\mathsf{a}$ must hold. This gives rise to the variable overlap $\mathsf{b}+\mathsf{c} \leftarrow \mathsf{a}+\mathsf{c} \to \mathsf{c}+\mathsf{a} \to \mathsf{c}+\mathsf{b}$. As $\mathsf{b}+\mathsf{c}$ and $\mathsf{c}+\mathsf{b}$ have to be incomparable in $>$ the overlap is not joinable.

To solve this problem we restrict the termination checks in oKBtt inferences.

Definition 10. *An* $\mathsf{oKBtt}_{\mathcal{P}}$ procedure *refers to any program which implements the inference rules of* oKBtt *and employs the termination strategy* $\mathcal{P}$ *for termination checks in* orient, $\mathsf{simplify}_2$, $\mathsf{compose}_2$ *and* $\mathsf{collapse}_2$ *inferences. An* $\mathsf{oKBtt}_{\mathsf{total}}$ *procedure is an* $\mathsf{oKBtt}_{\mathcal{P}}$ *procedure where* $\mathcal{P}$ *ensures* total *termination [15, Section 6.3.2] of the checked system.*

Examples of such termination strategies are LPO, KBO and MPO with total precedences as well as polynomial interpretations over $\mathbb{N}$.

Thus, for any constraint system $\mathcal{C}_n$ derived by an $\mathsf{oKBtt}_{\mathsf{total}}$ procedure in finitely many steps, there is a reduction order $>$ extending $\to^+_{\mathcal{C}_n}$ which is total on ground terms.

Fairness

Theorem 2 requires a run to be *fair*, meaning that all extended critical pairs among persistent equations and rules are considered. In the context of oKBtt, the set of extended critical pairs cannot be computed during a run since the reduction order $\to^+_{\mathcal{C}}$ is not known in advance.

We solve this problem by observing that any reduction order $>$ which is total on ground terms contains the embedding relation $\rhd_{\mathsf{emb}}$ [18, Proposition 2]. Since $CP_>(\mathcal{E}) \subseteq CP_\succ(\mathcal{E})$ whenever $\succ \subseteq >$, the idea is now to over-approximate $CP_>(\mathcal{E}_\omega \cup \mathcal{R}_\omega)$ by $CP_{\rhd_{\mathsf{emb}}}(\mathcal{E}_\omega \cup \mathcal{R}_\omega)$. This motivates the following definition.

Definition 11. *A run* γ *is* sufficiently fair *if* $CP_{\rhd_{\mathsf{emb}}}(\mathcal{E}_\omega \cup \mathcal{R}_\omega) \subseteq \bigcup_i \mathcal{E}_i$.

It follows that a sufficiently fair run of $\mathsf{oKBtt}_{\mathsf{total}}$ is fair with respect to (any total extension of) the final reduction order $\to^+_{\mathcal{C}}$.

3.1 Correctness and Completeness

With the above considerations, we can carry over the correctness result of ordered completion to the $\mathsf{oKBtt}_{\mathsf{total}}$ setting.

Theorem 12. *If* $(\mathcal{E}, \mathcal{R}, \mathcal{C}) \vdash^* (\mathcal{E}_n, \mathcal{R}_n, \mathcal{C}_n)$ *is a sufficiently fair, finite* $\mathsf{oKBtt}_{\mathsf{total}}$ *run with* $\mathcal{R} \subseteq \to^+_{\mathcal{C}}$ *then* $\mathcal{E}_n \cup \mathcal{R}_n$ *is ground-complete for* $\mathcal{E}$ *with respect to any reduction order* $>$ *total on ground terms that extends* $\to^+_{\mathcal{C}_n}$.

Proof. By Lemma 7, there exists a corresponding oKB run γ' using the reduction order $\to^+_{\mathcal{C}_n}$. Any reduction order $>$ which is total on ground terms contains the embedding relation. Hence $CP_>(\mathcal{E}_n \cup \mathcal{R}_n) \subseteq CP_{\rhd_{\mathsf{emb}}}(\mathcal{E}_n \cup \mathcal{R}_n)$ and as a consequence the sufficiently fair run γ' is also fair with respect to $>$. By correctness of ordered completion, $\mathcal{E}_n \cup \mathcal{R}_n$ is ground-complete for $\mathcal{E}$ with respect to $>$. □

Lemma 8 states that oKBtt is complete in that any oKB run γ can be simulated by an oKBtt run γ'. If γ is fair then also γ' is fair, although it need not be sufficiently fair. Nevertheless, sufficiently fair $\mathsf{oKBtt}_{\mathsf{total}}$ procedures are complete for deriving complete systems if additional equations are considered.

Theorem 13. *Assume $\mathcal{R}$ is a complete system for $\mathcal{E}$ and $\succ$ is a reduction order containing $\mathcal{R}$ which can be extended to a reduction order that is total on ground terms. There exists a sufficiently fair* oKBtt$_{\mathsf{total}}$ *run starting from $(\mathcal{E}, \varnothing, \varnothing)$ which produces the result $\mathcal{R}_\omega = \mathcal{R}$ and $\mathcal{E}_\omega = \varnothing$.*

Proof. According to Theorem 3, there exists an oKB run γ producing $\mathcal{R}_\omega = \mathcal{R}$ and $\mathcal{E}_\omega = \varnothing$. By Lemma 8 there is a corresponding oKBtt run $(\mathcal{E}, \varnothing, \varnothing) \vdash^* (\varnothing, \mathcal{R}, \mathcal{C})$. This run can be extended to $(\mathcal{E}, \varnothing, \varnothing) \vdash^* (\varnothing, \mathcal{R}, \mathcal{C}) \vdash^* (\mathcal{E}', \mathcal{R}, \mathcal{C})$ by deducing the remaining equations in $\mathcal{E}' = CP_{\rhd_{\mathsf{emb}}}(\mathcal{E}_\omega \cup \mathcal{R}_\omega) \setminus CP_{\succ}(\mathcal{E}_\omega \cup \mathcal{R}_\omega)$ in order to make it sufficiently fair. Since $\mathcal{R}$ is complete for $\mathcal{E}$, all equations in $\mathcal{E}'$ can be simplified to trivial ones which allows to derive the result $(\varnothing, \mathcal{R}, \mathcal{C})$. □

4 Ordered Multi-Completion with Termination Tools

Ordered multi-completion with termination tools (oMKBtt) simulates multiple oKBtt processes. Similar as in MKBtt, inference steps among these processes are shared. For this purpose, a *process* p is modeled as a bit string in $\mathcal{L}((0+1)^*)$. A set of processes P is called *well-encoded* if there are no processes $p, p' \in P$ such that p is a proper prefix of p'.

Definition 14. *An* oMKBtt node $\langle s : t, R_0, R_1, E, C_0, C_1 \rangle$ *consists of a pair of terms $s : t$ (the* data*) and well-encoded sets of processes R_0, R_1, E, C_0, C_1 (the* labels*) such that $R_0 \cup C_0$, $R_1 \cup C_1$ and E are mutually disjoint.*

The set of processes occurring in a node n and a node set $\mathcal{N}$ are denoted by $\mathcal{P}(n)$ and $\mathcal{P}(\mathcal{N})$. The *projection* of a node set $\mathcal{N}$ to a process p is defined below.

Definition 15. *Given a node $n = \langle s : t, R_0, R_1, E, C_0, C_1 \rangle$ and a process p, let P_p denote the set of prefixes of p, and set*

$$E_p(n) = \begin{cases} \{s \approx t\} & \text{if } P_p \cap E \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases} \qquad R_p(n) = \begin{cases} \{s \to t\} & \text{if } P_p \cap R_0 \neq \varnothing \\ \{t \to s\} & \text{if } P_p \cap R_1 \neq \varnothing \\ \varnothing & \text{otherwise} \end{cases}$$

The set $C_p(n)$ is defined analogous to $R_p(n)$. Furthermore, we define $E_p(\mathcal{N}) = \bigcup_{n \in \mathcal{N}} E_p(n)$, $R_p(\mathcal{N}) = \bigcup_{n \in \mathcal{N}} R_p(n)$ and $C_p(\mathcal{N}) = \bigcup_{n \in \mathcal{N}} C_p(n)$.

Note that the above projections are well-defined if all process sets in $\mathcal{N}$ are well-encoded. The inference system oMKBtt works on sets of nodes $\mathcal{N}$ and consists of the rules given in Fig. 5 together with orient, delete and (optionally) subsume and gc as defined for MKBtt [12]. To refer to process *splitting*, i.e. replacing a process by two child processes in orewrite$_1$ and orewrite$_2$, the following operations are used.

Definition 16. *For a process p, a set of processes Q and $b \in \{0, 1\}$,*

$$\mathrm{app}_p^b(Q) = \begin{cases} (Q \setminus \{p\}) \cup \{pb\} & \text{if } p \in Q \\ Q & \text{otherwise} \end{cases}$$

$$\mathsf{orewrite}_1 \quad \frac{\mathcal{N} \cup \{\langle s : t, R_0, R_1, E, C_0, C_1\rangle\}}{\begin{array}{l}\mathrm{split}_S(\mathcal{N}) \cup \{\, \langle s : t, \mathrm{app}^0_S(R_0 \setminus R), R_1, E \setminus R, \mathrm{split}_S(C_0), C_1\rangle \\ \qquad \langle s : u, \mathrm{app}^1_S(R_0 \cap (R \cup S)), \varnothing, E \cap R, \varnothing, \varnothing\rangle \\ \qquad \langle l\sigma : r\sigma, \varnothing, \varnothing, \varnothing, \{p1 \mid p \in S\}, \varnothing\rangle \,\}\end{array}}$$

if
- $\langle l : r, R, \ldots, E'', \ldots\rangle \in \mathcal{N}$ and $t \xrightarrow{l\sigma \to r\sigma} u$ where t and l are variants
- $S \subseteq E'' \cap R_0$ such that $C_p(\mathcal{N}) \cup \{l\sigma \to r\sigma\}$ terminates for all $p \in S$
- $((R_0 \cup E) \cap R) \cup S \neq \varnothing$

$$\mathsf{orewrite}_2 \quad \frac{\mathcal{N} \cup \{\langle s : t, R_0, R_1, E, C_0, C_1\rangle\}}{\begin{array}{l}\mathrm{split}_S(\mathcal{N}) \cup \{\, \langle s : t, R'_0, R'_1, E', \mathrm{split}_S(C_0), \mathrm{split}_S(C_1)\rangle \\ \qquad \langle s : u, \mathrm{app}^1_S(R_0 \cap (R \cup S)), \varnothing, \mathrm{app}^1_S((E \cup R_1) \cap (R \cup S)), \varnothing, \varnothing\rangle \\ \qquad \langle l\sigma : r\sigma, \varnothing, \varnothing, \varnothing, \{p1 \mid p \in S\}, \varnothing\rangle \,\}\end{array}}$$

if
- $\langle l : r, R, \ldots, E'', \ldots\rangle \in \mathcal{N}$ and $t \xrightarrow{l\sigma \to r\sigma} u$ where $t \rhd l$
- $S \subseteq E'' \cap (R_0 \cup R_1 \cup E)$ such that $C_p(\mathcal{N}) \cup \{l\sigma \to r\sigma\}$ terminates for all $p \in S$
- $R'_0 = \mathrm{app}^0_S(R_0 \setminus R)$, $R'_1 = \mathrm{app}^0_S(R_1 \setminus R)$ and $E' = \mathrm{app}^0_S(E \setminus R)$
- $(R_0 \cup R_1 \cup E) \cap (R \cup S) \neq \varnothing$

$$\mathsf{odeduce} \quad \frac{\mathcal{N}}{\mathcal{N} \cup \{\, \langle s : t, \varnothing, \varnothing, (R \cup E) \cap (R' \cup E'), \varnothing, \varnothing\rangle \,\}}$$

if
- $\langle l : r, R, \ldots, E, \ldots\rangle, \langle l' : r', R', \ldots, E', \ldots\rangle \in \mathcal{N}$
- $s \xleftarrow{l \to r} u \xrightarrow{l' \to r'} t$ and $(R \cup E) \cap (R' \cup E') \neq \varnothing$

Fig. 5. The orewrite and odeduce inference rules in oMKBtt.

Given a well-encoded set of processes $P = \{p_1, \ldots, p_n\}$, let $\mathrm{app}^b_P(Q)$ denote $\mathrm{app}^b_{p_1}(\ldots\ \mathrm{app}^b_{p_n}(Q)\ \ldots)$ and set $\mathrm{split}_P(Q) = \mathrm{app}^0_P(Q) \cup \mathrm{app}^1_P(Q)$.

Note that if Q is well-encoded then both $\mathrm{app}^b_P(Q)$ and $\mathrm{split}_P(Q)$ are well-encoded. Therefore, all inference rules preserve well-encodedness and the disjointness condition on labels. Given an oMKBtt run $\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \mathcal{N}_2 \vdash \ldots$, the set $\mathcal{N}_\omega = \bigcup_i \bigcap_{j>i} \mathcal{N}_j$ collects *persisting nodes*. For a set of equations $\mathcal{E}$, the *initial* node set $\mathcal{N}_\mathcal{E}$ consists of all nodes $\langle s : t, \varnothing, \varnothing, \{\epsilon\}, \varnothing, \varnothing\rangle$ such that $s \approx t$ belongs to $\mathcal{E}$.

Example 17. We illustrate oMKBtt on the equations of Example 5. We start with the initial node set

$$\mathcal{N}_0 = \left\{ \begin{array}{ll} \langle \mathsf{g}(\mathsf{f}(x, \mathsf{b})) : \mathsf{a}, \varnothing, \varnothing, \{\epsilon\}, \varnothing, \varnothing\rangle & (1) \\ \langle \mathsf{f}(\mathsf{g}(x), y) : \mathsf{f}(x, \mathsf{g}(y)), \varnothing, \varnothing, \{\epsilon\}, \varnothing, \varnothing\rangle & (2) \end{array} \right\}$$

In the first step one may orient node (1), where only the direction from left to right is possible. Concerning the second node, both constraint systems

$$\{\, \mathsf{g}(\mathsf{f}(x, \mathsf{b})) \to \mathsf{a},\ \mathsf{f}(\mathsf{g}(x), y) \to \mathsf{f}(x, \mathsf{g}(y)) \,\}$$

and

$$\{\, \mathsf{g}(\mathsf{f}(x,\mathsf{b})) \to \mathsf{a},\ \mathsf{f}(x,\mathsf{g}(y)) \to \mathsf{f}(\mathsf{g}(x),y) \,\}$$

terminate, the first using LPO with precedence $\mathsf{f} > \mathsf{g} > \mathsf{a}$ and the second with the polynomial interpretation from Example 5. Hence the process ϵ is split:

$$\begin{array}{ll} \mathsf{g}(\mathsf{f}(x,\mathsf{b})) : \mathsf{a}, \{0,1\}, \varnothing, \varnothing, \{0,1\}, \varnothing\rangle & (1) \\ \langle \mathsf{f}(\mathsf{g}(x),y) : \mathsf{f}(x,\mathsf{g}(y)), \{0\}, \{1\}, \varnothing, \{0\}, \{1\}\rangle & (2) \end{array}$$

Starting from the overlap $\mathsf{g}(\mathsf{f}(x,\mathsf{g}(\mathsf{b}))) \leftarrow \mathsf{g}(\mathsf{f}(\mathsf{g}(x),\mathsf{b})) \to \mathsf{a}$ between nodes (1) and (2), if process 0 is advanced further then infinitely many nodes of the form $\langle \mathsf{g}(\mathsf{f}(x,\mathsf{g}^n(\mathsf{b}))) : \mathsf{a}, \{0\}, \varnothing, \varnothing, \{0\}, \varnothing\rangle$ are generated. On the other hand, similarly as in Example 5, one can deduce $\mathsf{f}(\mathsf{g}(x),\mathsf{f}(y,\mathsf{b})) \approx \mathsf{f}(x,\mathsf{c})$, orient the corresponding new node (3) and add the critical pair (4) between nodes (2) and (3). It remains to consider the overlaps between node (4) and itself to obtain

$$\begin{array}{ll} \langle \mathsf{f}(\mathsf{g}(x),\mathsf{f}(y,\mathsf{b})) : \mathsf{f}(x,\mathsf{c}), \{1\}, \varnothing, \varnothing, \{1\}, \varnothing\rangle & (3) \\ \langle \mathsf{f}(\mathsf{f}(x,\mathsf{b}),\mathsf{a}) : \mathsf{f}(\mathsf{a},\mathsf{f}(y,\mathsf{b})), \varnothing, \varnothing, \{1\}, \varnothing, \varnothing\rangle & (4) \\ \langle \mathsf{f}(\mathsf{f}(x,\mathsf{b}),\mathsf{a}) : \mathsf{f}(\mathsf{f}(y,\mathsf{b}),\mathsf{a}), \varnothing, \varnothing, \{1\}, \varnothing, \varnothing\rangle & (5) \\ \langle \mathsf{f}(\mathsf{a},\mathsf{f}(x,\mathsf{b})) : \mathsf{f}(\mathsf{a},\mathsf{f}(y,\mathsf{b})), \varnothing, \varnothing, \{1\}, \varnothing, \varnothing\rangle & (6) \end{array}$$

at which point process 1 is saturated. Applying the projections $E_1(\mathcal{N})$ and $R_1(\mathcal{N})$ to the current node set $\mathcal{N} = \{(1), \ldots, (6), \ldots\}$ yields the ground-complete system $(\mathcal{E}, \mathcal{R})$ derived in Example 5.

In the following paragraphs we briefly sketch the roles of these inference rules.

- As in MKBtt, $\mathsf{orewrite}_1$ simulates the oKBtt inferences compose and simplify whenever t and l are variants: If there is a node with data $s : t$ and t can be rewritten to u using a rule $l \to r$ then $\mathsf{orewrite}_1$ creates a node $s : u$. For processes in the new node's rewrite label $R \cap R_0$, this models a compose, for processes in the equation label $R \cap E$ a simplify inference step. Moreover, $\mathsf{orewrite}_1$ models $\mathsf{compose}_2$ for processes p in $E'' \cup R_0$ which might not orient $l : r$ but can orient the instance $l\sigma \to r\sigma$ used to rewrite t to u, i.e., $C_p \cup \{l\sigma \to r\sigma\}$ terminates. For these processes a respective constraint node with data $l\sigma : r\sigma$ is added. Since such a process p might also be able to orient the reversed rule $l\sigma \to r\sigma$, which might result in a considerably different run it gets split at that point. Therefore the descibed inference step is actually only performed for $p1$, whereas for $p0$ the state remains the same. [2]
- In a similar way, $\mathsf{orewrite}_2$ models these inference steps together with collapse, $\mathsf{simplify}_2$ and $\mathsf{collapse}_2$ if $t \rhd l$.
- If nodes with data $l : r$ and $l' : r'$ allow for an overlap $s \xleftarrow{l \to r} u \xrightarrow{l' \to r'} t$ and P is a set of processes which consider these term pairs either as equations or

[2] For the sake of readability, the paper contains a simplified version of the inference rules where no splitting occurs in orewrite steps. However, this change does not affect the results established below (i.e., Lemmata 19 – 22 and Theorems 23 and 24).

rules, deduce derives a respective new node with data $s : t$ and equation label P. In our implementation, the application of this inference rule is limited to an overapproximation of extended critical pairs.

Definition 18. *If an oMKBtt inference step $\mathcal{N} \vdash \mathcal{N}'$ applies orient, orewrite$_1$ or orewrite$_2$ then S is called the step's* split set*. In all other cases, the split set is empty. For a step with split set S and $p' \in \mathcal{P}(\mathcal{N}')$, the predecessor of p' is defined as*

$$\mathrm{pred}_S(p') = \begin{cases} p & \text{if } p' = p0 \text{ or } p' = p1 \text{ for some } p \in S \\ p' & \text{otherwise} \end{cases}$$

Lemma 19. *If $\mathcal{N} \vdash \mathcal{N}'$ is an oMKBtt step with split set S then*

$$(E_p(\mathcal{N}), R_p(\mathcal{N}), C_p(\mathcal{N})) \vdash^{=} (E_{p'}(\mathcal{N}'), R_{p'}(\mathcal{N}'), C_{p'}(\mathcal{N}'))$$

is a valid oKBtt inference for all $p' \in \mathcal{P}(\mathcal{N}')$, where $p = pred_S(p')$. Moreover, the strict part $\vdash$ holds for at least one $p' \in \mathcal{P}(\mathcal{N}')$.

Here, $\vdash^{=}$ denotes the reflexive closure of the oMKBtt inference relation $\vdash$.

Proof. By case distinction on the applied oMKBtt inference rule in $\mathcal{N} \vdash \mathcal{N}'$. In the case of orewrite$_1$, for any process where $p = p'$ and $p \notin (R_0 \cup E) \cap (R \cup S)$ we have $(E_p(\mathcal{N}), R_p(\mathcal{N}), C_p(\mathcal{N})) = (E_{p'}(\mathcal{N}'), R_{p'}(\mathcal{N}'), C_{p'}(\mathcal{N}'))$. Otherwise, we distinguish different cases using the notations in the applied inference rule. Due to the disjointness condition on labels, these cases are disjoint.

i. If $p' \in R_0 \cap R$ then $p = p'$ and there are rules $s \to t$ and $l \to r$ in $R_p(\mathcal{N})$ such that $t \xrightarrow{l \to r} u$. Thus compose can be applied to obtain the rule $s \to u$, which indeed belongs to $R_p(\mathcal{N}')$.

ii. If $p' \in E \cap R$, again $p = p'$ and there is an equation $s \approx t$ in $E_p(\mathcal{N})$ and a rule $l \to r$ in $R_p(\mathcal{N})$ such that $t \xrightarrow{l \to r} u$. Thus simplify is applicable, yielding $s \approx u$, which is in $E_p(\mathcal{N}')$.

iii. In case $p' \in \mathrm{split}(S)$ we have $p' = p0$ or $p' = p1$ for some $p \in S$. For $p0$ we have $(E_p(\mathcal{N}), R_p(\mathcal{N}), C_p(\mathcal{N})) = (E_{p0}(\mathcal{N}'), R_{p0}(\mathcal{N}'), C_{p0}(\mathcal{N}'))$. For $p1$, as there are a rule $s \to t$ in $R_p(\mathcal{N})$ and an equation $l \approx r$ in $E_p(\mathcal{N})$ such that $t \xrightarrow{l\sigma \to r\sigma} u$ and $C_p(\mathcal{N}) \cup \{l\sigma \to r\sigma\}$ terminates, compose$_2$ is applicable. Since $s \to u \in R_{p1}(\mathcal{N}')$ and $l\sigma \to r\sigma \in C_{p1}(\mathcal{N}')$ the resulting sets match.

Next we consider orewrite$_2$. For any process p where $p = p'$ and $p \notin (R_0 \cup R_1 \cup E) \cap (R \cup S)$ we have $(E_p(\mathcal{N}), R_p(\mathcal{N}), C_p(\mathcal{N})) = (E_{p'}(\mathcal{N}'), R_{p'}(\mathcal{N}'), C_{p'}(\mathcal{N}'))$. Otherwise, we distinguish the following (disjoint) cases. If $p \in R_0 \cap R$, $p \in E \cap R$ or $p \in R_0 \cap S$ then one may argue as in cases (i), (ii) and (iii) above, respectively.

iv. If $p \in R_1 \cap R$ we have $p = p'$ and the rule $t \to s$ in $R_p(\mathcal{N})$ can be collapsed by $l \to r$ as $t \rhd l$, resulting in the equation $s \approx u$ which indeed belongs to $E_p(\mathcal{N}')$.

v. Next consider the case where $p' \in \text{split}(S)$ and thus $p' = p0$ or $p' = p1$ for some $p \in E \cap S$. For $p0$ we have that $(E_p(\mathcal{N}), R_p(\mathcal{N}), C_p(\mathcal{N}))$ equals $(E_{p0}(\mathcal{N}'), R_{p0}(\mathcal{N}'), C_{p0}(\mathcal{N}'))$. For $p1$ there are equations $s \approx t$ and $l \approx r$ in $E_p(\mathcal{N})$ such that $t \xrightarrow{l\sigma \to r\sigma} u$ and $C_p(\mathcal{N}) \cup \{l\sigma \to r\sigma\}$ terminates. As $t \rhd l$, $\mathsf{simplify}_2$ can be applied to obtain the equation $s \approx u$ belonging to $E_{p1}(\mathcal{N}')$. Also, $l\sigma \to r\sigma$ belongs to $C_{p1}(\mathcal{N}')$.

vi. Similarly, if $p' \in \text{split}(S)$ and $p' = p0$ or $p' = p1$ for some $p \in R_1 \cap S$ then for $p0$ we have an equality step while for $p1$ $\mathsf{collapse}_2$ applies: Since $t \to s$ is in $R_p(\mathcal{N})$, $l\sigma \to r\sigma$ rewrites t to u where $t \rhd l$, and $C_p(\mathcal{N}) \cup \{l\sigma \to r\sigma\}$ terminates, the inference yields an equation $s \approx u$ which is in $E_{p1}(\mathcal{N}')$.

If $\mathsf{odeduce}$ is applied, we always have $p' = p$ since no splitting occurs. For every process $p \in (R \cup E) \cap (R' \cup E')$ we have $l \approx r$ and $l' \approx r'$ in $E_p(\mathcal{N}) \cup R_p(\mathcal{N})$. Thus applying deduce_2 yields the equation $s \approx t$ which is also in $E_p(\mathcal{N}')$. Next, assume orient is applied. If $p' \notin R_{lr} \cup R_{rl}$ then, by definition of R_{lr} and R_{rl}, $p \notin S$. Thus $p' = p$ and the projection yields an identity step.

i. If $p' \in R_{lr}$ then $s \to t$ is in both $R_{p'}(\mathcal{N}')$ and $C_{p'}(\mathcal{N}')$. By definition of R_{lr} we have either $p' = p \in E_{lr} \setminus E_{rl}$ or $p' = p0$ for some $p \in E_{rl} \cap E_{lr}$. In both cases $p \in E$, thus the inference rule orient of oKBtt applies since $C_p(\mathcal{N}) \cup \{s \to t\}$ terminates. The resulting sets $R_p(\mathcal{N}) \cup \{s \to t\}$ and $C_p(\mathcal{N}) \cup \{s \to t\}$ coincide with $R_{p'}(\mathcal{N}')$ and $C_{p'}(\mathcal{N}')$.

ii. The case $p' \in R_{rl}$ is symmetric to the previous case. Hence orient applies as well.

If delete removed a node $\langle s : s, \varnothing, \varnothing, E, \varnothing, \varnothing\rangle$ then $s \approx s \in E_p(\mathcal{N})$ for any $p \in E$, and hence the corresponding oKBtt inference applies. For all $p \notin E$ we obtain an identity step. Finally, for every inference rule the non-emptiness requirement of a respective label set ensures that the strict part $\vdash$ holds for at least one $p' \in \mathcal{P}(\mathcal{N}')$. □

Lemma 20. *Consider an oKBtt inference step $(\mathcal{E}, \mathcal{R}, \mathcal{C}) \vdash (\mathcal{E}', \mathcal{R}', \mathcal{C}')$. Assume there exist a node set $\mathcal{N}$ and a process p such that $\mathcal{E} = E_p(\mathcal{N})$, $\mathcal{R} = R_p(\mathcal{N})$ and $\mathcal{C} = C_p(\mathcal{N})$. Then there are a node set $\mathcal{N}'$, an inference step $\mathcal{N} \vdash \mathcal{N}'$ with split set S, and a process $p' \in \mathcal{P}(\mathcal{N}')$ such that $p = \text{pred}_S(p')$, $\mathcal{E}' = E_{p'}(\mathcal{N}')$, $\mathcal{R}' = R_{p'}(\mathcal{N}')$ and $\mathcal{C}' = C_{p'}(\mathcal{N}')$.*

Proof. The proof obligations

$$\mathcal{E}' = E_{p'}(\mathcal{N}'),\ \mathcal{R}' = R_{p'}(\mathcal{N}'),\ \mathcal{C}' = C_{p'}(\mathcal{N}')$$

will be referred to as $(\star)$. We use case distinction on the applied oKBtt rule.

Assume orient was used to turn an equation $s \approx t$ into a rule $s \to t$, so $\mathcal{C}_p(\mathcal{N}) \cup \{s \to t\}$ terminates and there must be a node $\langle s : t, R_0, R_1, E \uplus \{p\}, C_0, C_1\rangle$ in $\mathcal{N}$. We distinguish two further cases. If $\mathcal{C}_p(\mathcal{N}) \cup \{t \to s\}$ does not terminate, we can apply the oMKBtt rule orient with $S = \varnothing$, so we set

$$\begin{aligned}\mathcal{N}' = (\mathcal{N} \setminus \{\langle s : t, R_0, R_1, E \uplus \{p\}, C_0, C_1\rangle\}) \cup \\ \{\langle s : t, R_0 \cup \{p\}, R_1, E, C_0 \cup \{p\}, C_1\rangle\}\end{aligned}$$

and $p' = p$ (trivially satisfying $p = \mathrm{pred}_S(p')$) such that condition $(\star)$ holds. If $\mathcal{C}_p(\mathcal{N}) \cup \{t \to s\}$ terminates as well, we apply orient with $S = \{p\}$ to obtain

$$\begin{aligned}\mathcal{N}' = (\mathcal{N} \setminus \{\langle s : t, R_0, R_1, E \uplus \{p\}, C_0, C_1\rangle\}) \cup \\ \{\langle s : t, R_0 \cup \{p0\}, R_1 \cup \{p1\}, E, C_0 \cup \{p0\}, C_1 \cup \{p1\}\rangle\}\end{aligned}$$

For $p' = p0$ we have $p = \mathrm{pred}_S(p')$ and $(\star)$ is satisfied.

Next, assume compose rewrites $s \to t$ to $s \to u$ using the rule $l \to r$. Hence there are nodes $n = \langle s : t, R_0 \uplus \{p\}, R_1, E, C_0, C_1\rangle$ and $\langle l : r, R \cup \{p\}, \dots\rangle$ in $\mathcal{N}$. If t and l are variants we can apply $\mathsf{orewrite}_1$, if $t \rhd l$ we can apply $\mathsf{orewrite}_2$ to obtain

$$\mathcal{N}' = (\mathcal{N} \setminus \{n\}) \cup \{\langle s : t, R_0, R_1, E, C_0, C_1\rangle, \langle s : u, \{p\}, \varnothing, \varnothing, \varnothing, \varnothing\rangle\}$$

For $p' = p$ we thus have $\mathcal{R}' = R_p(\mathcal{N}') = (R_p(\mathcal{N}) \setminus \{s \to t\}) \cup \{s \to u\}$ and $p = \mathrm{pred}_S(p')$ such that $(\star)$ is satisfied. Similarly, simplify can be simulated by $\mathsf{orewrite}_1$ if the left-hand side is a variant of the redex, and by $\mathsf{orewrite}_2$ otherwise. For collapse only $\mathsf{orewrite}_2$ applies.

Assume $\mathsf{compose}_2$ rewrites $s \to t$ to $s \to u$ using an instance $l\sigma \to r\sigma$ of an equation $l \approx r$. Hence $\mathcal{N}$ contains nodes $n = \langle s : t, R_0 \uplus \{p\}, R_1, E, C_0, C_1\rangle$ and $\langle l : r, \dots, E \cup \{p\}, \dots\rangle$. If t and l are variants we can apply $\mathsf{orewrite}_1$, if $t \rhd l$ $\mathsf{orewrite}_2$ to obtain

$$\begin{aligned}\mathcal{N}' = \mathrm{split}_{\{p\}}(\mathcal{N} \setminus \{n\}) \cup \{&\langle s : t, R_0 \cup \{p0\}, R_1, E, \mathrm{split}_{\{p\}}(C_0), C_1\rangle, \\ &\langle s : u, \{p1\}, \varnothing, \varnothing, \varnothing, \varnothing\rangle, \\ &\langle l\sigma : r\sigma, \varnothing, \varnothing, \varnothing, \{p1\}, \varnothing\rangle\}\end{aligned}$$

since $C_p(\mathcal{N}) \cup \{l\sigma \to r\sigma\}$ terminates. For $p' = p1$ we thus have $p = \mathrm{pred}_S(p')$, $\mathcal{R}' = R_{p'}(\mathcal{N}') = (R_p(\mathcal{N}) \setminus \{s \to t\}) \cup \{s \to u\}$ and $\mathcal{C}' = C_{p'}(\mathcal{N}') = C_p(\mathcal{N}) \cup \{l\sigma \to r\sigma\}$ such that condition $(\star)$ is satisfied. Similarly, $\mathsf{simplify}_2$ and $\mathsf{collapse}_2$ can be simulated by $\mathsf{orewrite}_2$.

In the remaining cases no splitting occurs, so we always set $p' = p$. If delete removes an equation $s \approx s$ then there must be a node $\langle s : s, \varnothing, \varnothing, E \uplus \{p\}, \varnothing, \varnothing\rangle$ in $\mathcal{N}$ which can be deleted by the corresponding oMKBtt rule so $(\star)$ is satisfied.

Finally, assume deduce_2 adds an equation $s \approx t$ resulting from an overlap

$$s \xleftarrow{l\sigma \to r\sigma} u \xrightarrow{l'\sigma \to r'\sigma} t$$

such that $l \approx r, l' \approx r' \in \mathcal{E} \cup \mathcal{R}$. So there are nodes $\langle l : r, R_0, \dots, E, \dots\rangle$ and $\langle l' : r', R'_0, \dots, E', \dots\rangle$ in $\mathcal{N}$ such that $p \in (R_0 \cup E) \cap (R'_0 \cup E')$. Hence we can deduce $\langle s : t, \varnothing, \varnothing, \{p\}, \varnothing, \varnothing\rangle$ in oMKBtt, and $(\star)$ holds. □

Projecting an oMKBtt run γ to a process $p \in \mathcal{P}(\mathcal{N}_\alpha)$ thus yields a valid oKBtt run, which is denoted by γ_p in the sequel. Before correctness can be addressed, we adapt the definition of (sufficient) fairness and note that oMKBtt is sound.

Definition 21. *A run γ of length α is* sufficiently fair *if either $\alpha < \omega$ and γ_p is sufficiently fair for at least one process $p \in \mathcal{P}(\mathcal{N}_\alpha)$, or $\alpha = \omega$ and γ_p is sufficiently fair for all $p \in \mathcal{P}(\mathcal{N}_\alpha)$.*

Lemma 22. *Consider an oMKBtt step $\mathcal{N} \vdash \mathcal{N}'$ with split set S and a process $q \in \mathcal{P}(\mathcal{N}')$ with predecessor $p = \mathrm{pred}_S(q)$. For $\mathcal{E} = E_p(\mathcal{N})$, $\mathcal{R} = R_p(\mathcal{N})$ and $\mathcal{E}' = E_q(\mathcal{N}')$, $\mathcal{R}' = R_q(\mathcal{N}')$ the relations $\leftrightarrow^*_{\mathcal{E}\cup\mathcal{R}}$ and $\leftrightarrow^*_{\mathcal{E}'\cup\mathcal{R}'}$ coincide.*

Similar to the oKBtt case, an *oMKBtt$_\mathcal{P}$ procedure* refers to a program that takes a set of equations $\mathcal{E}$ as input and uses the inference rules of oMKBtt to generate a derivation starting from $\mathcal{N}_\mathcal{E}$, where termination checks are performed with respect to a termination strategy $\mathcal{P}$. An oMKBtt$_{\mathsf{total}}$ procedure is any oMKBtt$_\mathcal{P}$ procedure where $\mathcal{P}$ guarantees total termination of the checked systems.

Using the simulation properties expressed in Lemmata 19 and 20, correctness and completeness easily follow from the corresponding results for oKBtt.

Theorem 23. *Let $\mathcal{N}_0 = \mathcal{N}_\mathcal{E}$ be the initial node set for $\mathcal{E}$ and let $\mathcal{N}_0 \vdash^* \mathcal{N}_n$ be a finite oMKBtt$_{\mathsf{total}}$ run. If $\mathcal{N}_0 \vdash^* \mathcal{N}_n$ is sufficiently fair for $p \in \mathcal{P}(\mathcal{N}_n)$ then $E_p(\mathcal{N}_n) \cup R_p(\mathcal{N}_n)$ is ground-complete for a reduction order $>$ that is total on ground terms and extends $\rightarrow^+_\mathcal{C}$, where $\mathcal{C} = C_p(\mathcal{N}_n)$.*

Proof. By Lemma 19 there exists a corresponding oKBtt$_{\mathsf{total}}$ run

$$(E_p(\mathcal{N}_0), R_p(\mathcal{N}_0), C_p(\mathcal{N}_0)) \vdash^* (E_p(\mathcal{N}_n), R_p(\mathcal{N}_n), C_p(\mathcal{N}_n))$$

which is sufficiently fair. Let $\mathcal{C}$ denote the final constraint system $C_p(\mathcal{N}_n)$. By Theorem 12, $E_p(\mathcal{N}_n) \cup R_p(\mathcal{N}_n)$ is ground-complete with respect to any reduction order $>$ that is total on ground terms and contains $\rightarrow^+_\mathcal{C}$. Since $\mathcal{C}$ is totally terminating such an order exists. □

Theorem 24. *Assume $\mathcal{R}$ is a complete rewrite system for $\mathcal{E}$ and $\succ$ is a reduction order containing $\mathcal{R}$ which can be extended to a total reduction order. Then there exists a sufficiently fair and simplifying oMKBtt$_{\mathsf{total}}$ run $\mathcal{N}_\mathcal{E} \vdash^* \mathcal{N}_\alpha$ such that some process $p \in \mathcal{P}(\mathcal{N}_\alpha)$ satisfies $R_p(\mathcal{N}_\alpha) = \mathcal{R}$ and $E_p(\mathcal{N}_\alpha) = \varnothing$.*

Proof. According to Theorem 13, there exists a sufficiently fair run of oKBtt$_{\mathsf{total}}$ which produces the result $\mathcal{R}_\omega = \mathcal{R}$ and $\mathcal{E}_\omega = \varnothing$. Lemma 20 entails that this run can be simulated in an oMKBtt$_{\mathsf{total}}$ run starting from $\mathcal{N}_\mathcal{E}$. □

5 Theorem Proving with oMKBtt

The use of ordered completion for refutational theorem proving proposed in [2] can easily be adapted to the oMKBtt setting. For a term s, we write $\hat{s}$ to denote the term where each variable is replaced by its corresponding Skolem constant. In the sequel, given equations $\mathcal{E}$ and a goal $s \approx t$, let $\mathcal{N}^{s\approx t}_\mathcal{E}$ denote the set

$$\mathcal{N}_\mathcal{E} \cup \{\ \langle \mathsf{equal}(x,x) : \mathsf{true}, \varnothing, \varnothing, \{\epsilon\}, \varnothing, \varnothing\rangle,\ \langle \mathsf{equal}(\hat{s},\hat{t}) : \mathsf{false}, \varnothing, \varnothing, \{\epsilon\}, \varnothing, \varnothing\rangle\ \}$$

As the following results show, theorem proving with oMKBtt is sound, independent of the applied termination techniques. To obtain completeness we restrict to oMKBtt$_{\mathsf{total}}$ procedures.

Lemma 25. *If an oMKBtt run starting from $\mathcal{N}_0 = \mathcal{N}_{\mathcal{E}}^{s \approx t}$ generates a node $\langle \mathsf{true} : \mathsf{false}, \ldots, E, \ldots \rangle$ in some set $\mathcal{N}_i$ and $E \neq \varnothing$ then $s \approx t$ is valid in $\mathcal{E}$.*

Proof. By Lemma 19 there is a process $p \in E$ such that the oKBtt run γ_p derives $\mathsf{true} \approx \mathsf{false}$. As oKBtt is sound as stated in Lemma 6, $\mathsf{true} \approx \mathsf{false}$ and thus also $\hat{s} \approx \hat{t}$ is in the equational theory of $\mathcal{E}$. Since the introduced Skolem constants are fresh, also $s \approx_{\mathcal{E}} t$ holds. □

Lemma 26. *If $s \approx t$ is valid in $\mathcal{E}$ then any sufficiently fair $\mathsf{oMKBtt}_{\mathsf{total}}$ run $\mathcal{N}_0 \vdash \mathcal{N}_1 \vdash \cdots \vdash \mathcal{N}_\alpha$ starting from $\mathcal{N}_0 = \mathcal{N}_{\mathcal{E}}^{s \approx t}$ generates a node $\langle \mathsf{true} : \mathsf{false}, \ldots, E, \ldots \rangle$ in some set $\mathcal{N}_i$ such that $E \neq \varnothing$.*

Proof. Since the run is sufficiently fair it is sufficiently fair for some process p. By Lemma 19 there is a sufficiently fair oKBtt run

$$(E_p(\mathcal{N}_0), R_p(\mathcal{N}_0), C_p(\mathcal{N}_0)) \vdash^* (E_p(\mathcal{N}_\alpha), R_p(\mathcal{N}_\alpha), C_p(\mathcal{N}_\alpha))$$

According to Lemma 7, there is a corresponding fair oKB run using the reduction order $\rightarrow_{\mathcal{C}}^+$, where $\mathcal{C} = C_p(\mathcal{N}_\alpha)$. Moreover, $\rightarrow_{\mathcal{C}}^+$ can be extended to a reduction order $>$ that is total on ground terms. By [2, Theorem 3], such a fair ordered completion run starting from $\mathcal{E}_0 = \mathcal{E} \cup \{\mathsf{equal}(x, x) \approx \mathsf{true},\ \mathsf{equal}(\hat{s}, \hat{t}) \approx \mathsf{false}\}$ will have the contradictory statement $\mathsf{true} \approx \mathsf{false}$ in some set $\mathcal{E}_i \cup \mathcal{R}_i$, so there is a node $\langle \mathsf{true} : \mathsf{false}, \ldots \rangle$ in some $\mathcal{N}_i$. □

Example 27. In Example SYN080-1 from TPTP 3.6.0 [14] the conjecture $\mathsf{f}(\mathsf{f}(\mathsf{a})) \approx \mathsf{f}(\mathsf{g}(\mathsf{b}))$ is to be proven from the axiom $\mathsf{f}(x) \approx \mathsf{g}(y)$. In the setting of oMKBtt, one starts with the nodes

$$\begin{array}{ll} \langle \mathsf{f}(x) : \mathsf{g}(y), \varnothing, \varnothing, \{\epsilon\}, \varnothing, \varnothing \rangle & (1) \\ \langle \mathsf{equal}(x, x) : \mathsf{true}, \varnothing, \varnothing, \{\epsilon\}, \varnothing, \varnothing \rangle & (2) \\ \langle \mathsf{equal}(\mathsf{f}(\mathsf{f}(\mathsf{a})), \mathsf{f}(\mathsf{g}(\mathsf{b}))) : \mathsf{false}, \varnothing, \varnothing, \{\epsilon\}, \varnothing, \varnothing \rangle\} & (3) \end{array}$$

Node (2) can be oriented, and thus gets modified as follows:

$$\langle \mathsf{equal}(x, x) : \mathsf{true}, \{\epsilon\}, \varnothing, \varnothing, \{\epsilon\}, \varnothing \rangle \quad (2)$$

We can apply $\mathsf{orewrite}_2$ to node (3) by using the instance $\mathsf{f}(\mathsf{a}) \rightarrow \mathsf{g}(\mathsf{b})$ of node (1) since the constraint system $\{\mathsf{equal}(x, x) \rightarrow \mathsf{true},\ \mathsf{f}(\mathsf{a}) \rightarrow \mathsf{g}(\mathsf{b})\}$ terminates. This inference modifies node (3) and adds node (4) as well as the constraint node (5).

$$\begin{array}{ll} \langle \mathsf{equal}(\mathsf{f}(\mathsf{f}(\mathsf{a})), \mathsf{f}(\mathsf{g}(\mathsf{b}))) : \mathsf{false}, \varnothing, \varnothing, \{\mathsf{0}\}, \varnothing, \varnothing \rangle\} & (3) \\ \langle \mathsf{equal}(\mathsf{f}(\mathsf{g}(\mathsf{b})), \mathsf{f}(\mathsf{g}(\mathsf{b}))) : \mathsf{false}, \{\mathsf{1}\}, \varnothing, \varnothing, \varnothing, \varnothing \rangle & (4) \\ \langle \mathsf{f}(\mathsf{a}) : \mathsf{g}(\mathsf{b}), \varnothing, \varnothing, \varnothing, \{\mathsf{1}\}, \varnothing \rangle\} & (5) \end{array}$$

Now $\mathsf{orewrite}_2$ can use node (2) to simplify node (4) and derive node (6).

$$\begin{array}{ll} \langle \mathsf{equal}(\mathsf{f}(\mathsf{g}(\mathsf{b})), \mathsf{f}(\mathsf{g}(\mathsf{b}))) : \mathsf{false}, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing \rangle & (4) \\ \langle \mathsf{true} : \mathsf{false}, \varnothing, \varnothing, \{\mathsf{1}\}, \varnothing, \varnothing \rangle & (6) \end{array}$$

Since node (6) has a non-empty equation label this proves the goal.

Example 28. Since $\mathsf{omkb}_{\mathsf{TT}}$ completes the theory of Example 17, also the conjecture $\mathsf{g}(\mathsf{f}(\mathsf{b}, \mathsf{g}^{200}(\mathsf{b}))) \approx \mathsf{c}$ is proved in a few seconds while Waldmeister cannot verify this goal within 1800 seconds.

6 Implementation

This section briefly outlines our tool omkbTT. Extending the existing mkbTT implementation [11, 17], it is implemented in OCaml in about 10.000 lines of code. To check constraint systems for termination, omkbTT either uses an external tool which is compatible with a minimal interface or interfaces TTT2 [8] internally.

Our tool omkbTT is equipped with a simple command-line interface. The input system is expected in the TPTP-3 [14] format. Among other options, users can fix the global time limit and the time limit for a termination call, specify either an external executable for termination checks or configure how TTT2 should be used internally, and control which indexing technique, node selection strategy or goal representation to use. For further details we refer to the website and [17].

In the original presentation of completion-based theorem proving [2], given a goal $s \approx t$ the equations $\mathsf{equal}(x, x) \approx \mathsf{true}$ and $\mathsf{equal}(\hat{s}, \hat{t}) \approx \mathsf{false}$ are added. Waldmeister uses a different representation of the goal [5]. The reducts of $\hat{s}$ and $\hat{t}$ are kept in two sets R_s and R_t. Whenever a term in R_s or R_t can be reduced, the new reducts are added to R_s or R_t, respectively. The goal is proven as soon as $R_s \cap R_t$ is non-empty. This approach is supported in omkbTT as well. Sets R_s and R_t of pairs (u, P) where u is a term and P the set of processes for which this reduct was derived are maintained. The goal is proven if there exists a term u such that $(u, P) \in R_s$, $(u, P') \in R_t$ and $P \cap P'$ is non-empty.

7 Experimental Results

This section summarizes experimental results obtained with omkbTT. All tests were run on a single core of a server equipped with eight dual-core AMD Opteron® processors 885 running at a clock rate of 2.6GHz and 64GB of main memory.

In all of the following tests omkbTT internally interfaces TTT2 for termination checks. To compare the applicability of different termination techniques, different TTT2 strategies were used: kbo, lpo and mpo denote the well-known reduction orders and poly refers to linear polynomial interpretations with coefficients in $\{0, \ldots, 7\}$. The strategy where all these techniques performed in parallel are applied iteratively is denoted by ttt2total. The strategy ttt2fast involves dependency pairs so total termination is not ensured. It is therefore only used for theorem proving, which is sound according to Lemma 25, although incomplete because completeness of refutational theorem proving holds only for totalizable reduction orders [2].

Examples stem from the unit equality division of TPTP 3.6.0 [14]. The test set e consists of 215 problems rated *easy*, d contains 565 problems classified as *difficult*. The sets et and dt consist of the 204 and 563 different theories associated with these problems. Table 1 shows ordered completion results obtained with omkbTT. The columns list (1) the number of successes, (2) the average time for a successful run in seconds (given a timeout of 600 seconds), and (3) the percentage of time spent on termination checks. In order to compare with other ordered completion tools, we ran E [13] on the same test set in *auto* mode,

	ttt2total		kbo		lpo		poly		mpo		E	
	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
et	37	22.8	38	23.5	23	22.1	35	34.1	37	29.0	10	0.04
dt	45	24.5	55	17.4	24	156.5	44	11.5	45	10.5	35	0.06

Table 1. Completing theories associated with TPTP UEQ systems.

	ttt2total			kbo			lpo			poly			ttt2fast		
	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)	(1)	(2)	(3)
e	149	43.9	82	163	16.6	8	164	24.3	14	143	59.1	90	138	49.9	80
d	116	66.0	64	148	64.8	4	152	50.6	6	109	95.7	79	121	55.0	17

Table 2. Performance of oMKBtt on TPTP UEQ problems.

such that it heuristically determines the reduction order to use.[3] As an example, $\mathsf{omkb}_{\mathsf{TT}}$ using ttt2total completes the theory underlying problem GRP447-1 from TPTP within 3 seconds, while neither E nor $\mathsf{mkb}_{\mathsf{TT}}$ produce a solution within 1 hour. For example, $\mathsf{omkb}_{\mathsf{TT}}$ using ttt2total completes the theory underlying problem GRP447-1 from TPTP within 3 seconds, while neither E nor $\mathsf{mkb}_{\mathsf{TT}}$ produce a solution within 1 hour.

The over-approximation of extended critical pairs with the embedding relation, i.e., the use of $CP_{\rhd_{\mathsf{emb}}}$ instead of $CP_{\varnothing}$ allows for a performance gain of about 28%.

Table 2 shows theorem proving results obtained with $\mathsf{omkb}_{\mathsf{TT}}$. Both Waldmeister and E solve about 200 problems in e and more than 400 of the d set.[4]

Although the considered termination strategies are incomparable in power, kbo handles the most problems, both for ordered completion and theorem proving. The reason for that is that little time is spent on termination checks, as can be seen from Table 2. Although the combination of multiple techniques in ttt2total is theoretically more powerful than each technique separately, the larger number of processes (25% more than kbo and twice as much as in lpo or poly) decreases performance and causes more timeouts. The evaluation of different combinations of termination strategies, such as the incremental use of polynomial interpretations, is subject to future work.

We compared the simple approach where the goal is represented as two nodes with the Waldmeister-like approach described in Section 6. According to our results, the latter is faster and therefore able to prove about 3% more examples. However, in some cases the simple approach succeeds whereas the Waldmeister-like approach fails due to a "combinatorial explosion".

[3] We did not use Waldmeister here since, according to personal communication with the developers, its *auto* mode should not be used for ordered completion.

[4] It should be noted that $\mathsf{omkb}_{\mathsf{TT}}$ cannot (yet) cope with existentially quantified goals. There are 16 such problems in e and 61 in d.

8 Conclusion

We outlined how termination tools can replace a fixed term order in ordered completion and completion-based theorem proving. This approach can also be combined with multi-completion. Besides the advantage that no reduction order has to be provided as input, this novel approach allows to derive ground-complete systems for problems that are not compatible with standard orders such as LPO and KBO. Hence our tool $\mathsf{omkb}_{\mathsf{TT}}$ can deal with input systems that cannot be solved with other tools, to the best of our knowledge.

In contrast to standard completion, in the case of ordered completion the reduction order implicitly developed in the inference sequence needs to be extensible to a reduction order which is complete for the theory. Hence $\mathsf{omkb}_{\mathsf{TT}}$ restricts to termination techniques which entail *total termination*. It is subject to further research whether the existence of a suitable order $>$ can be guaranteed by other means such that applicable termination techniques are less restricted.

Acknowledgements The comments of the anonymous referees helped to improve the paper.

References

1. L. Bachmair and N. Dershowitz. Equational inference, canonical proofs, and proof orderings. *Journal of the ACM*, 41(2):236–276, 1994.
2. L. Bachmair, N. Dershowitz, and D. A. Plaisted. Completion without failure. In H. Aït Kaci and M. Nivat, editors, *Resolution of Equations in Algebraic Structures*, volume 2: Rewriting Techniques of *Progress in Theoretical Computer Science*, pages 1–30. Academic Press, 1989.
3. M. Bofill, G. Godoy, R. Nieuwenhuis, and A. Rubio. Paramodulation and Knuth–Bendix completion with nontotal and nonmonotonic orderings. *Journal of Automated Reasoning*, 30(1):99–120, 2003.
4. N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982.
5. T. Hillenbrand and B. Löchner. The next Waldmeister loop. In *Proc. 18th CADE*, volume 2392 of *LNAI*, pages 486–500, 2002.
6. S. Kamin and J.J. Lévy. Two generalizations of the recursive path ordering. Unpublished manuscript, University of Illinois, 1980.
7. D.E. 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.
8. M. Korp, C. Sternagel, H. Zankl, and A. Middeldorp. Tyrolean termination tool 2. In *Proc. 20th RTA*, volume 5595 of *LNCS*, pages 295–304, 2009.
9. M. Kurihara and H. Kondo. Completion for multiple reduction orderings. *Journal of Automated Reasoning*, 23(1):25–42, 1999.
10. D. Lankford. On proving term rewrite systems are noetherian. Technical Report MTP-3, Louisiana Technical University, 1979.
11. H. Sato, S. Winkler, M. Kurihara, and A. Middeldorp. Multi-completion with termination tools (system description). In *Proc. 4th IJCAR*, volume 5195 of *LNAI*, pages 306–312, 2008.

12. H. Sato, S. Winkler, M. Kurihara, and A. Middeldorp. Constraint-based multi-completion procedures for term rewriting systems. *IEICE Transactions on Electronics, Information and Communication Engineers*, E92-D(2):220–234, 2009.
13. S. Schulz. The E Equational Theorem Prover, 2009. Available from http://www.eprover.org.
14. G. Sutcliffe. The TPTP problem library and associated infrastructure. *Journal of Automated Reasoning*, 43(4):337–362, 2009.
15. Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.
16. I. Wehrman, A. Stump, and E.M. Westbrook. Slothrop: Knuth-Bendix completion with a modern termination checker. In *Proc. 17th RTA*, volume 4098 of *LNCS*, pages 287–296, 2006.
17. S. Winkler, H. Sato, A. Middeldorp, and M. Kurihara. Optimizing mkbTT (system description). In *Proc. 21st RTA*, LIPIcs, 2010. To appear.
18. H. Zantema. Total termination of term rewriting is undecidable. *Journal of Symbolic Computation*, 20(1):43–60, 1995.