# Multi-Completion with Termination Tools (System Description)*

Haruhiko Sato[1] and Sarah Winkler[2] and
Masahito Kurihara[1] and Aart Middeldorp[2]

[1] Graduate School of Information Science and Technology
Hokkaido University, Japan
[2] Institute of Computer Science
University of Innsbruck, Austria

**Abstract.** In this paper we describe a new tool for performing Knuth-Bendix completion with automatic termination tools. It is based on two ingredients: (1) the inference system for completion with multiple reduction orderings introduced by Kurihara and Kondo (1999) and (2) the inference system for completion with external termination provers proposed by Wehrman, Stump and Westbrook (2006) and implemented in the Slothrop system. Our tool can be used with any termination tool that satisfies certain minimal requirements. Preliminary experimental results show the potential of our tool.

## 1 Introduction

Knuth and Bendix [2] introduced in a landmark paper the completion procedure which aims to transform a given set of equations into a confluent and terminating rewrite system, which can then be used to decide validity problems. The procedure takes as input a reduction order which is used to orient rules. The success of the procedure depends very much on the choice of the reduction order.

Kurihara and Kondo [3] introduced a variant that works for multiple reduction orders. It simulates the parallel execution of completion processes with the individual orders. The common features of the different processes are captured in a special data structure and corresponding inference rules. The resulting procedure is more efficient than a parallel execution of completion procedures.

Wehrman, Stump and Westbrook [7] take a different approach. Instead of relying on a reduction order supplied by the user, they call an external termination prover to orient rules. Since modern termination provers typically combine a number of different powerful techniques, this opens the way to complete systems that cannot be handled by traditional implementations of the completion procedure. One such system is $\mathrm{CGE}_2$, the theory of two commuting group endomorphisms, that is completed by the Slothrop tool described in [7] without user interaction.

* This research is partially supported by FWF (Austrian Science Fund) project P18763 and JSPS Grant-in-Aid for Scientific Research (C), No. 19500020.

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

**Fig. 1.** Inference rule orient of KBtt.

We combine the two approaches sketched above in a single procedure. The underlying inference system is described in the next section. In Section 3 we present implementation details and in Section 4 we present the interface of our tool. Preliminary experimental results given in Section 5 show the advantage of our approach.

## 2 Inference System

Figure 1 shows the orient inference rule of KBtt ($\mathcal{A}$ in [7]), the inference system underlying SLOTHROP. KBtt operates on triples $(E, R, C)$ consisting of unoriented equations in $E$ and rewrite rules in $R$ and $C$.

In the orient rule, termination is checked of the combination of the new rewrite rule $s \to t$ and all previously added rewrite rules, which are stored in the third component $C$. (The other inference rules do not modify $C$.) Checking termination of the combination of $s \to t$ and the present rules in $R$ would be unsound [4]. When both $C \cup \{s \to t\}$ and $C \cup \{t \to s\}$ can be proved terminating, in an implementation one faces the question which branch to explore. SLOTHROP uses a best-first search strategy in connection with a cost function that determines which branch to advance [7, Section 4].

The advantage of our approach is that both branches are explored simultaneously by incorporating the ideas of Kurihara and Kondo [3]. The completion procedure described in their paper simulates the execution of multiple completion processes in parallel. Whenever a process encounters an equation that can be oriented in either direction, the process is split into two child processes. As a process corresponds to a sequence of decisions on how to orient equations, processes will in the following be considered as bit strings. The set of all processes will be denoted by $\mathcal{P}$ and the initial process is naturally represented by the empty string $\epsilon$.

**Definition 1.** *The inference system MKBtt operates on sets of* nodes*. A node $\langle s : t, R_1, R_2, E, C_1, C_2 \rangle$ consists of an ordered pair of terms $s : t$ and sets of processes $R_1, R_2, E, C_1, C_2 \subseteq \mathcal{P}$ such that $R_1$, $R_2$, and $E$ are mutually disjoint. A node $\langle s : t, R_1, R_2, E, C_1, C_2 \rangle$ is identified with $\langle t : s, R_2, R_1, E, C_2, C_1 \rangle$. Given a set of equations $\mathcal{F}$, the initial node set consists of all nodes $\langle s : t, \varnothing, \varnothing, \{\epsilon\}, \varnothing, \varnothing \rangle$ such that $s \approx t$ occurs in $\mathcal{F}$. The inference rules of MKBtt are displayed in Figure 2.*

The term pair $s : t$ is also referred to as the node's *datum*, the remaining components as its *labels*. Intuitively, the sets $R_1$ and $C_1$ consist of processes where

orient
$$\frac{N \cup \{\langle s : t, R_1, R_2, E, C_1, C_2\rangle\}}{\mathrm{split}_P(N) \cup \{\langle s : t, R_1 \cup R_{lr}, R_2 \cup R_{rl}, E', C_1 \cup R_{lr}, C_2 \cup R_{rl}\rangle\}}$$

with $E_{lr}, E_{rl} \subseteq E$ such that $E_{lr} \cup E_{rl} \neq \varnothing$, $P = E_{lr} \cap E_{rl}$, $E' = E \setminus (E_{lr} \cup E_{rl})$, $C[N, p] \cup \{s \to t\}$ terminates for all $p \in E_{lr}$, $C[N, p] \cup \{t \to s\}$ terminates for all $p \in E_{rl}$, $R_{lr} = (E_{lr} \setminus E_{rl}) \cup \{p0 \mid p \in P\}$, $R_{rl} = (E_{rl} \setminus E_{lr}) \cup \{p1 \mid p \in P\}$ and where $\mathrm{split}_P(N)$ replaces every $p \in P$ in any label of any node in $N$ by $p0$ and $p1$

delete
$$\frac{N \cup \{\langle s : s, \varnothing, \varnothing, E, \varnothing, \varnothing\rangle\}}{N}$$

deduce
$$\frac{N}{N \cup \{\langle s : t, \varnothing, \varnothing, R \cap R', \varnothing, \varnothing\rangle\}}$$

if there exist nodes $\langle l : r, R, \dots\rangle, \langle l' : r', R', \dots\rangle \in N$ and a term $u$ such that $s \leftarrow_{l \to r} u \to_{l' \to r'} t$ and $R \cap R' \neq \varnothing$

rewrite$_1$
$$\frac{N \cup \{\langle s : t, R_1, R_2, E, C_1, C_2\rangle\}}{\begin{array}{c} N \cup \{\langle s : t, R_1 \setminus R, R_2, E \setminus R, C_1, C_2\rangle\} \\ \cup \{\langle s : u, R_1 \cap R, \varnothing, E \cap R, C_1, C_2\rangle\}\end{array}}$$

if $\langle l : r, R, \dots\rangle \in N$, $t \to_{l \to r} u$, $t \doteq l$, and $R \cap (R_1 \cup E) \neq \varnothing$ [a]

rewrite$_2$
$$\frac{N \cup \{\langle s : t, R_1, R_2, E, C_1, C_2\rangle\}}{\begin{array}{c} N \cup \{\langle s : t, R_1 \setminus R, R_2 \setminus R, E \setminus R, C_1, C_2\rangle\} \\ \cup \{\langle s : u, R_1 \cap R, \varnothing, (R_2 \cup E) \cap R, \varnothing, \varnothing\rangle\}\end{array}}$$

if $\langle l : r, R, \dots\rangle \in N$, $t \to_{l \to r} u$, $t \rhd l$, and $R \cap (R_1 \cup R_2 \cup E) \neq \varnothing$ [b]

gc
$$\frac{N \cup \{\langle s : t, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing\rangle\}}{N}$$

subsume
$$\frac{N \cup \{\langle s : t, R_1, R_2, E, C_1, C_2\rangle\} \cup \{\langle s' : t', R_1', R_2', E', C_1', C_2'\rangle\}}{N \cup \{\langle s : t, R_1 \cup R_1', R_2 \cup R_2', E'', C_1 \cup C_1', C_2 \cup C_2'\rangle\}}$$

if $s : t$ and $s' : t'$ are variants and $E'' = (E \setminus (R_1' \cup R_2' \cup C_1' \cup C_2')) \cup (E' \setminus (R_1 \cup R_2 \cup C_1 \cup C_2))$

[a] $t \doteq l$ specifies that $t$ and $l$ are variants

[b] $\rhd$ denotes the encompassment relation

**Fig. 2.** Inference rules of MKBtt.

$s \to t$ is contained in the current set of rules and the current set of constraints for this process, respectively. The sets $R_2$ and $C_2$ play the analogous role with respect to the rule $t \to s$. The set $E$ contains processes where $s \approx t$ is in the current set of equations. In the following we make some clarifying remarks on some of the inference rules.

– Unlike its KB counterpart, the orient rule does not orient an equation in just one direction. Instead, for the chosen node $n = \langle s : t, R_1, R_2, E, C_1, C_2 \rangle$ one checks for every process $p$ occurring in the node's equation label $E$ if the constraint system for $p$ remains terminating if either $s \to t$ or $t \to s$ is added. Formally

$$C[N,p] = \bigcup_{n \in N} C[n,p] \quad \text{with} \quad C[n,p] = \begin{cases} \{s' \to t'\} & \text{if } p \in C_1' \\ \{t' \to s'\} & \text{if } p \in C_2' \\ \varnothing & \text{otherwise} \end{cases}$$

denotes the set of constraints for process $p$, where for every $p \in E$ it is determined whether $C[N,p]$ terminates in combination with $s \to t$ or $t \to s$. If only one orientation is possible, $p$ is moved to the respective rule label in the node. If both orientations are possible, we have to keep track of both alternatives. Thus the process splits into two child processes $p0$ and $p1$. Each of them is put into the corresponding rule label. Moreover, $\text{split}_P(N)$ replaces $p$ by its derivatives $p0$ and $p1$ in all non-selected nodes.

– If $l \to r$ and $l' \to r'$ allow for a peak $s \leftarrow_{l \to r} u \to_{l' \to r'} t$, deduce adds the equation $s \approx t$ to those processes that have both rules present in their rule set. In our implementation, only critical pairs are considered.

– Given a term pair $s : t$ and a rewrite step $t \to_R u$, the inference rules compose, simplify, and collapse of KBtt create an equation or rule with terms $s$ and $u$. The effect of these rules is simulated by $\mathsf{rewrite}_1$ and $\mathsf{rewrite}_2$, as explained in [3].

It is not difficult to state and prove (partial) correctness and completeness criteria for MKBtt by relating MKBtt to KBtt, akin to [3, Section 2.2].

## 3 Implementation

In our implementation of MKBtt the inference rules of Figure 2 are employed according to the strategy described in Figure 3, closely resembling the algorithm proposed in [3]. The mkbTT procedure maintains two node sets, *No* containing *open* and *Nc* containing *closed* nodes. The union $No \cup Nc$ represents all nodes present in the completion process. Intuitively, every node in *Nc* has been completely exploited with respect to the inference rules orient, delete, and gc. Moreover, every pair of nodes in *Nc* has been fully exploited with respect to the inference rules that involve two nodes: deduce, $\mathsf{rewrite}_{1,2}$ and subsume. Initially, *No* contains all nodes and *Nc* is empty. Below we shortly comment on the functions involved in mkbTT and their implementation.

– At the start of every recursive call of mkbTT, it is checked whether some process $p$ was successful. This is the case if every equation was oriented and every rule was fully considered with respect to the inferences involving two nodes, i.e., if all of $E[Nc,p]$, $R[No,p]$, and $E[No,p]$ are empty.

– If no successful process was found, *choose* selects an open node. The measure applied in this selection has considerable impact on the overall performance.

```
procedure $mkbTT(No,\ Nc)$
if $success$ then            return successful $p$
else if $No = \varnothing$ then       $fail$
else $n := choose(No)$;
    $No = add(delete(rewrite(\{n\}, Nc)), No \setminus \{n\})$;
    if $n \neq \langle \dots, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing \rangle$ then
      $(n, No, Nc) := orient(n, No, Nc)$;
      if $n \neq \langle \dots, \varnothing, \varnothing, \dots, \dots, \dots \rangle$ then
        $No := add(delete(rewrite(Nc, \{n\})), No)$; $Nc := gc(Nc)$;
        $No := add(delete(deduce(n, Nc)), No)$;
        $No := gc(add(rewrite(No, Nc), No))$;        $Nc := add(\{n\}, Nc)$;
    $mkbTT(No, Nc)$;
```

**Fig. 3.** Procedure implementing MKBtt.

In our implementation we first choose a process $p$ for which $|E[Nc \cup No, p]| + |R[Nc \cup No, p]|$ is minimal and then a node for this process by considering the term size and timestamp, the latter to ensure fairness of the derivation.

- $rewrite(N, N')$ applies $\mathsf{rewrite}_{1,2}$ to nodes in $N$ by using rules in $N'$. Nodes are considered as mutable structures. Thus the node objects in $N$ are modified and only newly created nodes are returned.
- Immediately after rewriting, *delete* is called, applying the corresponding inference rule to avoid creating nodes with equal terms.
- $orient(n, No, Nc)$ applies the inference rule orient to $n$ and, if required, splits processes occurring in labels of $No$ and $Nc$. The modified node $n$ and the node sets $No$ and $Nc$ are returned.
- $gc(N)$ removes nodes from $N$ where all labels are empty.
- $deduce(n, N)$ returns nodes derived from the respective inference such that at least one rule comes from $n$.
- $add(N, N')$ merges the nodes in $N$ into $N'$ such that subsume is fully exploited. Only inferences where at least one node comes from $N$ have to be considered.

## 4 Interface

The mkbTT procedure is implemented in OCaml. A binary compiled for Linux is available from `http://cl-informatik.uibk.ac.at/mkbtt`. The tool is equipped with a simple command-line interface. The termination prover is given as argument to the `-tp` option. Any termination prover that adheres to the format of the International Competitions of Termination Tools[3] can be used: an executable that takes as argument the name of a file describing the termination problem in the TPDB[4] format and prints `YES` on the first line of the output if termination

[3] `http://www.lri.fr/~marche/termination-competition/`

[4] Termination Problem Data Base, `http://www.lri.fr/~marche/tpdb/`

```
SUCCESS
246.64 (total time)

STATISTICS
number of inference steps: 77
orient:      218.09          rewrite:      19.25
deduce:        2.89          termination: 209.64

external termination prover: ttt2fast
calls to termination prover: 1072 (yes: 933, timeouts: 0)
time limit per call:         1.0
```

**Fig. 4.** Sample output (slightly reformatted).

could be established. Our tool accepts two time limits: for the overall procedure (specified with `-t`) and for each call to the termination prover (`-T`). Further options are `-ct` to print the completed system and `-st` to obtain some useful statistics. Figure 4 shows the output for the call

```
mkbtt -t 3600 -T 1 -st -tp ttt2fast WSW06_CGE2.trs
```

## 5 Experimental Results

In Table 1 we present some experimental data.[5] All tests were performed on a workstation equipped with an Intel® Pentium™ M processor running at a CPU rate of 2 GHz on 1 GB of system memory and with a time limit of 1 hour. Column (1) shows the total time in seconds, column (2) the percentage spent on termination, column (3) the number of calls to the external termination prover, and column (4) the number of inference steps. A timeout is indicated by $\infty$.

In the first SLOTHROP and MKBTT blocks, AProVE [1] is used as termination prover with a time limit of 5 seconds per call. (We modified the function for termination checks in the SLOTHROP source[6] such that the same back-end can be used with both approaches.) In the second SLOTHROP and MKBTT blocks we use a special version of $\mathsf{T_TT_2}$[7] with a time limit of 1 second per call. This version of $\mathsf{T_TT_2}$, which uses dependency pairs and the recursive SCC algorithm together with the subterm criterion and some simple strategies like counting function symbols and linear polynomials, is considerably weaker than AProVE when it comes to termination proving power, but also considerably faster. We anticipate that further savings can be achieved by more tightly coupling the termination prover and the MKBTT code.

As can be seen from line SL-cge2 in Table 1, completing the theory of two communicating group endomorphism (CGE$_2$), which is termed SLOTHROP*'s*

[5] Further details can be obtained from `http://cl-informatik.uibk.ac.at/mkbtt`.
[6] `http://www.cs.utexas.edu/~iwehrman/slothrop-1.1.0-src.tar.gz`
[7] `http://colo6-c703.uibk.ac.at/ttt2`

**Table 1.** Experimental Results

| | Slothrop | | | MKBtt | | | | Slothrop | | | MKBtt | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TRS | (1) | (2) | (3) | (1) | (2) | (3) | (4) | (1) | (2) | (3) | (1) | (2) | (3) | (4) |
| SK90_3.01 | 800.37 | 97 | 326 | 85.30 | 99 | 51 | 29 | 71.52 | 67 | 304 | 4.45 | 79 | 51 | 29 |
| SK90_3.03 | 163.51 | 98 | 86 | 90.97 | 98 | 53 | 29 | 9.97 | 65 | 86 | 6.19 | 63 | 53 | 29 |
| SK90_3.04 | $\infty$ | | | $\infty$ | | | | $\infty$ | | | 508.24 | 55 | 658 | 140 |
| SK90_3.05 | $\infty$ | | | 913.77 | 98 | 225 | 94 | 78.48 | 32 | 258 | 46.45 | 43 | 220 | 92 |
| SK90_3.06 | $\infty$ | | | $\infty$ | | | | $\infty$ | | | 44.22 | 74 | 290 | 75 |
| SK90_3.07 | $\infty$ | | | 513.27 | 99 | 242 | 67 | $\infty$ | | | 46.03 | 68 | 282 | 77 |
| SK90_3.19 | 84.25 | 99 | 43 | 112.65 | 99 | 65 | 11 | 3.39 | 90 | 43 | 5.01 | 96 | 65 | 11 |
| SK90_3.22 | $\infty$ | | | $\infty$ | | | | $\infty$ | | | 617.33 | 33 | 1406 | 141 |
| SK90_3.27 | $\infty$ | | | $\infty$ | | | | 73.61 | 95 | 70 | 118.56 | 65 | 143 | 37 |
| SL-ack | 12.08 | 99 | 8 | 13.26 | 99 | 10 | 5 | 0.24 | 96 | 8 | 0.33 | 91 | 10 | 5 |
| SL-cge2 | $\infty$ | | | 2793.21 | 99 | 1220 | 77 | 665.29 | 36 | 1384 | 246.64 | 85 | 1072 | 77 |
| SL-cge3 | $\infty$ | | | $\infty$ | | | | $\infty$ | | | $\infty$ | | | |
| SL-endo | 246.56 | 95 | 101 | 218.96 | 99 | 135 | 45 | 12.64 | 39 | 105 | 7.87 | 74 | 135 | 45 |
| SL-ep | $\infty$ | | | $\infty$ | | | | 54.47 | 82 | 266 | 230.06 | 92 | 1101 | 26 |
| SL-groups | 46.71 | 97 | 30 | 96.94 | 99 | 49 | 35 | 2.10 | 46 | 30 | 2.28 | 71 | 49 | 35 |

*defining achievement* in [6], takes about 246 seconds using MKBtt with the fast variant of $\mathsf{T_TT_2}$. We also managed to complete CGE$_3$, which has not been achieved with Slothrop, although it takes nearly 2 hours. (The completed system is slightly different from the one described in [5], which was obtained by hand.)

## References

1. J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In *Proc. 3rd IJCAR*, volume 4130 of *LNAI*, pages 281–286, 2006.
2. 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.
3. M. Kurihara and H. Kondo. Completion for multiple reduction orderings. *Journal of Automated Reasoning*, 23(1):25–42, 1999.
4. A. Sattler-Klein. About changing the ordering during Knuth-Bendix completion. In *Proc. 11th STACS*, volume 775 of *LNCS*, pages 175–186, 1994.
5. A. Stump and B. Löchner. Knuth-Bendix completion of theories of commuting group endomorphisms. *Information Processing Letters*, 98(5):195–198, 2006.
6. I. Wehrman. Knuth-Bendix completion with modern termination checking. Master's thesis, Washington University in St. Louis, 2006. Technical report WUCSE-2006-45.
7. 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.