AC Completion with Termination Tools*

Sarah Winkler and Aart Middeldorp

Institute of Computer Science, University of Innsbruck, Austria

Abstract. We present mascott, a tool for Knuth-Bendix completion modulo the theory of associative and commutative operators. In contrast to classical completion tools, mascott does not rely on a fixed AC-compatible reduction order. Instead, a suitable order is implicitly constructed during a deduction by collecting all oriented rules in a similar fashion as done in the tool Slothrop. This allows for convergent systems which cannot be completed using standard orders. We outline the underlying inference system and comment on implementation details such as the use of multi-completion, term indexing techniques, and critical pair criteria.

1 Introduction

Reasoning modulo an equational theory A is required in many practical problems. The generalization of the classical Knuth-Bendix completion algorithm to rewriting modulo A is well-known (see [3] for an overview). Like ordinary completion, completion modulo A critically depends on the choice of the A-compatible reduction order supplied as input. In this system description we show how the use of termination tools supporting termination modulo A can replace a fixed reduction order, in a similar fashion as proposed by the authors of the tool Slothrop [20]. Recent developments in the area of termination proving can thus be directly exploited to obtain convergent systems for theories which were difficult to complete before. Our method can be combined with the multi-completion approach proposed by Kondo and Kurihara [14]. For equational theories A consisting of AC axioms, this approach is implemented in our new tool mascott. Our contribution can thus be viewed as an extension of the completion tool $\mathsf{mkb_{TT}}$ [19, 21] to AC theories. As an example, mascott successfully completes the following system (adapted from [17]) describing addition on natural numbers represented in binary:

$$\begin{array}{ccc} \#0 \simeq \# & (x+y)1 \simeq x0 + y1 & \mathsf{triple}(x) \simeq x0 + x \\ (x+y)0 \simeq x0 + y0 & x0 + y0 + \#10 \simeq x1 + y1 & \end{array}$$

Here $+$ is an AC operator, $\mathsf{0}$ and $\mathsf{1}$ are unary operators in postfix notation, and $\#$ denotes the empty bit sequence. For example, $\#100$ represents the number 4 in binary. The following completed system is obtained when using (e.g.) AProVE [11] as a termination prover, but cannot be shown terminating by any

* The first author is supported by a DOC-fFORTE fellowship of the Austrian Academy of Sciences.

standard AC-compatible simplification order:

$$\begin{array}{rcl}
\#0 \to \# & \qquad & \mathsf{triple}(x) \to x0 + x \\
(x + \#)0 \to x0 + \# & & (x + \#)1 \to x1 + \# \\
x0 + y0 \to (x + y)0 & & x0 + y0 + z \to (x + y)0 + z \\
x0 + y1 \to (x + y)1 & & x0 + y1 + z \to (x + y)1 + z \\
x1 + y1 \to (x + y + \#1)0 & & x1 + y1 + z \to (x + y + \#1)0 + z
\end{array}$$

Our tool mascott can be accessed via a simple web interface.[1] The sources and a binary are available as well. In the sequel we outline the underlying inference system, describe the implementation, and give some (preliminary) experimental results for mascott.

2 Inference System

We assume familiarity with term rewriting and Knuth-Bendix completion, and recall only some central notions. We consider a rewrite system R and a set of equations A. A term s rewrites to t in R modulo A, denoted by $s \to_{R/A} t$, whenever $s \leftrightarrow_A^* \cdot \to_R \cdot \leftrightarrow_A^* t$. The system R *terminates modulo* A whenever the relation $\to_{R/A}$ is well-founded. It is *convergent modulo* A if in addition for every conversion $s \leftrightarrow_{A \cup R}^* t$ there exist terms u and v such that $s \to_R^* u \leftrightarrow_A^* v \leftarrow_R^* t$. To check termination of R modulo A, A-compatible reduction orders $>$ satisfying $\leftrightarrow_A^* \cdot > \cdot \leftrightarrow_A^* \subseteq >$ can be used. Since the relation $\to_{R/A}$ is undecidable in general, one typically considers the rewrite system R_A consisting of all rules $s \to t$ such that $s \leftrightarrow_A^* \ell\sigma$ and $t = r\sigma$ for some rule $\ell \to r$ in R and substitution σ. We obviously have $\to_R \subseteq \to_{R_A} \subseteq \to_{R/A}$. Thus, if R is convergent modulo A then also R_A is convergent modulo A [3], and defines the same normal forms as R/A. Hence rewriting using rules in R_A constitutes a decidable way to compute with respect to R/A.

We confine our analysis to theories A for which minimal sets of complete unifiers are computable, and denote by $CP_A(R)$ the set of A-critical pairs among rules in R.[2] For a rule $\ell \to r$ and a variable-disjoint equation $u \simeq v$ in A such that a proper non-variable subterm $u|_p$ of u and ℓ are A-unifiable, $u[\ell]_p \to u[r]_p$ is an A-extended rule [18]. The set of A-extended rules of R is denoted by $EXT_A(R)$.

Our tool is based on a variant of the inference system $\mathcal{E}$ for extended completion developed by Bachmair [3, Chapter 3]. In order to get rid of a fixed reduction order and have termination checks as side conditions, the system was modified to resemble the calculus underlying Slothrop [20]. The inference rules thus operate on a set of equations E, a set of rewrite rules R partitioned into unprotected rules N and protected rules S, and a constraint system C. The resulting inference system $\mathcal{E}_{\mathsf{TT}}$ for completion modulo the theory A is depicted in Figure 1.

[1] `http://cl-informatik.uibk.ac.at/software/mascott`

[2] Although our tool is restricted to the theory of associative and commutative operators, the underlying inference system is presented for arbitrary theories A that satisfy the stated condition.

deduce	$\dfrac{E, N, S, C}{E \cup \{s \simeq t\}, N, S, C}$	if $s \leftrightarrow^*_{A \cup R} t$
extend	$\dfrac{E, N, S, C}{E, N, S \cup \{s \to t\}, C}$	if $s \simeq t \in \mathit{EXT}_A(R)$
orient	$\dfrac{E \cup \{s \simeq t\}, N, S, C}{E, N \cup \{s \to t\}, S, C \cup \{s \to t\}}$	if $C \cup \{s \to t\}$ terminates modulo A
protect	$\dfrac{E, N \cup \{s \to t\}, S, C}{E, N, S \cup \{s \to t\}, C}$	
delete	$\dfrac{E \cup \{s \simeq t\}, N, S, C}{E, N, S, C}$	if $s \leftrightarrow^*_A t$
simplify	$\dfrac{E \cup \{s \simeq t\}, N, S, C}{E \cup \{s \simeq u\}, N, S, C}$	if $t \to_{R/A} u$
compose	$\dfrac{E, N \cup \{s \to t\}, S, C}{E, N \cup \{s \to u\}, S, C}$	if $t \to_{R/A} u$
	$\dfrac{E, N, S \cup \{s \to t\}, C}{E, N, S \cup \{s \to u\}, C}$	if $t \to_{R/A} u$
collapse	$\dfrac{E, N \cup \{t \to s\}, S, C}{E \cup \{u \simeq s\}, N, S, C}$	if $t \leftrightarrow^{\leqslant p}_A t' \to^p_{\ell \to r} u$ for some rule $\ell \to r$ in R with $t \gg \ell$

Fig. 1. System $\mathcal{E}_{\mathsf{TT}}$ of extended completion with termination checks.

Here $\gg$ is some well-founded order on terms such as the encompassment order[3] and the relation $\simeq$ is assumed to be symmetric.

A sequence $(E_0, \varnothing, \varnothing, \varnothing) \vdash (E_1, N_1, S_1, C_1) \vdash (E_2, N_2, S_2, C_2) \vdash \cdots$ of inference steps in $\mathcal{E}_{\mathsf{TT}}$ is called a *run*. Note that orient is the only inference rule which actually modifies the set C of constraint rules. Since an A-termination check is performed whenever a rule is added, all constraint systems C_n are terminating modulo A. Hence the transitive closure of the rewrite relation $\to^+_{C_n/A}$ is an A-compatible reduction order, so runs in $\mathcal{E}_{\mathsf{TT}}$ can be simulated in $\mathcal{E}$:

Lemma 1.

1. *For every finite run $(E_0, \varnothing, \varnothing, \varnothing) \vdash^* (E_n, N_n, S_n, C_n)$ in $\mathcal{E}_{\mathsf{TT}}$ there is a corresponding run $(E_0, \varnothing, \varnothing) \vdash^* (E_n, N_n, S_n)$ in $\mathcal{E}$ using the A-compatible reduction order $\to^+_{C_n/A}$.*
2. *Every run $(E_0, \varnothing, \varnothing) \vdash^* (E_n, N_n, S_n)$ in $\mathcal{E}$ using an A-compatible reduction order $>$ can be simulated in an $\mathcal{E}_{\mathsf{TT}}$ run $(E_0, \varnothing, \varnothing, \varnothing) \vdash^* (E_n, N_n, S_n, C_n)$ such that $C_n \subseteq >$ holds.* □

[3] in which s is greater than t if a subterm of s is an instance of t but not vice versa.

The straightforward induction proofs closely resemble the respective counterparts for standard completion and are thus omitted. Since our implementation is restricted to the theory AC of associative and commutative operators in $\mathcal{F}_{AC}$, we will now focus on this setting. Let R^e denote the rewrite system containing R, extended with all rules of the form $f(\ell, x) \to f(r, x)$ such that $f \in \mathcal{F}_{AC}$, $\ell \to r \in R$ and x is a fresh variable.

Corollary 1. *If a non-failing finite $\mathcal{E}_{\mathsf{TT}}$ run $(E_0, \varnothing, \varnothing, \varnothing) \vdash^* (\varnothing, N_n, S_n, C_n)$ satisfies $CP_{AC}(R_n) \subseteq \bigcup_i E_i$ and $(R_n)^e \subseteq \bigcup_i S_i$ then $(R_n)_{AC}$ is convergent modulo AC.* $\square$

3 Implementation

In this section we present some implementation details of mascott, which stands for multi-associative/commutative completion with termination tools.

If an equation $s \simeq t$ can be oriented in both directions, the orient rule in $\mathcal{E}_{\mathsf{TT}}$ allows for a choice. In order not to restrict to one orientation, we adapted the multi-completion approach proposed by Kondo and Kurihara [14] to the setting of completion modulo a theory A. Similar to standard completion the obtained method can be described by an inference system operating on sets of nodes $\mathcal{N}$, which are defined as in [19], the difference being that a rewrite label R_i is now split into unprotected and protected labels (N_i, S_i). Figure 2 shows the inference rules orient and extend which are specific to completion modulo A.

As an example, on input $\{\mathsf{d}(\mathsf{s}(x){+}y) \simeq \mathsf{d}(\mathsf{p}(\mathsf{s}(x)){+}y), \mathsf{p}(\mathsf{s}(\mathsf{s}(x))) \simeq \mathsf{s}(\mathsf{p}(\mathsf{s}(x)))\}$ with $+$ an AC symbol, any completion procedure using standard AC-compatible simplification orders orients the first equation from right to left, causing divergence of the procedure. In contrast, our tool keeps track of both orientations and immediately outputs the AC-convergent system obtained when orienting both rules from left to right.

The termination checks required in orient inference steps may be performed by an external tool supporting AC termination such as AProVE or muterm [1]. Alternatively, a modified version of T$_\mathsf{T}$T$_2$ [13] can be used internally, supporting AC-dependency pairs [10, 17, 2] and reduction pairs induced by polynomial or matrix interpretations. A criterion for AC-compatibility of polynomial interpretations was given in [5]. It is not difficult to check that matrix interpretations [7] are AC-compatible if every AC symbol f is interpreted as $f_{\mathcal{M}}(x, y) = Ax{+}By{+}b$ where the square matrices A and B satisfy $A = A^2 = B$ in addition to the usual constraint that the top-left entry of A is positive.

In order to limit the number of equational consequences, only *prime* critical pairs are computed [12]. For AC-unification, the algorithms proposed in [16, 8] were used, in the latter case incorporating the SMT solver Yices to solve linear Diophantine equations. For rewriting, AC-discrimination trees allow for a fast pre-selection of matching rules [4].

The tool is equipped with a simple command-line interface. The termination prover is given as argument to the `-tp` option. It is supposed to take the name of orient $$\frac{\mathcal{N} \cup \{\ \langle s : t, (N_0, S_0), (N_1, S_1), E, C_0, C_1\rangle\ \}}{\text{split}_P(\mathcal{N}) \cup \{\ \langle s : t, (N_0 \cup R_{lr}, S_0), (N_1 \cup R_{rl}, S_1), E', C_0 \cup R_{lr}, C_1 \cup R_{rl}\rangle\ \}}$$

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 modulo A for all $p \in E_{lr}$, $C[N, p] \cup \{t \to s\}$ terminates modulo A for all $p \in E_{rl}$, $R_{lr} = (E_{lr} \setminus E_{rl}) \cup \{p0 \mid p \in P\}$ and $R_{rl} = (E_{rl} \setminus E_{lr}) \cup \{p1 \mid p \in P\}$ where $\text{split}_P(N)$ replaces every $p \in P$ in any label of a node in $\mathcal{N}$ by $p0$ and $p1$

extend $$\frac{\mathcal{N}}{\mathcal{N} \cup \{\ \langle \ell' : r', (\varnothing, N_0 \cup S_0), (\varnothing, \varnothing), \varnothing, \varnothing, \varnothing\rangle\ \}}$$

if $\langle \ell : r, (N_0, S_0), \dots\rangle \in \mathcal{N}$, $\ell' \to r' \in EXT_A\{\ell \to r\}$ and $N_0 \cup S_0 \neq \varnothing$

Fig. 2. Two inference rules for multi-completion modulo A.

a file describing the termination problem in the TPDB[4] format and print `YES` on the first line of the output if termination modulo AC could be established. Our tool accepts two time limits: for the overall procedure (`-t`) and for each call to the termination prover (`-T`). The option `-cp prime` allows to apply primality as a critical pair criterion. Further options are `-ct` to print the completed system and `-st` to obtain some statistics. An example call might thus look as follows:

```
mascott -t 300 -T 1 -st -tp muterm binary_arithmetic.trs
```

4 Experiments

For our experiments we collected AC completion problems from a number of different sources and ran mascott with different termination provers as backends. All of the tests were performed on an Intel Core Duo running at a clock rate of 1.4 GHz with 2.8 GB of main memory.

The results are summarized in Table 1, where the superscripts attached to the problems indicate their source: *a* refers to [9], *b* refers to [17], and *c* is associated with [15]. The remaining examples were added by the authors. Columns (1) list the total time in seconds while columns (2) give the percentage of time spent on termination. The symbol ∞ marks a timeout of 300 seconds. For internal termination checks a termination strategy employing dependency pairs and matrix interpretations was used. As expected, this strategy is far less powerful than the techniques used by AProVE or muterm.

We also include a comparison with C*i*ME [6], the only other current tool for AC completion that we are aware of, although this requires the specification of a concrete AC-RPO or AC-compatible polynomial interpretation by the user. For our experiments we supplied an appropriate order whenever possible, and in

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

	masco*tt*						C*i*ME
	internal		AProVE		muterm		
	(1)	(2)	(1)	(2)	(1)	(2)	(1)
Abelian groups (AG)[a]	14.48	63	9.33	70	3.02	7	0.05
AG + homomorphism	169.62	87	73.87	84	30.20	15	0.05
arithmetic[a]	∞		24.58	47	35.03	8	?
AC-ring with unit[a]	∞		64.15	53	55.96	38	0.1
associative ring with unit[a]	∞		∞		163.96	71	0.1
binary arithmetic[b]	∞		78.36	89	23.94	20	?
commutative monoid[a]	0.5	2	0.7	95	0.03	32	0.01
example 5.4.2[c]	8.26	97	5.94	98	0.39	79	0.01
example from Section 3	∞		∞		0.74	91	?
ICS[a]	12.75	7	9.03	34	6.10	1	0.01
max[c]	∞		8.34	98	0.29	58	?
multisets over $\{0, 1\}$	∞		117.11	96	9.76	52	?
nondeterministic machine[a]	∞		∞		∞		0.2
ring[a]	∞		224.99	75	125.88	67	0.07
ring with unit[a]	∞		201.11	76	81.94	62	0.1
semiring[a]	∞		24.90	75	12.35	45	0.1
semilattice[a]	10.12	4	5.66	12	5.33	1	0.01
sum	∞		7.58	98	0.33	54	?
completed systems	6		13		17		12
average time for success	33.54		58.60		42.71		0.07

Table 1. Comparison of masco*tt* using different termination backends and C*i*ME.

these cases C*i*ME completed the given problems considerably faster. However, a suitable order does not always exist (as for the example mentioned in Section 3) or is not known (as for the binary addition example from the introduction or the arithmetic problem). In Table 1 the symbol ? marks these cases.

In line with previous work on AC completion, the use of critical pair criteria turned out to be highly beneficial. Restricting to so-called prime critical pairs increases performance on the examples from Table 1 by 40%. For example, without the criterion the theory of associative rings with unit cannot be completed within 300 seconds.

5 Conclusion

Apparently, masco*tt* is the only tool for AC completion which is automatic in that it does not require a fixed reduction order as input. To the best of our knowledge, masco*tt* is also the first AC completion tool not restricted to AC-RPO or AC-compatible polynomial interpretations as termination methods. Instead, all techniques developed for AC termination can be exploited, such as the dependency pair framework or matrix interpretations. Our tool is thus able to produce novel complete systems such as the one mentioned in the introduction.

References

1. B. Alarcón, R. Gutiérrez, S. Lucas, and R. Navarro-Marset. Proving termination properties with MU-TERM. In *Proc. 13th AMAST*, volume 6486 of *LNCS*, pages 201–208, 2011.
2. B. Alarcón, S. Lucas, and J. Meseguer. A dependency pair framework for $A \vee C$-termination. In *Proc. 8th WRLA*, volume 6381 of *LNCS*, pages 35–51, 2010.
3. L. Bachmair. *Canonical Equational Proofs*. Progress in Theoretical Computer Science. Birkhäuser, 1991.
4. L. Bachmair, T. Chen, and I.V. Ramakrishnan. Associative-commutative discrimination nets. In *Proc. 5th TAPSOFT*, volume 668 of *LNCS*, pages 61–74, 1993.
5. A. Ben Cherifa and P. Lescanne. Termination of rewriting systems by polynomial interpretations and its implementation. *SCP*, 9(2):137–159, 1987.
6. E. Contejean and C. Marché. C*i*ME: Complet*i*on modulo *E*. In *Proc. 7th RTA*, volume 1103 of *LNCS*, pages 416–419, 1996.
7. J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. *JAR*, 40(2-3):195–220, 2008.
8. A. Fortenbacher. An algebraic approach to unification under associativity and commutativity. *JSC*, 3(3):217–229, 1987.
9. W. Gehrke. Detailed catalogue of canonical term rewrite systems generated automatically. Technical report, RISC Linz, 1992.
10. J. Giesl and D. Kapur. Dependency pairs for equational rewriting. In *Proc. 12th RTA*, volume 2051 of *LNCS*, pages 93–108, 2001.
11. 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.
12. D. Kapur, D.R. Musser, and P. Narendran. Only prime superpositions need be considered in the Knuth-Bendix completion procedure. *JSC*, 6(1):19–36, 1988.
13. 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.
14. M. Kurihara and H. Kondo. Completion for multiple reduction orderings. *JAR*, 23(1):25–42, 1999.
15. K. Kusakari. *AC-Termination and Dependency Pairs of Term Rewriting Systems*. PhD thesis, JAIST, 2000.
16. P. Lincoln and J. Christian. Adventures in associative-commutative unification. *JSC*, 8:393–416, 1989.
17. C. Marché and X. Urbain. Modular and incremental proofs of AC-termination. *JSC*, 38(1):873–897, 2004.
18. G.E. Peterson and M.E. Stickel. Complete sets of reductions for some equational theories. *JACM*, 28(2):233–264, 1981.
19. 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.
20. 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.
21. S. Winkler, H. Sato, A. Middeldorp, and M. Kurihara. Optimizing mkbTT (system description). In *Proc. 21st RTA*, volume 6 of *LIPIcs*, pages 373–384, 2010.