Termination Modulo Combinations of Equational Theories

Francisco Durán, LCC U. Málaga, Spain
Salvador Lucas, DSIC, U. Politécnica de Valencia, Spain
José Meseguer, CS Dept, U. of Illinois at Urbana-Champaign, USA

Motivation: Termination of programs

Reasoning about *termination of programs* in modern (rule-based) equational languages requires support for advanced features such as:

1. Sorts, subsorts, and operator overloading
2. Memberships
3. Conditions, which may introduce extra variables
4. Context-sensitivity, which permits the introduction of annotations to specify the arguments which can be evaluated in each function call
5. Fixed evaluation strategies (e.g., leftmost-innermost or leftmost-outermost)
6. Programmable evaluation strategies
7. *Rewriting modulo axioms like associativity (A), commutativity (C), identity (I), AC, ACI,...*

Motivation: Termination of programs

```
fmod LIST&SET is
  sorts Bool Nat List Set .
  subsorts Nat < List Set .
    ops _and_ _or_ : Bool Bool -> Bool [assoc comm] .
    op 0 : -> Nat .   op s_ : Nat -> Nat .
    op _;_ : List List -> List [assoc] .
    op null : -> Set .
    op __ : Set Set -> Set [assoc comm id: null] .
    op _in_ : Nat Set -> Bool .
    op _==_ : List List -> Bool [comm] .
    op list2set : List -> Set .
  var  B : Bool .             vars N M : Nat .
  vars L L’ : List .          var  S : Set .
    eq N N = N .
    eq true and B = B .         eq false and B = false .
    eq true or B = true .       eq false or B = B .
    eq 0 == s N = false .       eq s N == s M = N == M .
    eq N ; L == M = false .     eq N ; L == M ; L’ = (N == M) and L == L’ .
    eq L == L = true .
    eq list2set(N) = N .        eq list2set(N ; L) = N list2set(L) .
    eq N in null = false .      eq N in M S = (N == M) or N in S .
endfm
```

Motivation: Termination of programs

In this example:

1. The boolean operators `_and_` and `_or_` are *associative* and *commutative*
2. The list concatenation `_;_` is *associative* (but *not commutative*!)
3. The set union operator `_ _` is *associative*, *commutative*, and has a *unit element* `null` (the *empty set*)
4. The equality of lists `_==_` is *commutative* (but *not associative*!)

Unfortunately:

1. No *termination tool* (e.g., AProVE, MU-TERM, TTT2, ...) is able to handle this combination of axioms for the symbols
2. Existing *theoretical frameworks* (e.g., [GK01]) *fail* to provide a basis for proving termination of this example

Termination modulo equational theories

Definition (Termination of a TRS)

A TRS $\mathcal{R}$ is *terminating* if there is *no infinite rewrite sequence* $t_1 \to_{\mathcal{R}} t_2 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t_n \to_{\mathcal{R}} \cdots$

In this paper we are interested in proving *termination of rewriting modulo an equational theory*.

Definition (Termination modulo an equational theory)

A TRS $\mathcal{R}$ is *terminating* modulo a *set of equations* E if there is *no infinite sequence* $s_1 =_E t_1 \to_{\mathcal{R}} s_2 =_E t_2 \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} s_n =_E t_n \to_{\mathcal{R}} \cdots$

Our contribution:

1. A semantics-preserving *transformation* of *rewrite theories* (Σ, E, R) which *reduces* the equational component E (by adding rules to $\mathcal{R}$)
2. Termination preserving transformations for *removing* the associative and commutative axioms from specific equational theories E

Termination modulo combinations of equational theories

Summary:

1. Rewrite theories and rewriting modulo
2. Variants
3. Semantics-preserving transformation
4. Transformations for identity, associative, and commutative equational components
5. Implementation and use
6. Related work
7. Conclusions

Rewrite theories and rewriting modulo

Definition (Rewrite theory)

A *rewrite theory* is a triple $\mathcal{R} = (\Sigma, E, R)$ with Σ a (preregular) order-sorted signature such that each connected component has a top sort, E a set of (linear) Σ-equations, and R a set of Σ-rules.

Definition (Rewriting modulo)

Given a rewrite theory $\mathcal{R}$ as above, $t \rightarrow_{R/E} t'$ iff there exist u, v such that $t =_E u$ and $u \rightarrow_R v$ and $v =_E t'$.

This leads to an *undecidable* relation. The next definition provides a *decidable* alternative (provided that an *E-matching algorithm* is available).

Definition (Rewriting modulo II)

For any terms u, v with sorts in the same connected component, the relation $u \rightarrow_{R,E} v$ holds if there is a position p in u, a rule $l \rightarrow r$ in R, and a substitution σ such that $u|_p =_E l\sigma$ and $v = u[r\sigma]_p$ [PS81].

Rewrite theories and rewriting modulo

Of course, $\to_{R,E} \subseteq \to_{R/E}$, but *can any $\to_{R/E}$-step be simulated by a $\to_{R,E}$-step?* This is equivalent to the following *E-coherence* property:

$$\begin{tikzcd}
u \arrow[r, "R/E"'] \arrow[d, equal, "E"'] & v \arrow[d, dotted, dash, "E"] \\
u' \arrow[r, dotted, "{R,E}"'] & v'
\end{tikzcd}$$

We say that $\mathcal{R} = (\Sigma, E, R)$ is *E-confluent*, resp. *E-terminating*, if $\to_{R/E}$ is confluent, resp. terminating. If $\mathcal{R}$ is *E-coherent*, then *E-termination is equivalent to the termination of the $\to_{R,E}$ relation*.

We further require:

1. *E-preregularity*, i.e., $\{s \in S \mid \exists w' \in [w]_E \text{ s.t. } w' \in \mathcal{T}(\Sigma, \mathcal{X})_s\}$ has a least upper bound, denoted $ls[w]_E$ which can be *effectively computed*.
2. *E-sort-decreasing* of $\mathcal{R}$, i.e., for each rewrite rule $l \to r$, and for each specialization substitution ν we have $ls[r\nu]_E \leq ls[l\nu]_E$.

Variants and variant-based transformation

Under the conditions above, we can view $\mathcal{E} = (\Sigma, B, \Delta)$ as an *order-sorted equational theory* $(\Sigma, \widetilde{\Delta} \cup B)$, where $\widetilde{\Delta} = \{l = r \mid l \to r \in \Delta\}$.

We can then use the B-confluence, B-termination, B-preregularity, and B-sort-decreasingness of $\mathcal{E}$ to make the $\widetilde{\Delta} \cup B$-equality relation *decidable by $\to_{\Delta,B}$-rewriting*.

Remark

The problem of proving termination for a rewrite theory $\mathcal{R} = (\Sigma, E, R)$ is simplified by decomposing E into a union $E = \widetilde{\Delta} \cup B$ such that the axioms B are simpler and the rewrite theory $\mathcal{E}_E = (\Sigma, B, \Delta)$ is B-confluent, B-terminating, B-preregular, B-sort-decreasing, and B-coherent. Then, a new (simpler, yet equivalent) theory $\widehat{\mathcal{R}} = (\Sigma, B, \Delta \cup \widehat{R})$ is obtained.

Variants and variant-based transformation

Definition (Variant [CD05,EMS08])

Let $\mathcal{E}_E = (\Sigma, B, \Delta)$ be a rewrite theory as above. A *Δ, B-variant* of a Σ-term t is a pair $(t\theta{\downarrow}_{\Delta,B}, \theta)$ where $t\theta{\downarrow}_{\Delta,B}$ (or just $t\theta{\downarrow}$) denotes a *canonical form* for t, i.e., a term w such that $t \rightarrow^*_{\Delta,B} w$ and w cannot be further rewritten with $\rightarrow_{\Delta,B}$. Note that $t\theta{\downarrow}$ is unique up to B-equality.

We denote by $[\![t]\!]^*_{\Delta,B}$ the set of Δ, B-variants of t. It is ordered by $(u, \theta) \sqsubseteq_{\Delta,B} (v, \sigma)$, that holds if there is ρ such that $u =_B v\rho$, and $\theta{\downarrow} =_B \sigma\rho$ (that is, for each variable $x \in \mathcal{D}om(\theta)$ we have $x\theta{\downarrow} =_B x\sigma\rho$).

Let $[\![t]\!]_{\Delta,B} \subseteq [\![t]\!]^*_{\Delta,B}$ be a *complete set of maximal elements* in the preordered set $([\![t]\!]^*_{\Delta,B}, \sqsubseteq_{\Delta,B})$.

Definition (Finite variant property [CD05,EMS08])

We say that $\mathcal{E}_E$ has the *finite variant property* if for any Σ-term t we can find a *finite complete set* of most general variants $[\![t]\!]_{\Delta,B}$.

A semantics-preserving transformation

Definition ($\mathcal{R} \mapsto \widehat{\mathcal{R}}$ transformation)

Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory where Σ is E-preregular, R is E-coherent, and such that E can be decomposed as a B-confluent, B-terminating, B-preregular, B-sort-decreasing and B-coherent rewrite theory $\mathcal{E}_E = (\Sigma, B, \Delta)$.

We let $\widehat{\mathcal{R}} = (\Sigma, B, \Delta \cup \widehat{R})$, where $\widehat{R}$ is the *B-coherence completion* of the set of rules $\{\hat{\ell} \to r\alpha \mid \ell \to r \in R,\ \text{and}\ (\hat{\ell}, \alpha) \in [\![\ell]\!]_{\Delta,B}\}$.

This *semantics-preserving* transformation preserves *confluence* and *termination* in the following sense:

Theorem

1. *$\mathcal{R}$ is E-terminating iff $\widehat{\mathcal{R}}$ is B-terminating.*
2. *$\mathcal{R}$ is E-confluent iff $\widehat{\mathcal{R}}$ is B-confluent.*

Transformations for specific equational components

Existing tools for automatically proving termination are able to deal with *rewrite systems* with *AC-symbols*. *No specific techniques* are available for dealing with A or C symbols (only).

Consider a rewrite theory $\mathcal{R} = (\Sigma, E, R)$ with $E = \bigcup_{f:[s_1]\cdots[s_n]\to[s]\in\Sigma} E_f$, where if $n \neq 2$, then $E_f = \varnothing$, and if $n = 2$, then $E_f \subseteq \{A_f, C_f, LU_f, RU_f\}$, where:

- A_f is the axiom $f(f(x,y),z) = f(x,f(y,z))$,
- C_f is the axiom $f(x,y) = f(y,x)$,
- LU_f is the axiom $f(e,x) = x$, for e a given ground term of sort $[s_1]$, and
- RU_f is the axiom $f(x,e') = x$, for e' a given ground term of sort $[s_2]$,

and where the variables x, y, z are all of the appropriate top sorts.

Our goal:

Transform E into a *'pure' AC theory* by (re)moving the LU and RU components and the (pure) A and C components.

Removing the LU and RU equational components

From the *equational component* E of $\mathcal{R} = (\Sigma, E, R)$ we obtain a *rewrite theory* $\mathcal{E} = (\Sigma, B, \widetilde{U})$, with:

$$\begin{array}{rclcrcl} B & = & \bigcup_{f:[s_1]\cdots[s_n]\to[s]\in\Sigma} B_f & \text{for} & B_f & = & E_f \cap \{A_f, C_f\} \\ U & = & \bigcup_{f:[s_1]\cdots[s_n]\to[s]\in\Sigma} U_f & \text{for} & U_f & = & E_f \cap \{LU_f, RU_f\} \end{array}$$

with LU_f and RU_f understood as rewrite rules $f(e, x) \to x$, and $f(x, e) \to x$, and where $\widetilde{U}$ is the *B-coherence completion* of U:

Actually, $\widetilde{U} = \bigcup_{f:[s_1]\cdots[s_n]\to[s]\in\Sigma} \widetilde{U}_f$, where:

1. If $A_f \notin B_f$, or $A_f, C_f \in B_f$, then $\widetilde{U}_f = U_f$.
2. Otherwise, if $A_f \in B_f$, but $C_f \notin B_f$, then,
    1. if $LU_f \in U_f$, then we add the rule $f(x, f(e, y)) \to f(x, y)$ and
    2. if $RU_f \in U_f$, then we add the rule $f(f(x, e'), y) \to f(x, y)$.

Theorem

If $\mathcal{E} = (\Sigma, B, \widetilde{U})$ has a finite set of sorts, is B-preregular and B-sort decreasing, then $\mathcal{E}$ has the finite variant property.

Removing 'pure' A equational components

Given axioms B, where for each f we have $B_f \subseteq \{A_f, C_f\}$, we define a *rewrite theory* (Σ, B°, A), where for each $f \in \mathcal{F}$ we have $B^\circ_f = B_f$ if $B_f \neq \{A_f\}$, and $B^\circ_f = \varnothing$ if $B_f = \{A_f\}$, and where A consists of rules

$$f(f(x,y),z) \;\to\; f(x,f(y,z)) \quad \text{or} \quad f(x,f(y,z)) \;\to\; f(f(x,y),z)$$

(but only one of them) for each $f \in \Sigma$ such that $B_f = \{A_f\}$ (i.e., f is *associative but not commutative*).

Proposition

The theory (Σ, B°, A) is confluent and terminating modulo B°.

We could apply again the transformation $\mathcal{R} \mapsto \widehat{\mathcal{R}}$ (with $E = B$, $B = B^\circ$, and $\Delta = A$) to obtain from a theory $\mathcal{R} = (\Sigma, B, R)$ a semantically equivalent theory $\mathcal{R}_A = (\Sigma, B^\circ, R_A \cup A)$, where the rules R_A are the *A, B°-variants* of the rules $\mathcal{R}$.

Remark

The finite variant property for (Σ, B°, A) must be checked in each case!

Removing ‘pure’ C equational components

Given axioms $B = \bigcup_f B_f$, where for each f we have $B_f \subseteq \{A_f, C_f\}$, we develop a theory transformation $\mathcal{R} = (\Sigma, B, R) \mapsto (\Sigma_C, B_C, R_C) = \mathcal{R}_C$. The rules R must be *B-coherent* and such that all the variables in their left-hand sides are *C-linear* (i.e., without C-nonlinear variables).

Definition (C-nonlinear variable)

Given a rewrite rule $l \to r$ in R, we call a variable $x \in \mathcal{V}ar(l)$ of sort s *C-nonlinear* if (1) it is nonlinear in l, and (2) there exists a Σ-term t with $ls[t]_B \leq s$ with a position p such that $t|_p = f(u, v)$, with $B_f = \{C_f\}$.

The transformation $\mathcal{R} \mapsto \mathcal{R}_C$ is defined with $\mathcal{R}_C = (\Sigma, B_C, R_C)$, where:

1. For each $f \in \Sigma$, if $B_f \neq \{C_f\}$, then $B_{C_f} = B_f$, and if $B_f = \{C_f\}$, then $B_{C_f} = \varnothing$. We also require that Σ_C is *B_C-preregular*.
2. $\mathcal{R}_C$ contains the rules $\widetilde{l'} \to r$ for each $\widetilde{l'} \in [l']_{\widehat{C}}$ where $\widehat{C} = \bigcup_f \widehat{C}_f$, and $\widehat{C}_f = \{C_f\}$ if $B_f = \{C_f\}$, and $\widehat{C}_f = \varnothing$ otherwise. Note that *$[l']_{\widehat{C}}$ consists of permuting all the subterms of l' of the form $f(u, v)$ with $B_f = \{C_f\}$ in all possible ways*.

Removing ‘pure’ C equational components

Example

The application of the transformation $\mathcal{R} \mapsto \mathcal{R}_C$ to our running example is reduced to the addition of equations resulting from permuting all those subterms with a *commutative-only* operator at their top (_==_ in our case). The equations to be added are therefore the following:

```
eq s N == 0 = false .                eq M == N ; L = false .
```

The main result about this transformation is:

Theorem

$\mathcal{R}$ is terminating modulo B iff $\mathcal{R}_C$ is terminating modulo B_C.

Implementation and use

Tool support

All the transformations presented in this paper are currently part of an alpha version of Full Maude, where several commands are available so that the different transformations and checks can be executed.

Running example

1. The different versions of the running example have been obtained *with these commands*
2. The *Maude Termination Tool* (MTT, [DLM08]) has been used to obtain a version of the specification as an *AC-TRS*
3. The *AC-termination* of the obtained AC-TRS has been proved using *AProVE* [GST06]
4. According to our results, the *termination of the original rewrite theory follows from this proof of AC-termination*

Related work

Proving AC-termination

Methods for proving *AC-termination* have been developed by Ben Cherifa and Lescanne [SCP'87], Jouannaud and Marché [TCS'92], Marché and Urbain [RTA'98], etc. There are *tools* for *automatically proving* AC-termination of TRSs.

Proving equational termination

Giesl and Kapur [RTA'01] have developed methods for proving *termination modulo some generic class E* of equational axioms (satisfying some restrictions). We are not aware of *any implementation of them*.

Modular proofs of termination

Current research on *modularity of termination or rewriting* (including AC-rewriting), see [Ohl02] for a good survey, concerns *the rules* $\mathcal{R}$ only (i.e., given terminating TRSs $\mathcal{R}_1$ and $\mathcal{R}_2$, is $\mathcal{R}_1 \oplus \mathcal{R}_2$ terminating?). We are not aware about any modularity result concerning the *equations* E.

Conclusions

1. We have presented a new variant-based method to prove termination modulo *combinations of sets of equational axioms*.
2. Our method is modular both
    1. *vertically*, in the sense that it can be applied repeatedly to reduce such termination proofs modulo increasingly simpler sets of axioms which in the end can be handled by existing termination methods and tools, and
    2. *horizontally*, since it can naturally handle unions of different sets of axioms for different function symbols.
3. We have *illustrated* its usefulness in the very common case where the axioms E are an arbitrary combination of *associativity*, *commutativity*, *left-* and *right-identity* axioms for various function symbols.

Future work

1. Extend these methods to *conditional rewrite theories*
2. Explore how the requirements on E can be *relaxed* to handle even more general sets of axioms
3. Generalize modular termination methods for unions of term rewriting sytems modulo the *unions of their corresponding axioms*
4. Improve the *implementation*; integration into the MTT tool