ISR: Lecture 6

José Meseguer

Computer Science Department
University of Illinois at Urbana-Champaign

Termination

We need methods to check termination of an equational theory (Σ, E). For unconditional equations E this means proving that the rewriting relation $\longrightarrow_E$ (or, more generally, $\longrightarrow_{E/B}$ for $(\Sigma, E \cup B)$) is well-founded.

The key observation is that, if we exhibit a well-founded ordering $>$ on terms such that

$$(\clubsuit) \quad t \longrightarrow_E t' \quad \Rightarrow \quad t > t',$$

then we have obviously proved termination, since nontermination of $\longrightarrow_E$ would make the order $>$ non-well-founded.

Reduction Orderings

To show (♣) we need to consider an, infinite number of rewrites $t \longrightarrow_E t'$. We would like to reduce this problem to checking (♣) only for the equations in E. We need:

Definition: A well-founded ordering $>$ on $\cup_{s \in S} T_\Sigma(V)$ is called a reduction ordering iff it satisfies the following two conditions:

- **strict Σ-monotonicity**: for each $f \in \Sigma$, whenever $f(t_1, \ldots, t_n), f(t_1, \ldots, t_{i-1}, t'_i, t_{i+1}, \ldots, t_n) \in T_\Sigma(V)$ with $t_i > t'_i$, we have,

$$f(t_1, \ldots, t_n) \; > \; f(t_1, \ldots, t_{i-1}, t'_i, t_{i+1}, \ldots, t_n)$$

- **closure under substitutuion**: if $t > t'$, then, for any substitution $\theta : V \longrightarrow T_\Sigma(V)$ we have, $t\theta > t'\theta$.

Reduction Orderings (II)

Theorem: Let (Σ, E) be an (unconditional) equational theory. Then, E is terminating iff there exists a reduction order $>$ such that for each equation $u = v$ in E we have, $u > v$.

Proof: The $(\Rightarrow)$ part follows from the observation that, if E is terminating, the transitive closure $\xrightarrow{+}_E$ of the relation $\longrightarrow_E$ is a reduction order satisfying this requirement.

To see $(\Leftarrow)$, it is enough to show that a reduction order with the above property satisfies the implication $(\clubsuit)$. Let $t \longrightarrow_E t'$ this means that there is a position π in t, an equation $u = v$ in E, and a substitution θ such that $t = t[\pi \leftarrow \overline{\theta}(u)]$, and $t' = t[\pi \leftarrow \overline{\theta}(v)]$. But by closure under substitution we have, $\overline{\theta}(u) > \overline{\theta}(v)$ and by repeated application of strict Σ-monotonicity we then have, $t > t'$. q.e.d.

Recursive Path Ordering (RPO)

The recursive path ordering (RPO) is based on the idea of giving an ordering on the function symbols in Σ, which is then extended to a reduction ordering on all terms. Since if Σ is finite the number of possible orderings between function symbols in Σ is also finite, checking whether a proof of termination exists this way can be automated.

The intuitive idea that functions that are more complex should be bigger in the ordering (for example: $\_*\_ > \_+\_ > \mathtt{s}$) tends to work quite well, and can yield a reduction ordering containg the equations. Furthermore each symbol f in Σ is given a status $\tau(f)$ equal to either: $\tau(f) = lex(\pi)$ (lexicographic), or $\tau(f) = mult$ (multiset). $\tau(f)$ indicates how the arguments of f should be compared in the order.

RPO (II)

Given a finite signature Σ and an ordering $>$ and a status function τ on its symbols, the recursive path ordering $>_{rpo}$ on $\cup_{s \in S} T_{\Sigma}(V)$ is defined recursively as follows. $u >_{rpo} t$ iff:

$u = f(u_1, \ldots, u_n)$, and either:

1. $u_i \geq_{rpo} t$ for some $1 \leq i \leq n$, or
2. t $= g(t_1, \ldots, t_m)$, $u >_{rpo} t_j$ for all $1 \leq j \leq m$, and either:
   - $f > g$, or
   - $f = g$ and $\langle u_1, \ldots, u_n \rangle >_{rpo}^{\tau(f)} \langle t_1, \ldots, t_n \rangle$

where the extension of $>_{rpo}$ to an order $>_{rpo}^{\tau(f)}$ on lists of terms is explained below.

RPO (III)

The extension of $>_{rpo}$ to an order $>_{rpo}^{\tau(f)}$ on lists of terms is defined as follows:

- If f has n arguments and $\tau(f) = lex(\pi)$ with π a permutation on n elements, then $\langle u_1, \ldots, u_n \rangle >_{rpo}^{\tau(f)} \langle t_1, \ldots, t_n \rangle$ iff there exists i, $1 \leq i \leq n$ such that for $j < i$ $u_{\pi(j)} = t_{\pi(j)}$, and $u_{\pi(i)} > t_{\pi(i)}$.

- if $\tau(f) = mult$, then $\langle u_1, \ldots, u_n \rangle >_{rpo}^{\tau(f)} \langle t_1, \ldots, t_n \rangle$ iff we have $\{u_1, \ldots, u_n\} >_{rpo}^{mult} \{t_1, \ldots, t_n\}$

where, given any order $>$ on a set A, it extension to an order $>^{mult}$ on the set $Mult(A)$ of multisets on A is the transitive closure of the relation $>_{elt}^{mult}$ defined by $M \cup a >_{elt}^{mult} M \cup S$ iff $(\forall x \in S)\ a > x$, where S can be $\emptyset$.

RPO (IV)

It can be shown (for a detailed proof see the Terese book cited later) that for a finite signature Σ RPO is a reduction order. We can therefore use it to prove termination.

Consider for example the usual equations for natural number addition: $n + 0 = n$ and $n + s(m) = s(n + m)$. We can prove that they are terminating by using the RPO associated to the ordering $+ > s > 0$ with $\tau(f) = lex(id)$ for each symbol f. Indeed, it is then trivial to check that $n + 0 >_{rpo} n$ and $n + s(m) >_{rpo} s(n + m)$.

Termination Modulo Axioms

To prove that rewriting modulo axioms B are terminating, we need a reduction order that is compatible with the axioms B. That is, if $u > t$, $u =_B u'$ and $t =_B t'$, then we must always have $u' > t'$. This means that $>$ defines also an order on the set, $\cup_{s \in S} T_{\Sigma/B}(X)$. For example, RPO is compatible with commutativity axioms if we specify $\tau(f) = mult$ for each commutative symbol f.

To make RPO compatible with associative and or commutative symbols it has been generalized to the $A \vee C.RPO$ order, where some symbols can be associative and/or commutative.

Proving Termination with $A \vee C.RPO$

The Maude Termination Assistant (MTA) is a simple tool that can prove $A \vee C.RPO$-termination modulo any $A \vee C \vee U$ axioms (with U identity axioms) by first automatically transforming the specification, making axioms U into rules.

To prove a functional module or theory `foo.maude` $A \vee C.RPO$-terminating we specify an order on the module's operators using Maude's `metadata` attribute to:

- Give a rank number to each symbol $f \in \Sigma$ so that $f > g$ iff $rank(f) > rank(g)$.
- If $f \in \Sigma$ satisfies no axioms, specify a lexicographic priority on its arguments other than the default one.

Let us see an example:

Proving Termination with $A \vee C.RPO$ (II)

```
fmod LIST+MSET is
  sorts Element List MSet .
  subsorts Element < List .  subsorts Element < MSet .
  op a : -> Element [ctor metadata "1"] .
  op b : -> Element [ctor metadata "2"] .
  op c : -> Element [ctor metadata "3"] .
  op nil : -> List [ctor metadata "4"] .
  op _;_  : List List -> List [metadata "5 lex(2 1)"] .
  op _;_  : List Element -> List [ctor metadata "5 lex(2 1)"] .
  op _,_ : MSet MSet -> MSet [ctor assoc comm metadata "4"] .
  op null : -> MSet [ctor metadata "3"] .
  op l2m : List -> MSet [ctor metadata "5"] .
  vars L P Q : List .  var M : MSet .  var E : Element .
  eq L ; (P ; Q) = (L ; P) ; Q .        eq L ; nil = L .
  eq nil ; L = L .                      eq M , null = M .
  eq l2m(nil) = null .                  eq l2m(E) = E .
  eq l2m(L ; E) = l2m(L) , E .
endfm
```

Proving Termination with $A \vee C.RPO$ (III)

We then do the following:

1. load in Maude our functional module or theory `FOO` with the explained `metadata` annotations and excluding importation of any built-in modules by first giving the `set include BOOL off .` command.

2. load the file `ui.maude` containing the MTA tool implementation and providing a simple user interface as a Full Maude extension; and

3. type the command `(check-AvCrpo FOO .)`

For example, the command `(check-AvCrpo LIST+MSET .)` is answered with output:

```
Module is terminating by AvC-RPO order.
```

Polynomial Orderings

Another general method of defining suitable reduction orderings is based on polynomial orderings. In its simplest form we can just use polynomials on several variables whose coefficients are natural numbers. For example,

$$p = 7x_1^3x_2 + 4x_2^2x_3 + 6x_3^2 + 5x_1 + 2x_2 + 11$$

is one such polynomial. Note that a polynomial p whose biggest indexed variable is n (in the above example $n = 3$) defines a function $p_{\mathbf{N}_{\geq k}} : \mathbf{N}_{\geq k}^n \longrightarrow \mathbf{N}_{\geq k}$ (where $k \geq 3$ and $\mathbf{N}_{\geq k} = \{n \in \mathbf{N} \mid n \geq k\}$), just by evaluating the polynomial on a given tuple of numbers greater or equal to k. For p the polynomial above we have for example, $p_{\mathbf{N}_{\geq k}}(3, 3, 3) = 383$.

Polynomial Orderings (II)

Note also that we can order the set $[\mathbf{N}^n_{\geq k} \to \mathbf{N}_{\geq k}]$ of functions from $\mathbf{N}^n_{\geq k}$ to $\mathbf{N}_{\geq k}$ by defining $f > g$ iff for each $(a_1, \ldots a_n) \in \mathbf{N}^n_{\geq k}$ $f(a_1, \ldots a_n) > g(a_1, \ldots a_n)$. Notice that this order is well-founded, since if we have an infinite descending chain of functions

$$f_1 > f_2 > \ldots f_n > \ldots$$

by choosing any $(a_1, \ldots a_n) \in \mathbf{N}^n_{\geq k}$ we would get a descending chain of positive numbers

$$f_1(a_1, \ldots a_n) > f_2(a_1, \ldots a_n) > \ldots f_n(a_1, \ldots a_n) > \ldots$$

which is impossible.

Polynomial Orderings (III)

The method of polynomial orderings then consists in assigning to each function symbol $f : s_1 \ldots s_n \longrightarrow s$ in Σ a polynomial p_f involving exactly the variables $x_1, \ldots x_n$ (all of them, and only them must appear in p_f). If f is subsort overloaded, we assign the same p_f to all such overloadings. Also, to each constant symbol b we likewise associate a positive number $p_b \in \mathbf{N}_{\geq k}$.

Suppose, to simplify notation, that in our set E of equations we have used exactly m different variables, denoted $x_1, \ldots x_m$, each declared with its corresponding sort. Let us denote $X = \{x_1, \ldots x_m\}$. Then our assignment of a polynomial to each function symbol and a number to each constant extends to a function

Polynomial Orderings (IV)

$$p_{\_} : T_{\Sigma^u(X)} \longrightarrow \mathbf{N}[X]$$

where Σ^u is the unsorted version of Σ, $\mathbf{N}[X]$ denotes the polynomials with natural number coefficients in the variables X, and where $p_{\_}$ is defined in the obvious, homomorphic way:

- $p_b = p_b$
- $p_{x_i} = x_i$
- $p_{f(t_1,\ldots,t_n)} = p_f\{x_1 \mapsto p_{t_1}, \ldots, x_n \mapsto p_{t_n}\}$

Polynomial Orderings (V)

Note that the the polynomial interpretation p induces a well-founded ordering $>_p$ on the terms of $T_{\Sigma(X)}$ as follows:

$$t >_p t' \quad \Leftrightarrow \quad p_{t_{\mathbf{N}_{\geq k}}} > p_{t'_{\mathbf{N}_{\geq k}}}$$

where if $X = \{x_1, \ldots x_k\}$, we interpret $p_{t_{\mathbf{N}_{\geq k}}}$ and $p_{t'_{\mathbf{N}_{\geq k}}}$ as functions in $[\mathbf{N}^m_{\geq k} \to \mathbf{N}_{\geq k}]$. The relation $>_p$ is clearly an irreflexive and transitive relation on terms in $T_{\Sigma(X)} \subseteq T_{\Sigma^u(X)}$, therefore a strict ordering, and is clearly well-founded, because otherwise we would have an infinite descending chain of polynomial functions in $[\mathbf{N}^m_{\geq k} \to \mathbf{N}_{\geq k}]$, which is impossible.

Polynomial Orderings (VI)

We now need to check that this ordering is furthermore: (i) strictly Σ-monotonic, and (ii) closed under substitution. Condition (i) follows easily from the fact that for each function symbol $f : s_1 \ldots s_n \longrightarrow s$ in Σ p_f involves exactly the variables $x_1, \ldots x_n$ (p_f does not drop any variables and all coefficients are non-zero). Therefore, $p_{f_{\mathbf{N}_{\geq k}}}$, viewed as a function of n arguments, is strictly monotonic in each of its arguments. Condition (ii) follows from the following general property of the $p\_$ function, which is left as an excercise:

$$p_{t\{x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\}} = p_t\{x_1 \mapsto p_{u_1}, \ldots, x_n \mapsto p_{u_n}\}.$$

This then easily yields that if $t >_p t'$ then $t\{x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\} >_p t'\{x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\}$, as desired.

Polynomial Orderings (VII)

Therefore, polynomial interpretations of this kind define reduction orderings and can be used to prove termination. Consider for example the single equation $f(g(x)) = g(f(x))$ in an unsorted signature having also a constant a. Is this equation terminating? We can prove that it is so by the following polynomial interpretation:

- $p_f = {x_1}^3$
- $p_g = 2x_1$
- $p_a = 1$

since we have the following strict inequality of functions: $((2x)^3)_{\mathbf{N}_{\geq k}} > (2(x^3))_{\mathbf{N}_{\geq k}}$, showing that $f(g(x)) >_p g(f(x))$.

Polynomial Termination Modulo Axioms

Some polynomial interpretations are compatible with certain axioms. For example, a symmetric polynomial such that $p(x, y) = p(y, x)$ is compatible with commutativity and can therefore be used to interpret a commutative symbol. For example, $2x + 2y$ is symmetric. Similarly, a polynomial $p(x, y)$ which is symmetric ($p(x, y) = p(y, x)$) and furthermore satisfies the associativity equation $p(x, p(y, z)) = p(p(x, y), z)$ can be used to interpret an associative or associative-commutative symbol. As shown by Bencheriffa and Lescanne the polynomials satisfying AC axioms (and therefore A ones) have a simple characterization: they must be of the form $axy + b(x + y) + c$ with $ac + b - b^2 = 0$.

Proving Polynomial Termination with MTA

The MTA tool can prove polynomial termination using linear polynomials of the form $a_1x_1 + \ldots + a_nx_n + a_{n+1}$. A constant c should have $a_c \geq 2$.

Such polynomials are specified by the user for $f \in \Sigma$ with n arguments by annotating f with the `metadata` declaration:

```
metadata "a1 ... an+1"
```

If $f \in \Sigma$ is binary and commutative, we must have $a_1 = a_2$. If $f \in \Sigma$ is binary and associative or associative-commutative, we must have $a_1 = a_2 = 1$.

Let us see an example:

Proving Polynomial Termination with MTA (II)

```
set include BOOL off .

fmod NATU is
  sort Natu .
  op 0   :           -> Natu [ctor metadata "2"    ] .
  op s   : Natu      -> Natu [ctor metadata "1 1"  ] .
  op _+_ : Natu Natu -> Natu [comm metadata "2 2 1"] .
  var X Y : Natu .
  eq X + 0    = X .
  eq X + s(Y) = s(X + Y) .
endfm
```

Proving Polynomial Termination with MTA (III)

The interaction with MTA is similar to that for the $A \vee C.RPO$ case. We:

1. load in Maude our functional module or theory `FOO` with the explained `metadata` annotations and excluding importation of any built-in modules by first giving the `set include BOOL off .` command.

2. load the file `ui.maude` containing the MTA tool implementation and providing a simple user interface as a Full Maude extension; and

3. type the command `(check-poly FOO .)`

The MTT Tool

On the plus side, MTA gives the user a lot of freedom to choose an order. On the minus side: (i) it requires user interaction; and (ii) it does not support more powerful termination methods such as dependency pairs.

For fully automatic termination proofs, the Maude formal environment offers the Maude Termination Tool (MTT).

MTT uses theory transformations and invokes as backend the best currently available automatic termination tools such as **MU-TERM** and **AProVE**.

A good strategy is to first try an automated termination proof with MTT and if this fails use MTA.

Termination is Undecidable

All the termination tools try to prove that a set of oriented equations E, is terminating modulo axioms B by applying different proof methods; for example by trying to see if particular orderings can be used to prove the equations terminating.

But these termination proof methods are not decision procedures: in general termination of a set of equations is undecidable . However, termination is decidable for finite sets of unconditional equations E such that both the lefthand and the righthand sides are ground terms, or even if just the righthand sides are ground terms (see Chapter 5 in Baader and Nipkow, "Term Rewriting and All That", Cambridge U.P.).

Where to Go from Here

Besides RPO and polynomials there are various other orderings and a general "dependency pairs" method that can be used to prove termination. Good sources include:

TeReSe, "Term Rewriting Systems," Cambridge U. P., 2003.

Baader and Nipkow, "Term Rewriting and All That", Cambridge U.P., 1998.

N. Dershowitz and J.-P. Jouannaud, "Rewrite Systems," in J. van Leeuwen, ed., "Handbook of Theoretical Computer Science," Elsevier, 1990.

E. Ohlebusch, "Advanced Topics in Term Rewriting Systems," Springer Verlag, 2002.