Formal Methods: Lecture 8

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, $\longrightarrow_{E/A}$) is Noetherian (see **Ex**.6.1).

The key observation is that, if we exhibit a Noetherian 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-Noetherian.

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 Noetherian 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, $\overline{\theta}(t) > \overline{\theta}(t')$.

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 $\stackrel{+}{\longrightarrow}_E$ 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.

Polynomial Orderings

We are still left with the task of defining suitable reduction orderings. There are several general ways of defining such orderings. One general method 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 $p_{\mathbf{N}_+} : \mathbf{N}_+^n \longrightarrow \mathbf{N}_+$, just by evaluating the polynomial on a given tuple of positive numbers. For p the polynomial above we have for example, $p_{\mathbf{N}_+}(1, 1, 1) = 35$.

Polynomial Orderings (II)

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

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

by choosing any $(a_1, \ldots a_n) \in \mathbf{N}_+^n$ 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 $f : s_1 \dots s_n \longrightarrow s$ in Σ a polynomial p_f involving exactly the variables $x_1, \dots x_n$ (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 $p_b \in \mathbf{N}_+$.

Suppose, to simplify notation, that in our set E of equations we have used exactly k different variables, denoted $x_1, \dots x_k$, each declared with a different sort. Let us denote $X = \{x_1, \dots x_k\}$. Then our assignment of a polynomial to each function symbol and a number to each constant extends to an S-sorted family of functions

Polynomial Orderings (IV)

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

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

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

Polynomial Orderings (V)

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

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

where if $X = \{x_1, \ldots x_k\}$, we interpret $p_{t_{\mathbf{N}_+}}$ and $p_{t'_{\mathbf{N}_+}}$ as functions in $\mathbf{N}_+^{\mathbf{N}_+^k}$. The relation $>_p$ is clearly an irreflexive and transitive relation on terms in $T_{\Sigma(X)}$, therefore a strict ordering, and is clearly noetherian, because otherwise we would have an infinite descending chain of polynomial functions in $\mathbf{N}_+^{\mathbf{N}_+^k}$, 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 $x_1, \ldots x_n$ and has all its coefficients in $\mathbf{N}_+$. Therefore, $p_{f_{\mathbf{N}_+}}$, 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:

$$p_{t(x_1/u_1,\ldots,x_n/u_n)} = p_t(x_1/p_{u_1}, \ldots, x_n/p_{u_n}).$$

This then easily yields that if $t >_p t'$ then $t(x_1/u_1, \ldots, x_n/u_n) >_p t'(x_1/u_1, \ldots, x_n/u_n)$, 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: $((2x)^3)_{\mathbf{N}_+} > (2(x^3))_{\mathbf{N}_+}$, showing that $f(g(x)) >_p g(f(x))$.

The Lexicographic Path Ordering

A second class of reduction orderings are based on the idea of defining such orderings from some minimal information such as 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. One such method is the lexicographic path ordering.

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 contaning the equations. But since there is a finite number of possible orderings, a tool can try them all.

The Lexicographic Path Ordering (II)

Definition: Given a finite signature Σ an an ordering $>$ of its symbols, then the lexicographic path ordering $>_{lpo}$ on $\cup_{s \in S} T_\Sigma(V)$ is defined recursively as follows:

- if $x \in vars(t)$ and $x \neq t$, then $t >_{lpo} x$.
- $f(t_1, \ldots, t_n) >_{lpo} g(t'_1, \ldots, t'_m)$ if either:
  - for some i, $t_i >_{lpo} g(t'_1, \ldots, t'_m)$, or
  - $f > g$ and $f(t_1, \ldots, t_n) >_{lpo} t'_j$, for all j, or
  - $f = g$, $f(t_1, \ldots, t_n) >_{lpo} t'_j$, for all j, and there exist an i such that, $t_j = t'_j$, $1 \leq j < i$, and $t_i >_{lpo} t'_i$.

The Lexicographic Path Ordering (III)

It can be shown (for a detailed proof see the book by Baader and Nipkow cited later) that for a finite signature Σ LPO 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 LPO associated to the ordering $+ > s > 0$. Indeed, it is then trivial to check that $n + 0 >_{lpo} n$ and $n + s(m) >_{lpo} s(n + m)$.

A Caveat on Termination Modulo Axioms

Since when rewriting modulo axioms we are rewriting equivalence classes, some orderings may not be appropriate, because they may give different results on different representatives of the equivalence class. For example, the lexicograhic path order may lead to such conflicts, because of the strict left-to-right order used on the list of subterms, violated by commutativity.

What we need is an order compatible with the axioms, so that it defines also an order on the set, $\cup_{s \in S} T_{\Sigma/A}(X)$. Such orders do exist. For example, an order treating the subterms as a multiset, instead than as a list, may be compatible with a commutativity axiom.

A Caveat on Termination Modulo Axioms (II)

Some polynomial interpretations may also be 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, $2x^3 + 2y^3$ 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-commutative symbol. As shown by Bencheriffa and Lescanne the polynomials satisfying these two conditions have a simple characterization: they must be of the form $axy + b(x + y) + c$ with $ac + b - b^2 = 0$.

Conditional Termination

The issue of termination is more subtle for conditional equational theories, because it is not enough to just require that the relation $\xrightarrow{+}_E$ is a Noetherian order. The problem is that it is easy to give examples of conditional equational specifications (Σ, E) where $\xrightarrow{+}_E$ is Noetherian but any reasonable implementation (including Maude) will loop!

Consider, for example, a theory (Σ, E) with Σ having a single sort and just three constants a, b, and c, and with E consisiting of just the following conditional equation:

$$a = b \text{ if } a = c$$

The relation $\xrightarrow{+}_E$ is the empty set! therefore it is trivially Noetherian. However, Maude will loop when trying to rewrite a.

Conditional Termination (II)

The problem is that, when trying to simplify a the interpreter will try to apply the equation

$$a = b \text{ if } a = c$$

but to do so it will have to first check the condition $a = c$. It will do so by trying to simplify both sides to canonical form and then check syntactic equality of such canonical forms. The constant c is indeed in canonical form. But what about a? Well, we need to see if any equation applies, namely our only conditional equation, but we must first check the condition (the same condition again!) and we are down a silly infinite loop, in which the same condition is checked recursively ad infinitum.

Conditional Termination (III)

What can we then do to prove the actual termination in practice of conditional theories? We need to do three things:

1. give a general notion of conditional termination that captures the actual termination in practice of an implementation;
2. develop adequate methods to prove that a conditional theory terminates in this sense; and
3. use (or develop) adequate tools supporting those methods.

There is a substantial literature on conditional termination, but some of the notions are wrong.

Conditional Termination (V)

For example, so-called "effective termination" is wrong, because an implementation will still loop on some effectively terminating specifications; so their termination is not very "effective" after all. Recent work of Lucas, Marché and Meseguer proposes a notion called operational termination which does indeed capture the actual termination of an interpreter.

To get a feeling for the notion of operational termination we may consider its absence in the simple example we have discussed. The point is that, in trying to simplify a an interpreter will build an infinite proof tree, causing the looping. In our example the infinite proof tree is:

Conditional Termination (VI)

$$\dfrac{\dfrac{\vdots}{a \rightarrow^1 b} \quad b \rightarrow^* c}{\dfrac{a \rightarrow^* c}{a \rightarrow^1 b}}$$

where $\rightarrow^1$ denotes exactly one simplification step, and $\rightarrow^*$ denotes 0, one, or more simplification steps.

Intuitively, operational termination means that there are no infinite proof trees when we build them from left to right. That is, an implementation will build such proof trees incrementally, solving one goal at a time. For example, if the condition is a conjunction, it will solve the first conjunct first, then the second, and so on (complete details in Lucas, Marché and Meseguer).

Conditional Termination (VII)

What methods can we now use to prove operational termination? There are basically two kinds of methods: (i) direct methods; and (ii) transformational methods.

In a direct method, we work directly with the conditional equations as given. A good example of a direct method is the following. Given a conditional equational theory (Σ, E) find a reduction ordering $>$ such that it contains the subterm ordering and for each conditional equation

$$(\forall X)\, t = t' \Leftarrow u_1 = v_1 \wedge \ldots \wedge u_n = v_n$$

in E we have, $t > t'$, and $t > u_i$, $t > v_i$, for $1 \leq i \leq n$. If such an ordering can be found, we call (Σ, E) reductive.

Conditional Termination (VIII)

It is not hard to prove that if (Σ, E) is reductive, then it is operationally terminating: intuitively, we cannot get an infinite proof tree, because the terms in the conditions are getting smaller and they must bottom out. Therefore this is a method to prove operational termination when it can be applied.

One restriction implicit in such a method is that there are no extra variables in the condition. We have assumed this throughout for simplicity, but Maude does allow the use of such extra variables, provided some reasonable executability assumptions are satisfied (see the Maude 2.2 manual).

Conditional Termination (IX)

An alternative proof termination technique that does work in full generality and allows extra variables in conditions is to use a transformational method, in which a Maude functional module (Σ, E) is transformed into an unsorted and unconditional theory $(\Sigma^\bullet, E^\bullet)$ so that if $(\Sigma^\bullet, E^\bullet)$ terminates in the standard sense (that is, if $\xrightarrow{+}_{E^\bullet}$ is Noetherian), then (Σ, E) is operationally terminating.

The above transformation $(\Sigma, E) \mapsto (\Sigma^\bullet, E^\bullet)$ is carried out in two steps: first (Σ, E) is transformed into an unsorted conditional theory (Σ°, E°); and then (Σ°, E°) is transformed into an unsorted unconditional theory $(\Sigma^\bullet, E^\bullet)$. There are several very good tools to prove termination of unsorted unconditional theories. The MTT tool performs these transformations and then invokes those tools as backends.

The MTT Tool

The Maude Termination Tool (MTT) is a tool that can be used to prove the operational termination of Maude functional modules. In general, such modules can be conditional and may be not just order-sorted, but membership equational theories.

They may involve axioms like associativity and commutativity; and they may also have evaluation strategies (see Maude 2.2 manual, Section 4.4.7) indicating what arguments of a function symbol should be evaluated before applying equations for that symbol. For example, in an `if_then_else_fi` the first agument should be evaluated before equations for it are applied; and in a "lazy list cons" `_;_` the first argument is evaluated, but not the second.

The MTT Tool (II)

Features such as sorts, subsorts, memberships, and evaluation strategies may be essential for the termination of a Maude module. That is, ignoring them may result in a nonterminating module.

To preserve these features somehow, while still allowing using standard termination backend tools, the MTT implements the transformations of (Σ, E) first into an unsorted conditional theory (Σ°, E°), and then (Σ°, E°) is transformed into an unsorted unconditional theory $(\Sigma^\bullet, E^\bullet)$.

If the module declares evaluation strategies, they are also transformed; but at the end evaluation strategies can either be used directly by a termination tool like Mu-Term, or a further theory transformation can eliminate such strategies.

The MTT Tool (III)

The course web page indicates where MTT has been installed. By typing: `./MTT` and carriage return the tool's GUI comes up and the user can interact with it. By using the `File` menu one can enter a Maude module into the tool.

Once a Maude module (enclosed in parentheses, and not importing any built-in modules) has been entered, the user can perform the theory transformation $(\Sigma, E) \mapsto (\Sigma^\bullet, E^\bullet)$ in one of three increasingly simpler modes: (1) **Complete**; (2) **No Kinds**; and (3) **No Sorts**. In case (2) kinds are ignored; and in case (3) both kinds and sorts are ignored. There is a tradeoff between simplicity of the transformation and its tightness. Sometimes a simpler transformation works better, and sometimes a more complete one is essentially needed.

The MTT Tool (IV)

The choice of transformation can be made by clicking the appropriate buttons (a screenshot will show this). But one also needs to choose which backend termination tool for unsorted and uncondional specifications will be used. One among the **CiME**, **MU-TERM**, and **AProVE** termination tools can be chosen.

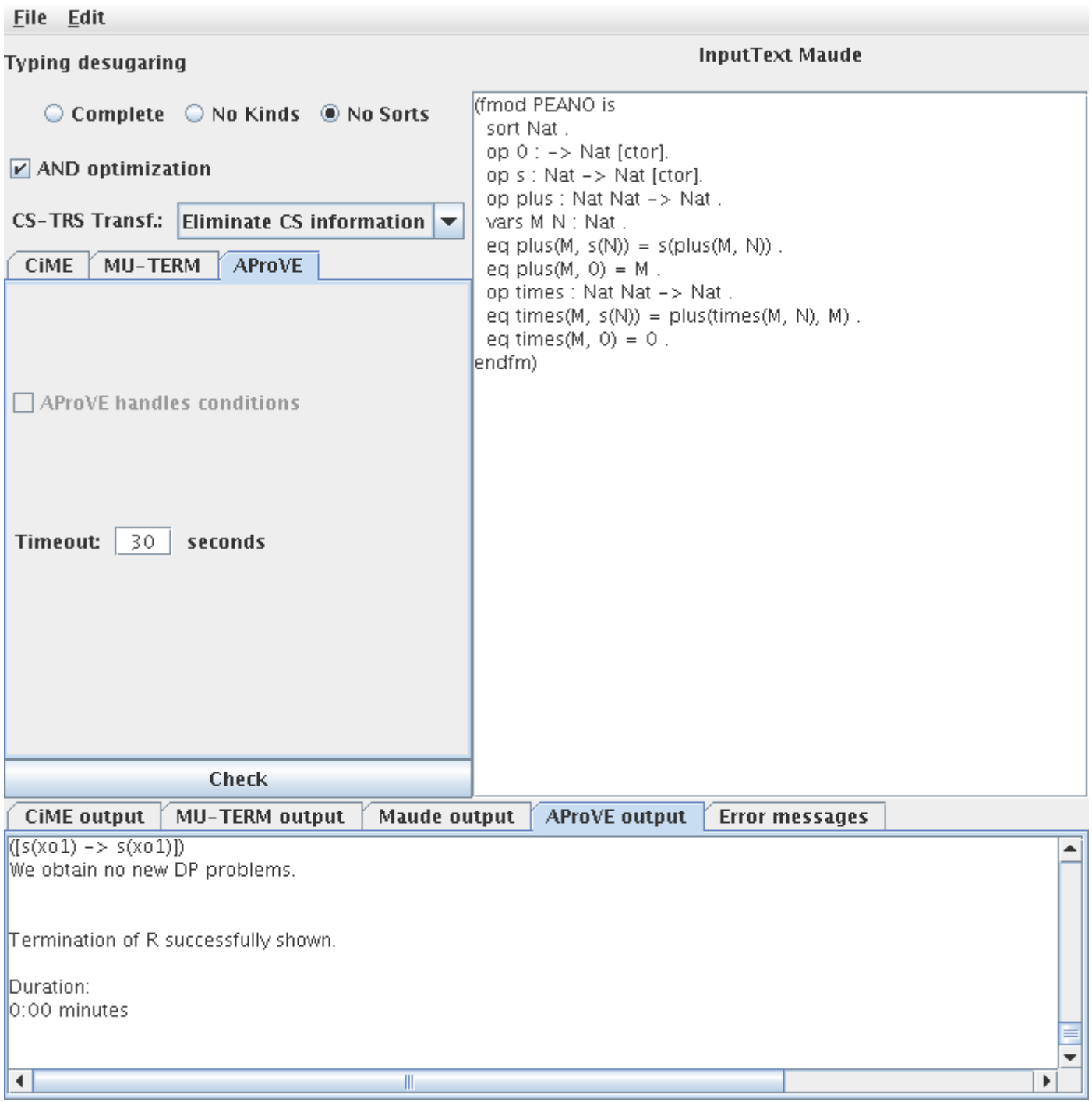
Then one can click on the `Check` bar to check the specification with the chosen tool. Some of these tools offer choices for different settings. So, we can try to prove termination using three different transformation variants, and then with one of three backend tools, sometimes customizing the particular tool choices. This maximizes the chances of obtaining a successful termination proof.

The MTT Tool (V)

What the tool then demonstrates is that the original Maude functional module is operationally terminating. The correctness of such a proof is based on:

- the correctness of the theory transformations (see paper in course web page); and
- the correctness of the chosen tools, that sometimes output a justification of how they proved termination.

A screeshot of a tool interaction is given in the next page.


File Edit
Typing desugaring
InputText Maude
Complete
No Kinds
No Sorts
AND optimization
CS-TRS Transf.:
Eliminate CS information
CiME
MU-TERM
AProVE
AProVE handles conditions
Timeout:
30
seconds
Check
(fmod PEANO is
sort Nat .
op 0 : -> Nat [ctor].
op s : Nat -> Nat [ctor].
op plus : Nat Nat -> Nat .
vars M N : Nat .
eq plus(M, s(N)) = s(plus(M, N)) .
eq plus(M, 0) = M .
op times : Nat Nat -> Nat .
eq times(M, s(N)) = plus(times(M, N), M) .
eq times(M, 0) = 0 .
endfm)
CiME output
MU-TERM output
Maude output
AProVE output
Error messages
([s(x01) -> s(x01)])
We obtain no new DP problems.
Termination of R successfully shown.
Duration:
0:00 minutes

Termination is Undecidable

All the termination tools try to prove that a set of equations E, conditional or unconditional, is terminating 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 (even if they are unconditional) 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 lexicographic path ordering there are many other ordering that can be used to prove termination. Two good sources, with references to the general literature, are:

F. Baader and T. Nipkow, "Term Rewriting and All That," Cambridge Univ. Press, 1998.

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

Furthermore, the Ölveczky lecture notes contain a very good introduction to termination with many examples.

Where to Go from Here (II)

On conditional termination, a very good general reference is

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

and for the work on operational termination and its MTT tool support:

S. Lucas, C. Marché and J. Meseguer, "Operational Termination of Conditional Term Rewriting Systems," *Information Processing Letters* 95, 446–453, 2005.

F. Durán, S. Lucas, C. Marché, J. Meseguer, and X. Urbain, "Proving Operational Termination of Membership Equational Programs," (in the course web page).