Programming Languages | J.J. Horning Editor

Proving Termination with Multiset Orderings

Nachum Dershowitz and Zohar Manna
Stanford University and
The Weizmann Institute of Science

A common tool for proving the termination of programs is the *well-founded set*, a set ordered in such a way as to admit no infinite descending sequences. The basic approach is to find a *termination function* that maps the values of the program variables into some well-founded set, such that the value of the termination function is repeatedly reduced throughout the computation. All too often, the termination functions required are difficult to find and are of a complexity out of proportion to the program under consideration.

***Multisets* (*bags*) over a given well-founded set S are sets that admit multiple occurrences of elements taken from S. The given ordering on S induces an ordering on the finite multisets over S. This *multiset ordering* is shown to be well-founded. The multiset ordering enables the use of relatively simple and intuitive termination functions in otherwise difficult termination proofs. In particular, the multiset ordering is used to prove the termination of *production systems*, programs defined in terms of sets of rewriting rules.**

Key Words and Phrases: program correctness, program termination, program verification, well-founded orderings, well-founded sets, multisets, bags, production systems, term rewriting systems, tree replacement systems, reduction rules

CR Categories: 5.24, 5.7

This research was supported in part by the United States Air Force Office of Scientific Research under Grant AFOSR-76-2909 (sponsored by the Rome Air Development Center, Griffiss Air Force Base, NY), by the National Science Foundation under Grant MCS 76-83655, and by the Advanced Research Projects Agency of the Department of Defense under Contract MDA 903-76-C-0206.

Authors' present addresses: N. Dershowitz, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801; Z. Manna, Computer Science Department, Stanford University, Stanford, CA 94305.

1. Introduction

The use of well-founded sets for proving that programs terminate has been suggested by Floyd [2]. A *well-founded set* $(S, >)$ consists of a set of elements S and an ordering $>$ defined on the elements, such that there can be no infinite descending sequences of elements. The idea is to find a well-founded set and a *termination function* that maps the values of the program variables into that set, such that the value of the termination function is repeatedly decreased throughout the computation. Since, by the nature of the set, that value cannot decrease indefinitely, the program must terminate. The well-founded sets most frequently used for this purpose are the natural numbers under the "greater-than" ordering and n-tuples of natural numbers under the lexicographic ordering.

In this paper, we define and illustrate a class of orderings on multisets. *Multisets*, sometimes called *bags*, are like sets, but allow multiple occurrences of identical elements. For example, {3, 3, 3, 4, 0, 0} is a multiset of natural numbers; it is identical to the multiset {0, 3, 3, 0, 4, 3} but distinct from {3, 4, 0}.

The ordering $>$ on any given well-founded set S can be extended to form a well-founded ordering $\gg$ on the finite multisets over S. In this ordering, $M \gg M'$, for two finite multisets M and M' over S, if M' can be obtained from M by replacing one or more elements in M by any finite number of elements taken from S, each of which is smaller than one of the replaced elements. In particular, a multiset is reduced by replacing an element with zero elements, i.e. by deleting it. Thus if S is the set of natural numbers 0, 1, 2, ... with the $>$ ordering, then under the corresponding *multiset ordering* $\gg$ *over* S, the multiset {3, 3, 4, 0} is greater than each of the three multisets {3, 4}, {3, 2, 2, 1, 1, 1, 4, 0}, and {3, 3, 3, 3, 2, 2}. In the first case, two elements have been removed; in the second case, an occurrence of 3 has been replaced by two occurrences of 2 and three occurrences of 1; and in the third case, the element 4 has been replaced by two occurrences each of 3 and 2, and in addition the element 0 has been removed. The empty multiset { } is clearly smaller than any other multiset.

As an example of the use of a multiset ordering for a proof of termination, consider the following trivial program to empty a shunting yard of all trains:

```
loop until the shunting yard is empty
    select a train
    if the train consists of only a single car
        then remove it from the yard
        else split it into two shorter trains
        fi
    repeat
```

This program is nondeterministic, as it does not indicate which train is to be selected nor how the train is to be split.

Let *trains(yard)* be the number of trains in the yard, and *cars(yard)* be the total number of cars in the yard. For any *train* $\in$ *yard*, let *cars(train)* be the number of cars it contains. We present two proofs of termination.

If we take the set of natural numbers as our well-founded set, then we are led to the selection of the termination function

$$\tau(yard) = 2 \cdot cars(yard) - trains(yard).$$

(See [1].) This solution uses the fact that "splitting" conserves the number of cars in the yard. Splitting a train into two increases *trains(yard)* by 1, thereby decreasing the current value of the termination function $\tau(yard)$ by 1. Removing a one-car train from the yard decreases $2 \cdot cars(yard)$ by 2 and increases $-trains(yard)$ by 1, thereby decreasing $\tau(yard)$ by 1.

If we use multisets of natural numbers as our well-founded set, then the function

$$\tau(yard) = \{cars(train) : train \in yard\}$$

demonstrates the termination of the shunting program. That is, for any configuration of the yard, $\tau(yard)$ denotes the multiset containing the length of each of the trains in the yard. Each iteration of the program loop clearly decreases the value of $\tau(yard)$ under the multiset ordering: removing a train from the yard reduces the multiset by removing one element; splitting a train replaces one element with two smaller ones, corresponding to the two shorter trains.

Programs are sometimes written in the form of a *production system*. The following system of three rewrite rules is an example:

white, red → *red, white*
blue, red → *red, blue*
blue, white → *white, blue.*

This program solves the "Dutch national flag" problem: Assuming that we have a series of marbles, colored *red*, *white*, or *blue* and placed side by side in no particular order, then the above program will rearrange the marbles so that all the *red* marbles are on the left, all *blue* marbles are on the right, and all *white* marbles are in the middle. The first rule, for example, states that if anywhere in the series there is an adjacent pair of marbles, the left one *white* and the right one *red*, then they should be exchanged so that the *red* marble is on the left and the *white* one is on the right.

The three rules may be applied in any order and to any pair of marbles matching a left-hand side of a rule. The program terminates when no rule can be applied. Clearly, if no rule can be applied, the marbles are in the desired order, since nowhere does a *red* marble have anything but a *red* marble to its immediate left (or else one of the first two rules could be applied), and nowhere does a *blue* marble have anything but a *blue* marble to its right (or else one of the last two rules could be applied). The only thing we need to ascertain is that the program will not run indefinitely, never reaching a situation when no rule can be applied; in other words, we must prove that the above production system terminates.

There are several ways of proving the termination of the program. The three given here all use the following ordering on colors:

blue is greater than *white* and *white* is greater than *red.*

It follows from the transitivity of orderings that *blue* is also greater than *red.*

The first method counts the total number of "inversions" of marbles, i.e. the number of pairs of marbles *a* and *b* (not necessarily adjacent), such that *a* appears to the left of *b* and the color of *a* is greater than the color of *b*. For example, if five marbles are arranged *blue, red, white, red, blue,* then there are four inversions: *blue-red, blue-white, blue-red,* and *white-red.* Thus, the well-founded set is the set of natural numbers under their standard $>$ ordering, and the termination function counts the number of inversions by summing, for each marble, the number of marbles with a greater color to its left. Each of the rules, when applied, eliminates one inversion by exchanging the positions of one inverted pair, without generating any additional inversions, thereby decreasing the value of the termination function by one.

For the second method, suppose that there are *n* marbles. The well-founded set we use is the set of *n*-tuples of colors. Such tuples are ordered *lexicographically*: They are reduced if some component is reduced without changing any component to its left. The termination function simply yields the tuple containing the colors of the marbles in order, from left to right. To prove termination, we note that whenever one of the rules is applied to two marbles, only the values of the two corresponding components of the tuple change. By the nature of the lexicographic ordering, we need only consider the change in the left component, and indeed it is reduced in its color: If it was *white*, then now it is *red*, and if it was *blue*, then now it is either *red* or *white*.

The third solution illustrates the use of multiset orderings. Each of the *n* positions in the series is assigned a number, beginning with $n - 1$ at the left and going down to 0 for the rightmost position. We take the multisets of pairs of the form (*position, color*) as the well-founded set. The position-color pairs are ordered lexicographically: We say that a pair is greater than another if it has a higher position number than the other or if it has the same position number but a greater color. For each marble, the termination function yields one pair, giving its position and color. When a rule is applied to the marbles at positions *i* and $i - 1$, it decreases the value of the multiset by decreasing the color of the marble at position *i*. The fact that the color at position $i - 1$ is increased does not matter, since any pair with position *i* is lexicographically greater than any pair with position $i - 1$, regardless of the colors.

These two examples have demonstrated how multiset orderings may be used in termination proofs. These

proofs, however, did not have a clear advantage over the alternative proofs using the more common "greater-than" relation on the natural numbers and lexicographic ordering on n-tuples. In practice, the use of these conventional orderings may lead to complex termination functions that are difficult to discover. For example, the termination proofs of programs involving stacks and production systems are often quite complicated and require much more subtle orderings and termination functions. Finding an appropriate ordering and termination function for such programs is a well-known challenge among researchers in the field of program verification. In the remainder of this paper, we shall demonstrate how the multiset ordering can sometimes permit the use of relatively simple and intuitive termination functions in otherwise difficult termination proofs.

In Section 2 we rigorously define the multiset ordering and prove that it is well-founded. In Section 3 we apply the multiset ordering to a number of termination proofs of programs. Then in Section 4 we use the multiset ordering to prove the termination of production systems.

2. The Multiset Ordering

A *partially-ordered set* $(S, \succ)$ consists of a set S and a transitive and irreflexive binary relation $\succ$ on elements of S. For example, both the set $\mathbb{Z}$ of all integers and the set $\mathbb{N}$ of nonnegative integers are ordered by the "greater-than" relation $>$. In general the ordering may be partial: For two distinct elements a and b of the set, we may have neither $a \succ b$ nor $b \succ a$.

A partially-ordered set $(S, \succ)$ is said to be *well-founded* if there can be no infinite descending sequences of elements $s_1 \succ s_2 \succ \cdots$ from the set S. Thus, the set $(N, >)$ is well-founded, since any descending sequence of natural numbers cannot go beyond 0. On the other hand, the set $(\mathbb{Z}, >)$ is not well-founded.

For a given partially-ordered set $(S, \succ)$, we consider *multisets* over S, i.e. unordered collections of elements that may have multiple occurrences of identical elements. We denote by $\mathcal{M}(S)$ the set of all finite multisets with elements taken from the set S and associate an ordering $\gg$ on $\mathcal{M}(S)$ that is induced by the given ordering $\succ$ on S.

In the following definition, as well as in the rest of this paper, set operators will denote their multiset analogs: The equality $A = B$ of two multisets, for example, means that any element occurring exactly n times in A, also occurs exactly n times in B, and *vice versa.* The union of two multisets $A \cup B$ is a multiset containing $m + n$ occurrences of any element occurring m times in A and n times in B. For example, the union of the multisets $\{2, 2, 4\}$ and $\{2, 0, 0\}$ is $\{2, 2, 4, 2, 0, 0\}$.

For a partially-ordered set $(S, \succ)$, the *multiset ordering* $\gg$ on $\mathcal{M}(S)$ is defined as follows:

$M \gg M'$

if for some multisets X, $Y \in \mathcal{M}(S)$, where $\{\ \} \neq X \subseteq M$,

$M' = (M - X) \cup Y$

and

$(\forall y \in Y)(\exists x \in X)\ x \succ y.$

In words, a multiset is reduced by the removal of at least one element (those in X) and their replacement with any finite number—possibly zero— of elements (those in Y), each of which is smaller than one of the elements that have been removed.

We must first show that $\gg$ is in fact a partial ordering, i.e. if $\succ$ is irreflexive and transitive, then $\gg$ is also:

(1) To show irreflexivity, we must show that there can be no multiset M such that $M \gg M$. Suppose that $M \gg M$; then there would be some nonempty finite multiset $X \subseteq M$ such that $(\forall y \in X)(\exists x \in X)x \succ y$. In other words, for every element of X there would be a distinct element of X greater than it, which is impossible for a finite X.

(2) To show transitivity of $\gg$, consider the following irreflexive relation $\gg'$ on multisets in $\mathcal{M}(S)$: $M \gg' (M - \{x\}) \cup Y$ if $(\forall y \in Y)x \succ y$. In other words, a finite multiset is reduced in the relation $\gg'$ by replacing a *single* element with zero or more smaller elements. Note that the multiset ordering $\gg$ is the (irreflexive) transitive closure of the relation $\gg'$, i.e. $M \gg M'$ if and only if M' can be obtained from M by replacing elements in M one by one. It follows that $\gg$ is transitive.

We have the following theorem.

THEOREM. *The multiset ordering* $(\mathcal{M}(S), \gg)$ *over* $(S, \succ)$ *is well-founded if and only if* $(S, \succ)$ *is well-founded.*

PROOF.

(1) "Only if" part. If $(S, \succ)$ is not well-founded, then there exists an infinite descending sequence $s_1 \succ s_2 \succ s_3 \succ \cdots$ of elements in S. The corresponding sequence of singletons $\{s_1\} \gg \{s_2\} \gg \{s_3\} \gg \cdots$ forms an infinite descending sequence of elements in $\mathcal{M}(S)$, and $(\mathcal{M}(S), \gg)$ is therefore not well-founded.

(2) "If" part. Assume that $(S, \succ)$ is well-founded. Let $S' = S \cup \{\perp\}$ be S extended with a least element $\perp$; i.e. for every element $s \in S$, $s \succ \perp$ in the ordering on S'. Clearly S' is well-founded if S is. Now suppose that $(\mathcal{M}(S), \gg)$ is not well-founded; therefore there exists an infinite descending sequence $M_1 \gg M_2 \gg M_3 \gg \cdots$ of multisets from $\mathcal{M}(S)$. We derive a contradiction by constructing the following tree. Each node in the tree is labeled with some element of S'; at each stage of the construction, the set of all leaf nodes in the tree forms a multiset in $\mathcal{M}(S')$.

Begin with a root node with children corresponding to each element of M_1. Since $M_1 \gg M_2$, there must exist multisets X and Y, such that $\{\ \} \neq X \subseteq M_1$, $M_2 = (M_1 - X) \cup Y$, and $(\forall y \in Y)(\exists x \in X)x \succ y$. Then for each $y \in Y$, add a child labeled y to the corresponding x. In addition, grow a child $\perp$ from each of the elements of X. (Since X is nonempty, growing $\perp$ ensures that even

if Y is empty, at least one node is added to the tree. Since Y is finite, the nodes corresponding to X each have a finite number of children.) Repeat the process for $M_2 \gg M_3$, $M_3 \gg M_4$, and so on.

Since at least one node is added to the tree for each multiset M_i in the sequence, were the sequence infinite, the tree corresponding to the sequence would also be infinite. But by Konig's Infinity Lemma, an infinite tree (with a finite number of children for each node) must have an infinite path. On the other hand, by our construction, all paths in the tree are descending in the well-founded ordering $\succ$ on S', and must be finite. Thus we have derived a contradiction, implying that the sequence $M_1, M_2, M_3, \cdots$ cannot be infinite. □

Remark. If $(S, \succ)$ is totally ordered, then for any two multisets $M, M' \in \mathcal{M}(S)$, one may decide whether $M \gg M'$ by first sorting the elements of both M and M' in descending order (with respect to the relation $\succ$) and then comparing the two sorted sequences lexicographically. For example, to compare the multisets {3, 3, 4, 0} and {3, 2, 1, 2, 0, 4}, one may compare the sorted sequences (4, 3, 3, 0) and (4, 3, 2, 2, 1, 0). Since (4, 3, 3, 0) is lexicographically greater than (4, 3, 2, 2, 1, 0), it follows that {3, 3, 4, 0} ≫ {3, 2, 1, 2, 0, 4}.

Remark. If $(S, \succ)$ is of order type α, then the multiset ordering $(\mathcal{M}(S), \gg)$ over $(S, \succ)$ is of order type ω^α. This follows from the fact that there exists a mapping ψ from $\mathcal{M}(S)$ onto ω^α that is one-to-one and order-preserving, i.e. if $M \gg M'$ for $M, M' \in \mathcal{M}(S)$, then the ordinal $\psi(M)$ is greater than $\psi(M')$. That mapping is

$$\psi(M) = \sum_{m \in M} \omega^{\varphi(m)},$$

where $\sum$ denotes the natural (i.e. commutative) sum of ordinals and φ is the one-to-one order-preserving mapping from S onto α.

Remark. Consider the case where there is a bound k on the number of replacement elements, i.e. restrict the ordering $\gg$ by taking the (irreflexive) transitive closure of the relation $M \gg' M'$ which holds if $|Y| < k$ when $M' = (M - X) \cup Y$. *Any termination proof using this bounded multiset ordering over $\mathbb{N}$ may be translated into a proof using* $(N, >)$. This may be done, for example, using the order-preserving function

$$\psi(M) = \sum_{n \in M} k^n$$

which maps multisets over the natural numbers into the natural numbers by summing the number k^n for every natural number n in a multiset M.

We turn now to consider *nested multisets*, by which we mean that the elements of the multisets may belong to some base set S, or may be multisets of elements of S, or may be multisets containing both elements of S and multisets of elements of S, and so on. For example,

{{1, 1}, {{0}, 1, 2}, 0}

is a nested multiset. More formally, given a partially-ordered set $(S, \succ)$, a *nested multiset over* S is either an element of S, or else it is a finite multiset of nested multisets over S. We denote by $\mathcal{M}^*(S)$ the set of nested multisets over S.

We define now a *nested multiset ordering* $\gg^*$ on $\mathcal{M}^*(S)$; it is a recursive version of the standard multiset ordering. For two elements $M, M' \in \mathcal{M}^*(S)$, we say that

$$M \gg^* M'$$

if

(i) $M, M' \in S$ and $M \succ M'$ (two elements of the base set are compared using $\succ$); or else

(ii) $M \notin S$ and $M' \in S$ (any multiset is greater than any element of the base set), or else

(iii) $M, M' \notin S$, and for some X, $Y \in \mathcal{M}^*(S)$, where $\{\} \neq X \subseteq M$,

$$M' = (M - X) \cup Y \quad \text{and} \quad (\forall y \in Y)(\exists x \in X)\, x \gg^* y.$$

For example, the nested multiset

{{1, 1}, {{0}, 1, 2}, 0}

is greater than

{{1, 0, 0}, 5, {{0}, 1, 2}, 0},

since {1, 1} is greater than both {1, 0, 0} and 5. The same nested multiset

{{1, 1}, {{0}, 1, 2}, 0}

is also greater than

{{{ }, 1, 2}, {5, 5, 2}, 5},

since {{0}, 1, 2} is greater than each of the three elements {{ }, 1, 2}, {5, 5, 2}, and 5.

Let $\mathcal{M}^i(S)$ denote the set of all nested multisets of *depth i*. In other words, $\mathcal{M}^0(S) = S$ and $\mathcal{M}^{i+1}(S)$ contains the multisets whose elements are taken from $\mathcal{M}^0(S)$, $\mathcal{M}^1(S)$, ..., $\mathcal{M}^i(S)$, with at least one element taken from $\mathcal{M}^i(S)$. Thus the set $\mathcal{M}^*(S)$ is the infinite union of the disjoint sets $\mathcal{M}^0(S)$, $\mathcal{M}^1(S)$, $\mathcal{M}^2(S)$, ... The following property holds:

PROPERTY. *For two nested multisets, M and M', if the depth of M is greater than the depth of M', then $M \gg^* M'$.*

In other words, the nested multisets in $\mathcal{M}^i(S)$ are all greater than those in $\mathcal{M}^j(S)$ under the ordering $\gg^*$, for any $i > j$. By the antisymmetry of $\gg^*$, it follows that if $M \gg^* M'$, then the depth of M' cannot be greater than the depth of M.

PROOF. This property may be proved by induction on the depth of M. It holds vacuously for M of depth 0. For the inductive step, assume that nested multisets of depth i are greater than nested multisets of depth less than i; we must show that a nested multiset M of depth $i + 1$ is greater than any nested multiset M' of lesser depth. If the depth of M' is 0, then $M' \in S$ while $M \notin S$, and therefore $M \gg^* M'$, as desired. If the depth of M' is less than $i + 1$ but greater than 0, then each of the

elements in M' is of depth less than i. The nested multiset M, on the other hand, is of depth $i + 1$ and must therefore contain some element of depth i. By the inductive hypothesis, that element is greater than each of the elements in M'. Again it follows that $M \gg^* M'$. □

The relation $\gg^*$ is a partial ordering; it can be shown to be both irreflexive and transitive. The following theorem gives the condition under which it is well-founded:

THEOREM. *The nested multiset ordering* $(\mathcal{M}^*(S), \gg^*)$ *over* $(S, \succ)$ *is well-founded if and only if* $(S, \succ)$ *is well-founded.*

PROOF.

(1) "Only if" part. If $(S, \succ)$ is not well-founded, then there exists an infinite descending sequence $s_1 \succ s_2 \succ s_3 \succ \cdots$ of elements in S. This sequence is also an infinite descending sequence of elements in $\mathcal{M}^*(S)$ under $\gg^*$, and $(\mathcal{M}^*(S), \gg^*)$ is therefore not well-founded.

(2) "If" part. In order to show that $(\mathcal{M}^*(S), \gg^*)$ is well-founded, it suffices to show that each $\mathcal{M}^i(S)$ is itself well-founded under $\gg^*$. For if it were assumed that $\mathcal{M}^*(S)$ were not well-founded, then there would exist an infinite descending sequence of nested multisets $M_1 \gg^* M_2 \gg^* \cdots$. By the above property (and the antisymmetry of $\gg^*$), the depth of any nested multiset M_{k+1} in the sequence cannot be greater than the depth of its predecessor M_k. Since that sequence is infinite, it must have an infinite subsequence of nested multisets all of the same depth i, which contradicts the well-foundedness of $\mathcal{M}^i(S)$.

We prove that each $(\mathcal{M}^i(S), \gg^*)$ is well-founded by induction on i: The ordering $\gg^*$ on $\mathcal{M}^0(S) = S$ is simply the ordering $\succ$ on S, and it follows that $(\mathcal{M}^0(S), \gg^*)$ is well-founded. For the inductive step, assume that each $(\mathcal{M}^j(S), \gg^*)$, $j < i$, is well-founded, and note that each of the elements of $\mathcal{M}^i(S)$ is a member of the union of $\mathcal{M}^0(S), \mathcal{M}^1(S), \ldots, \mathcal{M}^{i-1}(S)$. By the induction hypothesis, each of these $\mathcal{M}^j(S)$ is well-founded under $\gg^*$; therefore their union is also well-founded under $\gg^*$. Since the ordering $\gg^*$ on two nested multisets from $\mathcal{M}^i(S)$ is exactly the standard multiset ordering over that well-founded union, and since a multiset ordering is well-founded if the ordering on the elements is, it follows that $\mathcal{M}^i(S)$ is also well-founded under $\gg^*$. □

Remark. If $(S, \succ)$ is totally ordered, then two nested multisets over S may be compared by first recursively sorting them at all levels and then comparing them lexicographically.

Remark. We have seen above that for $(S, \succ)$ of order type α, the multiset ordering $(\mathcal{M}(S), \gg)$ is of order type ω^α. In a similar manner, it can be shown that the order type of $(\mathcal{M}^i(S), \gg^*)$ is

$$\left.\begin{matrix} \omega^\alpha \\ \cdot \\ \cdot \\ \cdot \\ \omega \\ \omega \end{matrix}\right\} i \text{ times},$$

the limit of which is the ordinal ϵ_0—provided that α is an ordinal less than ϵ_0. Thus *if* $(S, \succ)$ *is of order type less than* ϵ_0, *then* $(\mathcal{M}^*(S), \gg^*)$ *is of order type* ϵ_0. (Gentzen [3] used an ϵ_0 ordering to prove the termination of his normalization procedure for proofs in arithmetic.)

In the following two sections we apply the multiset ordering to problems of termination, first proving the termination of conventional programs, and then proving the termination of production systems.

3. Termination of Programs

The following basic theorem is commonly used to prove the termination of programs:

THEOREM (Floyd). *A program P with variables $\bar{x}$ ranging over a domain $\bar{D}$ terminates if and only if there exist*

(i) a set of labels $\mathcal{L}$ cutting all the loops in P,

(ii) a well-founded set $(W, \succ)$, and

(iii) a termination function τ mapping $\mathcal{L} \times \bar{D}$ into W, such that whenever control traverses a path from one label to another, the value of the termination function $\tau_L(\bar{x})$ decreases for the current label L and value of $\bar{x}$.

The justification is straightforward:

If the program does not terminate, then there exists an infinite sequence of label-value pairs $(L_1, \bar{d}_1), (L_2, \bar{d}_2), \ldots$, corresponding to the sequence of labels through which control passes during a nonterminating computation and the values of the variables at those points. Since the function τ decreases with each traversal of a path, it follows that $\tau_{L_1}(\bar{d}_1) \succ \tau_{L_2}(\bar{d}_2) \succ \cdots$ forms an infinite descending sequence in the set W, contradicting its well-foundedness.

On the other hand, if the program does terminate, then the set $(\mathcal{L} \times \bar{D}, \succ_p)$ is well-founded, where the relation $\succ_p$ is defined so that $(L, \bar{d}) \succ_p (L', \bar{d}')$ if the program can reach the label L with the value $\bar{d}$ before it reaches L' with the value $\bar{d}'$. Thus, if $\tau_L(\bar{x})$ returns the pair $(L, \bar{x})$, then with each traversal of a path, the current value of $\tau_L(\bar{x})$ decreases.

In the following examples, we prove the termination of programs using multiset orderings as the well-founded set.

Example 1. *Counting tips of binary trees.* Consider a simple program to count the number of tips—leaf nodes (without descendents)—in a binary tree. Each tree y that is not a tip has two subtrees, *left*(y) and *right*(y). The program is

```
S := (t)
c := 0
loop until S = ( )
    y := head(S)
    if tip(y) then S := tail(S)
                  c := c + 1
              else S := left(y) ∘ right(y) ∘ tail(S)
              fi
    repeat
```

It employs a stack S and terminates when S is empty. At

that point, the variable c is to contain the total number of tip nodes in the given tree t.

Initially the given tree is placed in the stack. With each iteration the subtree at the top of the stack is tested to determine whether it is a tip: if it is, then it is removed from the stack and the count is incremented by 1; if it is not a tip, then it is replaced in the stack by its two subtrees, so that the number of tips in each subtree may be counted.

The termination of this program may be proved using the well-founded set $(\mathbb{N}, >)$. The appropriate termination function is

$$\tau(S) = \sum_{s \in S} nodes(s),$$

where $nodes(s)$ is the total number of nodes in the subtree s—not just the tip nodes. To show that the value of τ decreases with each loop iteration, we must consider two cases: If $y = head(S)$ is a tip node, then that node is removed from the stack, and the sum is decreased by 1. If y is not a tip, then it is replaced by its two subtrees, $left(y)$ and $right(y)$. But y contains one node—the root—more than $left(y)$ and $right(y)$ combined, and again the sum is decreased.

Using the multiset ordering over trees, we can prove termination with the simple termination function

$$\tau(S) = \{s : s \in S\},$$

giving the multiset of trees appearing in the stack. The trees themselves are ordered by the natural well-found subtree ordering, i.e. any tree is greater than its subtrees. Thus removal of a tree from the stack decreases τ in the multiset ordering by removing an element, and the replacement of a tree with two smaller subtrees decreases τ by replacing one element in the multiset with two smaller elements.

This example is similar to the shunting yard example. In general, any program in which elements are repeatedly removed from a stack, queue, bag, etc., and replaced with any number of smaller elements (in some well-founded ordering) can be shown to terminate with the corresponding multiset ordering.

Example 2. *McCarthy's* 91*-function.* The following is a contrived program to compute the simple function

$$f(x) = \textit{if } x > 100 \quad \textit{then } x - 10 \quad \textit{else } 91,$$

over the set of integers $\mathbb{Z}$, in a round-about manner. Though this program is short, the proof of its correctness and termination are nontrivial, and for this reason it is often used to illustrate proof methods.

The program is:

```
n := 1
z := x
loop L: assert f(x) = f^n(z),   n ≥ 1
    if z > 100 then n := n − 1
                    z := z − 10
               else n := n + 1
                    z := z + 11
               fi
    until n = 0
    repeat
assert z = f(x).
```

The predicates $f(x) = f^n(z)$ and $n \geq 1$, in the **assert** clause at the head of the loop, are loop invariants; they hold whenever control is at label L. When the program terminates, the variable z contains the value of $f(x)$, since the loop is exited if control reaches the **until** clause with $n = 0$; at that point, $f(x) = f^0(z) = z$.

Using the conventional well-founded set $(\mathbb{N}, >)$, Katz and Manna [6] prove the termination of this program with the termination function

$$\tau(n, z) = -2 \cdot z + 21 \cdot n + 2 \cdot max(111, x)$$

at L.

For an alternative proof of termination, we consider the following well-founded partial-ordering $\succ$ on the integers:

$$a \succ b \quad \text{if and only if} \quad a < b \leq 111.$$

(This is the same ordering on integers as in the familiar structural-induction proof, due to Rod Burstall, of the recursive version of this program.) As the well-founded set, we use the set $(\mathcal{M}(\mathbb{Z}), \gg)$ of all multisets of integers, under the corresponding multiset ordering. The appropriate termination function τ at L yields a multiset in $\mathcal{M}(\mathbb{Z})$, and is defined as

$$\tau(n, z) = \{z, f(z), \ldots, f^{n-1}(z)\}.$$

We must show that for each loop iteration this function decreases. There are three cases to consider:

(1) $z > 100$ at L. In this case, the **then** branch of the conditional is executed and both n and z are decremented. When control returns to L (assuming that the loop has not been exited), we have, in terms of the old values of n and z,

$$\tau(n - 1, z - 10) = \{z - 10, f(z - 10), \ldots, f^{n-2}(z - 10)\}.$$

Since $z > 100$, we have $f(z) = z - 10$, and therefore

$$\tau(n - 1, z - 10) = \{f(z), f^2(z), \ldots, f^{n-1}(z)\}.$$

Thus, the value of the termination function τ has been decreased by removing the element z from the original multiset $\{z, f(z), \ldots, f^{n-1}(z)\}$.

(2) $90 \leq z \leq 100$ at L. In this case, the **else** branch is taken and both n and z are incremented, yielding

$$\tau(n + 1, z + 11) = \{z + 11, f(z + 11), f^2(z + 11), \ldots, f^n(z + 11)\}.$$

Since $z + 11 > 100$, we have $f(z + 11) = z + 1$ and $f^2(z + 11) = f(z + 1)$. Furthermore, either $z + 1 = 101$ or else $z + 1 \leq 100$, and in both cases $f(z + 1) = 91 = f(z)$ and consequently $f^2(z + 11) = f(z)$. Thus we get

$$\tau(n + 1, z + 11) = \{z + 11, z + 1, f(z), \ldots, f^{n-1}(z)\}.$$

Since $z < z + 1 < z + 11 \leq 111$, we have $z \succ z + 11$ and $z \succ z + 1$. Accordingly, the multiset has been reduced by replacing the element z with the two smaller elements, $z + 11$ and $z + 1$.

(3) $z < 90$ at L. The **else** branch is taken and we have

$$\tau(n+1, z+11) = \{z+11, f(z+11), f^2(z+11), \ldots, f^n(z+11)\}.$$

Since $z + 11 \leq 100$, we have $f(z + 11) = 91$ and $f^2(z + 11) = f(91) = 91 = f(z)$, and thus

$$\tau(n+1, z+11) = \{z+11, 91, f(z), \ldots, f^{n-1}(z)\}.$$

Again z has been replaced by two smaller elements (under the $\succ$ relation), $z + 11$ and 91.

Example 3. *Ackermann's function.* Ackermann's function $a(m, n)$ over pairs of natural numbers is defined recursively as

$a(m, n) \Leftarrow$ **if** $m = 0$ **then** $n + 1$
else if $n = 0$ **then** $a(m - 1, 1)$
else $a(m - 1, a(m, n - 1))$
fi fi.

The following iterative program computes this function:

```
S := (m)
z := n
loop L: assert a(m, n) = a(s_k, a(s_{k-1}, ..., a(s_2, a(s_1, z))...))
    y := head(S)
    S := tail(S)
    if y = 0 then z := z + 1
    else
    if z = 0 then S := (y - 1)∘S
                  z := 1
             else S := y∘(y - 1)∘S
                  z := z - 1
             fi fi
    until S = ( )
    repeat
assert z = a(m, n),
```

where the stack S is of the form $(s_1, s_2, \ldots, s_k)$ for some $k \geq 0$, $head(S) = s_1$, $tail(S) = (s_2, \ldots, s_k)$, and $y \circ S = (y, s_1, \ldots, s_k)$. This is achieved by keeping the relation

$$a(m, n) = a(s_k, a(s_{k-1}, \ldots, a(s_2, a(s_1, z)) \ldots))$$

invariantly true whenever control is at the head of the loop. Thus when the stack S is empty, the loop terminates with $a(m, n) = z$.

The underlying idea is to apply the recursive definition for $a(m, n)$ to the rightmost two elements of the sequence

$$s_k, \ldots, s_2, s_1, z.$$

The three branches of the conditional statement in the loop correspond to the three cases in the recursive definition, e.g. if $y = s_1 \neq 0$ and $z \neq 0$, then the sequence becomes $s_k, \ldots, s_2, s_1 - 1, s_1, z - 1$, since $a(s_1, z) = a(s_1 - 1, a(s_1, z - 1))$.

The termination of this program was proved by Manna and Waldinger [11] using the intermittent-assertion technique. We give here a proof using multisets.

Consider the set $N \times N$ of lexicographically-ordered pairs of natural numbers and the corresponding multiset ordering over $N \times N$. Let $y = head(S) = s_1$. The termination function at L is

$$\tau(S, z) = \{(s_k + 1, 0), (s_{k-1} + 1, 0), \ldots, (s_2 + 1, 0), (y, z)\}.$$

Thus $\tau(S, z)$ yields a multiset containing one pair per element in the stack S. Note that at L, the stack S is nonempty, and all the elements in S as well as z are nonnegative integers.

The proof considers three cases, corresponding to the three branches of the conditional in the loop:

(1) $y = 0$. If the loop is not exited, then the new value of τ at L is

$$\tau((s_2, \ldots, s_k), z+1) = \{(s_k + 1, 0), \ldots, (s_3 + 1, 0), (s_2, z + 1)\}.$$

This represents a decrease in τ under the multiset ordering, since the element (y, z) has been removed and the element $(s_2 + 1, 0)$ has been replaced by the smaller $(s_2, z + 1)$.

(2) $y \neq 0$ and $z = 0$. In this case we obtain

$$\tau((y-1, s_2, \ldots, s_k), 1) = \{(s_k + 1, 0), \ldots, (s_2 + 1, 0), (y - 1, 1)\}.$$

Thus the element (y, z) has been replaced by the smaller element $(y - 1, 1)$.

(3) $y \neq 0$ and $z \neq 0$. Here we have

$$\tau((y, y-1, s_2, \ldots, s_k), z-1) = \{(s_k + 1, 0), \ldots, (s_2 + 1, 0), (y, 0), (y, z - 1)\}.$$

The element (y, z) has been replaced by the two smaller elements $(y, 0)$ and $(y, z - 1)$.

Remark. The previous examples suggest the following heuristic for proving termination: Given a program over a domain $(D, \succ)$ that computes some function $f(x)$, if the program has a loop invariant of the form

$$f(x) = h(f(g_1(y)), f(g_2(y)), \ldots, f(g_n(y))),$$

then try the multiset ordering $(\mathcal{M}(D), \gg)$, and use the termination function

$$\tau(y) = \{g_1(y), g_2(y), \ldots, g_n(y)\}.$$

The idea underlying this heuristic is that τ represents the set of unevaluated arguments of some recursive expansion of the function f.

4. Termination of Production Systems

A *production system* Π (also called a *term-rewriting system*) over a set of expressions E is a (finite or infinite) set of rewriting rules, called *productions*. Each production is of the form

$$\pi(\alpha, \beta, \ldots) \rightarrow \pi'(\alpha, \beta, \ldots),$$

where π and π' are expression schemata containing variables $\alpha, \beta, \ldots$ ranging over E. (The variables appearing in π' must be a subset of those in π.) Instantiating the variables $\alpha, \beta, \ldots$ with expressions $a, b, \ldots$ in E, respectively, the rule indicates that an expression $\pi(a, b, \ldots) \in E$ may be replaced by the corresponding expression $\pi'(a, b, \ldots) \in E$.

A rule is applied in the following manner: Given an

expression $e \in E$ that contains a subexpression $\pi(a, b, \ldots)$, $\pi'(a, b, \ldots)$. We write $e \Rightarrow e'$ if the expression e' can be derived from e by a single application of some rule in Π to one of the subexpressions of e.

For example, the following is a production system that differentiates an expression, containing + and ·, with respect to x:

$$\boxed{\begin{array}{l} Dx \rightarrow 1 \\ Dy \rightarrow 0 \\ D(\alpha + \beta) \rightarrow (D\alpha + D\beta) \\ D(\alpha \cdot \beta) \rightarrow ((\beta \cdot D\alpha) + (\alpha \cdot D\beta)), \end{array}}$$

where y can be any constant or any variable other than x. Consider the expression

$$e = D(D(x \cdot x) + y).$$

We could apply either the third production to the outer D or the fourth production to the inner D. In the latter case, we obtain

$$e' = D(((x \cdot Dx) + (x \cdot Dx)) + y),$$

which now contains three occurrences of D. At this point, we can still apply the third production to the outer D, or we could apply the first production to either one of the inner D's. Applying the third production yields

$$(D((x \cdot Dx) + (x \cdot Dx)) + Dy).$$

Thus

$$e'' = D(D(x \cdot x) + y) \Rightarrow D(((x \cdot Dx) + (x \cdot Dx)) + y) \Rightarrow (D((x \cdot Dx) + (x \cdot Dx)) + Dy).$$

In general, at each stage in the computation there are many ways to proceed, and the choice is made nondeterministically. In our case, all choices eventually lead to the expression

$$((((1 \cdot 1) + (x \cdot 0)) + ((1 \cdot 1) + (x \cdot 0))) + 0),$$

for which no further application of a production is possible.

A production system Π *terminates* over E if there exist no infinite sequences of expressions $e_1, e_2, e_3, \ldots$ such that $e_1 \Rightarrow e_2 \Rightarrow e_3 \Rightarrow \cdots$ and $e_i \in E$. In other words, given any initial expression, execution always reaches a state for which there is no way to continue applying productions. The difficulty in proving the termination of a production system, such as the one for differentiation above, stems from the fact that while some productions (the first two) may decrease the size of an expression, other productions (the last two) may increase its size. Also, a production (the fourth) may actually duplicate occurrences of subexpressions. Furthermore, applying a production to a subexpression not only affects the structure of that subexpression but also affects the structure and size of higher level superexpressions, including the top-level expression. A proof of termination must hold for the many different possible sequences generated by the nondeterministic choice of productions and subexpressions.

The following theorem has provided the basis for most of the techniques used for proving the termination of production systems:

THEOREM (Manna and Ness). *A production system over E terminates if and only if there exists a well-founded set $(W, >)$ and a termination function $\tau: E \rightarrow W$, such that for any $e, e' \in E$,*

$$e \Rightarrow e' \quad \textit{implies} \quad \tau(e) > \tau(e').$$

To see why this theorem is true, suppose that the system does not always terminate although $e \Rightarrow e'$ implies $\tau(e) > \tau(e')$ in some well-founded set $(W, >)$. By definition, there must be an infinite sequence of expressions $e_i \in E$ such that $e_1 \Rightarrow e_2 \Rightarrow e_3 \Rightarrow \cdots$. In that case, there exists an infinite descending sequence $\tau(e_1) > \tau(e_2) > \tau(e_3) > \cdots$ in W, which contradicts the assumption that $>$ is a well-founded ordering. It follows that the system must terminate.

On the other hand, if the system does always terminate, then the set E is well-founded under the $\stackrel{+}{\Rightarrow}$ ordering, where $\stackrel{+}{\Rightarrow}$ is the (irreflexive) transitive closure of the relation $\Rightarrow$. Letting $(W, >)$ be $(E, \stackrel{+}{\Rightarrow})$ and τ be the identity function we clearly have $e \Rightarrow e'$ implies $\tau(e) = e \stackrel{+}{\Rightarrow} e' = \tau(e')$.

Several researchers have considered the problem of proving the termination of production systems. Among them: Gorn [4], Knuth and Bendix [7], and Plaisted [12, 13] define special well-founded orderings for this purpose; Manna and Ness [10] and Lankford [8] suggest the use of "monotonic" termination functions; Itturiaga [5] and Lipton and Snyder [9] give sufficient conditions under which certain classes of production systems terminate.

In the following examples, we illustrate the use of multisets in proving termination. We begin with a very simple example.

Example 1. *Associativity.* Consider the set of arithmetic expressions E constructed from some set of atoms (symbols) and the single operator +. The production system

$$\boxed{(\alpha + \beta) + \gamma \rightarrow \alpha + (\beta + \gamma)}$$

over E contains just one production which reparenthesizes a sum by associating to the right. For example, the expression $(a + b) + ((c + d) + g)$ becomes either $a + (b + ((c + d) + g))$ or $(a + b) + (c + (d + g))$, both of which become $a + (b + (c + (d + g)))$. Since the length of the expression remains constant when the production is applied, some other measure is needed to prove termination.

Solution 1 (*arithmetic*). Let the well-founded set be $(N, >)$. The termination function $\tau: E \rightarrow N$ maps expressions into the well-founded set, and is defined recursively as follows:

$$\tau(\alpha + \beta) = 2 \cdot \tau(\alpha) + \tau(\beta)$$

for expressions of the form $\alpha + \beta$, and

$$\tau(u) = 1$$

for any atom u. For example, the value of τ for the expression $(a + b) + ((c + d) + g)$ is 13.

The key point in the proof is that this function possesses the following two important properties (see [10]):

(1) The value of the termination function τ *decreases* for the subexpression that the production is applied to; i.e. for any possible values of α, β, and γ,

$$\tau((\alpha + \beta) + \gamma) > \tau(\alpha + (\beta + \gamma)).$$

This is so since

$$\tau((\alpha + \beta) + \gamma) = 2 \cdot \tau(\alpha + \beta) + \tau(\gamma) = 4 \cdot \tau(\alpha) + 2 \cdot \tau(\beta) + \tau(\gamma),$$

while

$$\tau(\alpha + (\beta + \gamma)) = 2 \cdot \tau(\alpha) + \tau(\beta + \gamma) = 2 \cdot \tau(\alpha) + 2 \cdot \tau(\beta) + \tau(\gamma),$$

and $\tau(\alpha)$ is at least 1.

(2) The function τ is *monotonic in each operand* in the sense that if

$$\tau(e_1) > \tau(e_2),$$

for some expressions e_1 and e_2, then for any expression e_3,

$$\tau(e_1 + e_3) > \tau(e_2 + e_3)$$

and

$$\tau(e_3 + e_1) > \tau(e_3 + e_2).$$

Thus if $e \Longrightarrow e'$, then some subexpression $(\alpha + \beta) + \gamma$ of e has been replaced by $\alpha + (\beta + \gamma)$ to obtain e'. We have $\tau((\alpha + \beta) + \gamma) > \tau(\alpha + (\beta + \gamma))$, by the first property. Therefore, by the monotonicity property, we obtain

$$e \Longrightarrow e' \quad \text{implies} \quad \tau(e) > \tau(e'),$$

and, by the preceding theorem, it follows that the production system must terminate.

Solution 2 (*multisets*). For this solution, we use the multiset ordering over the natural numbers, $(\mathcal{M}(N), \gg)$, and let the termination function $\tau: E \to \mathcal{M}(N)$ return the multiset of the lengths $|\alpha|$ of all the subexpressions of the form $\alpha + \beta$ in e, i.e.

$$\tau(e) = \{|\alpha| : \alpha + \beta \text{ in } e\}.$$

For example,

$$\tau((a + b) + ((c + d) + g)) = \{1, 3, 1, 3\},$$

since the left operands of the operator + are a, $a + b$, c, and $c + d$.

Again there are two crucial properties:

(1) The value of the termination function τ *decreases* with each application of a production, i.e.

$$\tau((\alpha + \beta) + \gamma) \gg \tau(\alpha + (\beta + \gamma))$$

Before an application of the production, the multiset $\tau((\alpha + \beta) + \gamma)$ includes one occurrence of $|\alpha + \beta|$ and one of $|\alpha|$, along with elements corresponding to the subexpressions of α, β, and γ. After application of the production, the new multiset $\tau(\alpha + (\beta + \gamma))$ includes one occurrence of $|\alpha|$ and one of $|\beta|$, leaving the subexpressions of α, β, and γ unchanged. Thus, the element $|\alpha + \beta|$ has been replaced by the smaller element $|\beta|$, and the multiset has accordingly been decreased.

(2) Since the production does not change the length of the expression it is applied to, i.e.

$$|\pi| = |\pi'|,$$

the length of superexpressions containing $(\alpha + \beta) + \gamma$ is also unchanged.

The only elements in $\tau(e)$ that are changed by the production are those in $\tau((\alpha + \beta) + \gamma)$, and they have been decreased by the production. Thus, $e \Longrightarrow e'$ implies that $\tau(e) \gg \tau(e')$.

Example 2. *Differentiation.* The following system symbolically differentiates an expression with respect to x:

Dx	$\to$	1
Dy	$\to$	0
$D(\alpha + \beta)$	$\to$	$(D\alpha + D\beta)$
$D(\alpha \cdot \beta)$	$\to$	$((\beta \cdot D\alpha) + (\alpha \cdot D\beta))$
$D(-\alpha)$	$\to$	$(-D\alpha)$
$D(\alpha - \beta)$	$\to$	$(D\alpha - D\beta)$
$D(\alpha/\beta)$	$\to$	$((D\alpha/\beta) - ((\alpha \cdot D\beta)/(\beta \uparrow 2)))$
$D(\ln \alpha)$	$\to$	$(D\alpha/\alpha)$
$D(\alpha \uparrow \beta)$	$\to$	$((D\alpha \cdot (\beta \cdot (\alpha \uparrow (\beta - 1)))) + (((\ln \alpha) \cdot D\beta) \cdot (\alpha \uparrow \beta)))$.

Solution 1 (*arithmetic*). Take $(N, >)$ as the well-founded set. Let the termination function $\tau: E \to N$ be defined by

$$\tau(\alpha \otimes \beta) = \tau(\alpha) + \tau(\beta),$$

where $\otimes$ is any of the binary operators $+$, $\cdot$, $-$, and $\uparrow$,

$$\tau(D\alpha) = \tau(\alpha)^2,$$
$$\tau(-\alpha) = \tau(\alpha) + 1,$$
$$\tau(\ln \alpha) = \tau(\alpha) + 1,$$

and

$$\tau(u) = 4,$$

for any atom u.

(1) For each of the nine productions $\pi \to \pi'$, the value of τ decreases, i.e. $\tau(\pi) > \tau(\pi')$. For example,

$$\tau(D(\alpha/\beta)) = (\tau(\alpha) + \tau(\beta))^2 = \tau(\alpha)^2 + \tau(\beta)^2 + 2 \cdot \tau(\alpha) \cdot \tau(\beta),$$

while

$$\tau(((D\alpha/\beta) - ((\alpha \cdot D\beta)/(\beta \uparrow 2)))) = \tau(\alpha)^2 + \tau(\beta)^2 + \tau(\alpha) + 2 \cdot \tau(\beta) + 4.$$

This is a decrease, since $\tau(\alpha)$, $\tau(\beta) \geq 4$ and therefore

$2 \cdot \tau(\alpha) \cdot \tau(\beta) \geq 4 \cdot \tau(\alpha) + 4 \cdot \tau(\beta) > \tau(\alpha) + 2 \cdot \tau(\beta) + 4.$

(2) τ is monotonic in each operand, for each of the operators, e.g. if $\tau(e_1)$ is greater than $\tau(e_2)$, then $\tau(e_1 \cdot e_3) = \tau(e_1) + \tau(e_3)$ is greater than $\tau(e_2 \cdot e_3) = \tau(e_2) + \tau(e_3)$, for any e_3.

It follows that $e \Rightarrow e'$ implies $\tau(e) > \tau(e')$.

Solution 2 (*multisets*). To prove termination, we use the multiset ordering over sequences of natural numbers. The sequences are compared under the well-founded *stepped lexicographic* ordering $\succ$, i.e. longer sequences are greater than shorter ones (regardless of the values of the individual elements), and equal length sequences are compared lexicographically. The termination function is

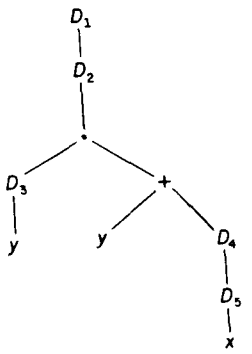
$$\tau(e) = \{(d_1(u), d_2(u), \ldots): u \text{ is an occurrence of an atom in } e\},$$

where $d_i(u)$ is the distance (number of operators) between u and the ith D enclosing it.

For example, consider the expression

$$e = DD(Dy \cdot (y + DDx)),$$

or in tree form (with the D's enumerated for expository purposes),



There are three atoms: y, y, and x. The left atom y contributes the element (0, 2, 3) to the multiset, since there are no operators between D_3 and y, there are two operators ($\cdot$ and D_3) between D_2 and y, and there are three operators (D_2, $\cdot$, and D_3) between D_1 and y. Similarly the other two atoms contribute (2, 3) and (0, 1, 4, 5). Thus

$$\tau(e) = \{(0, 2, 3), (2, 3), (0, 1, 4, 5)\}.$$

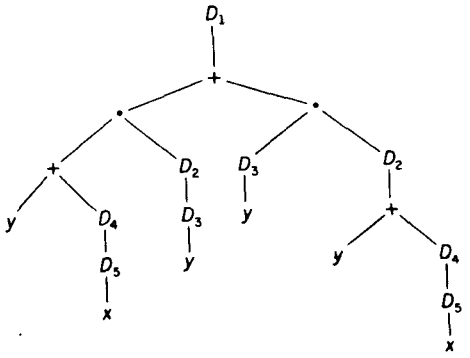
Applying the production

$$D(\alpha \cdot \beta) \rightarrow ((\beta \cdot D\alpha) + (\alpha \cdot D\beta))$$

to e yields

$$e' = D(((y + DDx) \cdot DDy) + (Dy \cdot D(y + DDx))).$$

In tree form (with the labeling of the D's retained), we have



and accordingly,

$$\tau(e') = \{(3), (0, 1, 5), (0, 1, 4), (0, 3), (1, 4), (0, 1, 3, 6)\}.$$

Thus $\tau(e) \gg \tau(e')$, since the element (0, 1, 4, 5) has been replaced by five shorter sequences and by the lexicographically smaller (0, 1, 3, 6).

In general, the following two properties hold:

(1) Applying any of the productions decreases τ. Consider, for example, what happens to the multiset $\tau(e)$ when the production

$$D(\alpha \cdot \beta) \rightarrow ((\beta \cdot D\alpha) + (\alpha \cdot D\beta))$$

is applied to some subexpression of e. Let u be an atom occurring in α. Applying the production results in replacing the sequence $s = (d_1(u), d_2(u), \ldots)$ corresponding to u, with two sequences, s' and s'', corresponding to the occurrences of u in $D\alpha$ and α, respectively. But s is greater than both s' and s'' in the stepped-lexicographic ordering: the sequence s'' is shorter than s, since there is one less D above u; the sequence s' is of the same length as s, but is lexicographically less, since a D has been pushed closer to u, while the distance from u to nearer D's remains unchanged. Similarly, the sequences corresponding to the atoms in β are replaced by two smaller sequences.

(2) The productions only affect the sequences in $\tau(e)$ corresponding to the atoms of the subexpression that they are applied to.

Therefore, for any application of a production, $e \Rightarrow e'$ implies $\tau(e) \gg \tau(e')$.

Solution 3 (*nested multisets*). Note that the arguments to D are reduced in length by each production. One would therefore like to prove termination using the well-founded set $(\mathcal{M}(\mathbb{N}), \gg)$ and a termination function that yields the multiset containing the lengths of the arguments of each occurrence of D, i.e. $\tau(e) = \{|\alpha| : D\alpha \text{ in } e\}$. The value of this function is decreased by the application of a production, i.e. $\tau(\pi) \gg \tau(\pi')$ for each of the productions $\pi \rightarrow \pi'$. The problem is that the length of superexpressions increases, since $|\pi'| > |\pi|$; applying a production to a subexpression of e may therefore increase $\tau(e)$.

To overcome this problem, we need a termination function that takes the nested structure of the expression

into consideration and gives more significance to more deeply nested subexpressions. Fortunately, this is exactly what nested multisets can do for us.

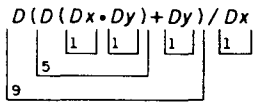
Let the well-founded set be the nested multisets over the natural numbers, $(\mathcal{M}^*(N), \gg^*)$, and let the termination function $\tau: E \to \mathcal{M}^*(N)$ yield $|\alpha|$ for each occurrence of $D\alpha$, while preserving the nested structure of the expression.[1]

$$\begin{aligned}\tau(\alpha \otimes \beta) &= \tau(\alpha) \cup \tau(\beta)\\ \tau(D\alpha) &= \{\{|\alpha|\} \cup \tau(\alpha)\},\\ \tau(-\alpha) &= \tau(ln\ \alpha) = \tau(\alpha),\end{aligned}$$

and

$$\tau(u) = \{\ \},$$

for any atom u. For example, the arguments of the six occurrences of D in the expression $D(D(Dx \cdot Dy) + Dy)/Dx$ are $D(Dx \cdot Dy) + Dy$, $Dx \cdot Dy$, x, y, y, and x. They are of lengths 9, 5, 1, 1, 1, and 1, respectively. Considering the nested depths of the D's, the structure of the expression is



Thus for

$$e = D(D(Dx \cdot Dy) + Dy)/Dx,$$

we have

$$\tau(e) = \{\ \{9, \{5, \{1\}, \{1\}\}, \{1\}\}, \{1\}\ \}.$$

(1) For each production $\pi \to \pi'$, we have $\tau(\pi) \gg^* \tau(\pi')$ under the nested multiset ordering. Consider, for example, the production

$$D(\alpha \cdot \beta) \to ((\beta \cdot D\alpha) + (\alpha \cdot D\beta)),$$

and let $\overline{\tau(\alpha)}$ and $\overline{\tau(\beta)}$ stand for the list of elements of the multisets $\tau(\alpha)$ and $\tau(\beta)$, respectively. Applying τ to the two sides of the production yields

$$\tau(D(\alpha \cdot \beta)) = \{\{|\alpha \cdot \beta|, \overline{\tau(\alpha)}, \overline{\tau(\beta)}\}\}$$

and

$$\begin{aligned}&\tau((\beta \cdot D\alpha) + (\alpha \cdot D\beta))\\ &\qquad = \{\overline{\tau(\beta)}, \{|\alpha|, \overline{\tau(\alpha)}\}, \overline{\tau(\alpha)}, \{|\beta|, \overline{\tau(\beta)}\}\}.\end{aligned}$$

This clearly is a decrease in $\mathcal{M}^*(N)$ regardless of the exact form of α and β, since $\{|\alpha \cdot \beta|, \overline{\tau(\alpha)}, \overline{\tau(\beta)}\}$ is greater than $\{|\alpha|, \overline{\tau(\alpha)}\}$ or $\{|\beta|, \overline{\tau(\beta)}\}$ and is also greater than each of the elements in $\overline{\tau(\alpha)}$ and $\overline{\tau(\beta)}$.

Consider, for example,

$$D(x \cdot Dy) \Rightarrow ((x \cdot DDy) + (Dy \cdot Dx)).$$

We have

$$\tau(D(x \cdot Dy)) = \{\{4, \{1\}\}\},$$

while

$$\tau(((x \cdot DDy) + (Dy \cdot Dx))) = \{\{2, \{1\}\}, \{1\}, \{1\}\}.$$

This is a decrease in the nested multiset ordering, since $\{4, \{1\}\}$ is greater than both $\{2, \{1\}\}$ and $\{1\}$. A similar argument applies to all of the other productions.

(2) It remains to ascertain what happens to the value of τ for superexpressions. The crucial point here is that the termination function gives greater weight to the more deeply nested D's by placing their lengths at a greater depth in the nested multiset. The decrease in τ for the subexpression to which the production is applied overshadows any increase in the length of a superexpression.

Consider, for example,

$$D(D(x \cdot x) + y) \Rightarrow D(((x \cdot Dx) + (x \cdot Dx)) + y),$$

The value of τ for the expression on the left is $\{\{6, \{3\}\}\}$, while for the right-hand side expression it is $\{\{11, \{1\}, \{1\}\}\}$. Note that this represents a decrease in the nested multiset ordering over N, despite the fact that the element 6, corresponding to the length of the top-level expression, has been increased to 11. This is the case since the production has replaced the element $\{3\}$ in the multiset $\{6, \{3\}\}$ by two occurrences of the smaller $\{1\}$, and $\{3\}$ is also greater than 11—or any number for that matter—on account of its greater depth.

Thus, $e \Rightarrow e'$ implies $\tau(e) \gg^* \tau(e')$.

In this section, we have illustrated the use of multiset and nested multiset orderings in proofs of termination of production systems. Along similar lines, using these orderings, one can give general theorems which express sufficient conditions for the termination of broad classes of production systems.

Acknowledgments. We thank R.S. Boyer, J. Doner, C. Goad, J. McCarthy, S. Ness, A. Pnueli, A. Pridor, W. Sherlis, and R. Weyhrauch for stimulating discussions and the referees for their helpful comments.

Received June 1978; revised November 1978

[1] An alternative solution would be to let $\tau(D\alpha) = \{\tau(d)\}$ and $\tau(u) = \{1\}$.

References

1. Dijkstra, E.W. A small note on the additive composition of variant functions. Note EWD592, Burroughs Corp., Neunen, The Netherlands, 1976.
2. Floyd, R.W. Assigning meanings to programs. Proc. Symp. in Applied Math., Vol. 19, Amer. Math. Soc., Providence, R.I., pp. 19–32.
3. Gentzen, G. New version of the consistency proof for elementary number theory (1938). In *The Collected Papers of Gerhart Gentzen*, M.E. Szabo, Ed., North-Holland, Amsterdam, 1969, pp. 252–286.
4. Gorn, S. Explicit definitions and linguistic dominoes. Proc. Conf. on Syst. and Comptr. Sci., London, Ontario, Sept. 1965, pp. 77–115.
5. Iturriaga, R. Contributions to mechanical mathematics. Ph.D. Th., Carnegie-Mellon U., Pittsburgh, Pa., May 1967.
6. Katz, S.M. and Manna, Z. A closer look at termination. *Acta Inform. 5*, 4 (1975), 333–352.

7. Knuth, D.E. and Bendix, P.B. Simple word problems in universal algebras. In *Computational Problems in Universal Algebras*, J. Leech, Ed., Pergamon Press, Oxford, 1969, pp. 263–297.
8. Lankford, D.S. Canonical algebraic simplification in computational logic. Memo ATP-25, Automatic Theorem Proving Project, U. of Texas, Austin, Texas, May 1975.
9. Lipton, R.J. and Snyder, L. On the halting of tree replacement systems. Proc. Conf. on Theoret. Comptr. Sci., Waterloo, Ontario, Aug. 1977, pp. 43–46.
10. Manna, Z. and Ness, S. On the termination of Markov algorithms. Proc. Third Hawaii Int. Conf. on Syst. Sci., Honolulu, Hawaii, Jan. 1970, pp. 789–792.
11. Manna, Z. and Waldinger, R.J. Is SOMETIME sometimes better than ALWAYS? Intermittent assertions in proving program correctness. *Comm. ACM 21*, 2 (Feb. 1978), 159–172.
12. Plaisted, D. Well-founded orderings for proving the termination of rewrite rules. Memo R-78-932, Dept. of Comptr. Sci., U. of Illinois, Urbana, Ill., July 1978.
13. Plaisted, D. A recursively defined ordering for proving termination of term rewriting systems. Memo R-78-943, Dept. of Comptr. Sci., U. of Illinois, Urbana, Ill., Oct. 1978.

Operating Systems
R. Stockton Gaines Editor

Secure Personal Computing in an Insecure Network

Dorothy E. Denning
Purdue University

A method for implementing secure personal computing in a network with one or more central facilities is proposed. The method employs a public-key encryption device and hardware keys. Each user is responsible for his own security and need not rely on the security of the central facility or the communication links. A user can safely store confidential files in the central facility or transmit confidential data to other users on the network.

Key Words and Phrases: personal computing, security, privacy, networks, public-key encryption
CR Categories: 2.12, 6.20

1. Introduction

Within the next ten years many of us will have personal computers linked to a central facility. The central facility (CF) will offer many attractive features: long-term storage, text editors, language processors, spe-

This work was supported in part by National Science Foundation Grant MCS77-04835.

Author's address: D.E. Denning, Computer Science Department, Purdue University, West Lafayette, IN 47907.