A complete term rewriting system for decimal integer arithmetic

H.R. Walters

Computer Science/Department of Software Technology

CS-R9435 1994

Report CS-R9435
ISSN 0169-118X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

A Complete Term Rewriting System for Decimal Integer Arithmetic

H.R. Walters

CWI

P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Abstract

We present a term rewriting system for decimal integers with addition and subtraction. We prove that the system is confluent and terminating.

CR Subject Classification (1991): D.1.1 [Programming Techniques]: Applicative (Functional) Programming; F.3.2 [Logics and Meanings of Programs]: Semantics of Programming Languages, *Algebraic approaches to semantics.*

AMS Subject Classification (1991): 68Q40: Symbolic computation, 68Q42: Rewriting Systems and 68Q65: Algebraic specification.

Keywords & Phrases: integers, term rewriting, specification languages, formal semantics, confluence, termination.

1. Introduction

In [CW91] a term rewriting system is presented of the integers with addition and multiplication. This specification is proved to be locally confluent using the LP theorem prover (Larch Prover). Termination is not established, and it is shown why common termination proof methods (Knuth Bendix Ordering and Recursive Path Ordering) fail on this rewrite system.

The termination of that rewrite system is listed as an open problem in [DJK93].

In [Wal91] and [BW89] a rewrite system is presented of the non-negative integers with addition, which is shown to be confluent and terminating. As with the rewrite system in [CW91], proof methods based on known reduction orderings fail. An ad-hoc proof is provided.

In the sequel we will extend that rewrite system with negative numbers and subtraction, and we will prove the confluence and termination of the extended rewrite system.

The relevance of such a specification is two-fold: the notation we present is identical to ordinary decimal arithmetic, which simplifies the use of this rewrite system when included, for example, in a library. In addition, base 10 arithmetic has logarithmic complexity as far as the representation of values is concerned, when compared to commonly used *successor-zero* specifications. An analysis of the space and time complexity of calculations falls outside the scope of this article.

Our notation and terminology are consistent with [Klo91, Wal91]. We consider ordinary (non-AC) matching, and we are interested exclusively in rewriting ground terms.

2. Integers

The key concept leading to the rewrite systems in [BW89] and [CW91] is the definition of a juxtaposition operator. This is the operator which combines digits into numbers. For example the number 12 is defined by applying the (invisible) juxtaposition operator to the digits 1 and 2. As we can see, the interpretation of this operator is defined as follows: $[\![xy]\!] = 10 * [\![x]\!] + [\![y]\!]$. The operator is left-associative, and may also be applied to other arguments than single digits. For example, $[\![123]\!] = [\![(12)3]\!] = 10 * [\![12]\!] + [\![3]\!] = 10*(10*[\![1]\!]+[\![2]\!])+[\![3]\!] = 100*[\![1]\!]+10*[\![2]\!]+[\![3]\!] = 123$. For our rewrite system we will keep this interpretation in mind, but the multiplication operator $*$ is not needed (as an auxiliary function); the rewrite rules operate on juxtaposition directly.

In normal forms, the right-hand argument of this operator is always a digit, but non-digits may occur in intermediate terms. Accordingly, our rewrite system is single-sorted.

A final choice is the base of our rewrite system. For human use (such as in this article) base 10 is appropriate; in many other cases base 2 is preferable due to the reduced number of equations. We use this rewrite system internally in a compiler for term rewriting systems, where base 2 is sufficient. That system consists of 25-or-so equations. The base is irrelevant for termination and confluence. In this article we will use base 10.

Next to the juxtaposition operator and the constants 0 through 9, the standard operators + and − (both unary and binary) are defined. Priorities of operations are: juxtaposition highest; then unary minus; then addition and subtraction (lowest). Where necessary parentheses have been added to improve readability.

$\langle 0\rangle$	$0x$	$\rightarrow$	x
$\langle 1.1\rangle$	$x(yz)$	$\rightarrow$	$(x+y)z$
$\langle 1.2\rangle$	$x(-(yz))$	$\rightarrow$	$-((y-x)z)$
$\langle 2.1.1\rangle$	$1(-1)$	$\rightarrow$	9
$\cdots$		$\cdots$	
$\langle 2.9.9\rangle$	$9(-9)$	$\rightarrow$	81
$\langle 3.0.1\rangle$	$x0(-1)$	$\rightarrow$	$x(-1)9$
$\cdots$		$\cdots$	
$\langle 3.9.9\rangle$	$x9(-9)$	$\rightarrow$	$x81$
$\langle 4\rangle$	$(-x)y$	$\rightarrow$	$-(x(-y))$
$\langle 5\rangle$	$--x$	$\rightarrow$	x
$\langle 6\rangle$	-0	$\rightarrow$	0
$\langle 7.1\rangle$	$0+x$	$\rightarrow$	x
$\langle 7.2\rangle$	$x+0$	$\rightarrow$	x
$\langle 8.1.1\rangle$	$1+1$	$\rightarrow$	2
$\cdots$		$\cdots$	
$\langle 8.9.9\rangle$	$9+9$	$\rightarrow$	18
$\langle 9.1\rangle$	$x+yz$	$\rightarrow$	$y(x+z)$
$\langle 9.2\rangle$	$xy+z$	$\rightarrow$	$x(y+z)$
$\langle 10.1\rangle$	$x+-y$	$\rightarrow$	$x-y$
$\langle 10.2\rangle$	$-x+y$	$\rightarrow$	$y-x$
$\langle 11.1\rangle$	$0-x$	$\rightarrow$	$-x$
$\langle 11.2\rangle$	$x-0$	$\rightarrow$	x
$\langle 12.1.1\rangle$	$1-1$	$\rightarrow$	0
$\cdots$		$\cdots$	
$\langle 12.9.9\rangle$	$9-9$	$\rightarrow$	0
$\langle 13.1\rangle$	$xy-z$	$\rightarrow$	$x(y-z)$
$\langle 13.2\rangle$	$x-yz$	$\rightarrow$	$-(y(z-x))$
$\langle 14.1\rangle$	$x--y$	$\rightarrow$	$x+y$
$\langle 14.2\rangle$	$-x-y$	$\rightarrow$	$-(x+y)$

First, we will sketch how the completeness and correctness of this rewrite system can be established.

Completeness

1. Proposition
The ground normal forms of this rewrite system are:

- the digit 0;
- the smallest set Φ such that:
  - for any non-zero digit x we have $x \in \Phi$
  - for any $y \in \Phi$ and any digit (possibly zero) z we have $yz \in \Phi$
- the set $\Phi^- = \{-\phi | \phi \in \Phi\}$.

For the most part this fact is easily verified. The single non-trivial case is a term of the form $\alpha\beta$. We will first focus on α:

- α is 0 or of the form $-x$. The term can be reduced by rules 0 and 4, respectively.
- α is a non-zero digit. Now consider β:
  - β is any digit. The term cannot be reduced at root level. If α does not contain a redex the term is a normal form.
  - β is of the form st. Rule 1.1 applies.
  - β is of the form $-\gamma$. If γ is 0 or of the form xy or $-z$, the term can be reduced with rule 6, 1.2 or 5, respectively. Otherwise γ is a non-zero digit, and one of rules 2.1.1 through 2.9.9 applies.
- α is of the form $\delta\varepsilon$. If β is a digit, or has the form st, the argument above applies. Otherwise β is of the form $-\gamma$. Again, if γ is 0 or of the form xy or $-z$ the term can be reduced as mentioned above. Hence γ is a non-zero digit.

  Now consider ε:
  - ε is a digit or of the form xy. The term can be reduced with rules 3.0.1 through 3.9.9 and rule 1.1.
  - ε is of the form $-x$. We are considering a term $\alpha\beta$ with β of the form $-d$ for some digit d and α of the form $\delta\varepsilon$ with ε of the form $-d'$. In other words: α has the same form as $\alpha\beta$. Clearly, further refinement of α leads to a recursion of the argument above. Hence, the only irreducible term not in our proposed set of normal forms ($\Phi \cup \Phi^- \cup \{0\}$) is of the form α_0, where $\alpha_n = \alpha_{n+1}(-d_n)$ and where each d_i is a non-zero digit. However, this term is infinite and we only consider finite terms.

Correctness
The integral numbers are a model for this rewrite system under the obvious interpretation; it is easily verified that all equations hold for integers under this interpretation.

Unique normal forms
Observe that no two distinct normal forms have the same value, in this model. From this and the previous observation we can conclude that the rewrite system has the ‘unique normal form’ property.

Termination
The termination proof for this rewrite system is non-trivial. We have not been able to establish any common ordering (reduction orderings, multi-set orderings, recursive path orderings; [DJ90]) upon which many termination proofs are based.

The structure of our proof will be as follows: first we will observe that the number of binary operators in any term cannot increase during rewriting. Then we will use induction on the number of binary operators. Verifying the finiteness of reduction sequences for terms with zero or one binary symbol (the induction base) is straightforward. Then we assume an infinite reduction sequence for a term with $n+1$ symbols. In this sequence, rules which decrease the number of function symbols cannot be applied; and infinitely many reductions must take place on the outermost binary symbol. Only two small sets of rules can be responsible for the latter reductions. A case based analysis finally refutes this possibility.

2. Lemma
The number of binary symbols does not increase by the application of any of the rules in the rewrite system.

This is easily verified.

3. Proposition (induction base)
No infinite reduction sequence exists for a term with less than two binary function symbols.

This is easily verified. For example, rule 4 can only be applied as long as the outermost symbol of its left-hand argument is a unary minus sign. Since no rule introduces that symbol in a term without binary operators this can only happen finitely many times.

4. Proposition
No infinite reduction sequence exists for a term with two or more binary function symbols.

Our induction hypothesis is: there is no term containing n or less binary function symbols for which an infinite reduction sequence exists.

5. Lemma
Suppose $s_1 \rightarrow s_2 \rightarrow \ldots$ is an infinite reduction sequence, where s_1 contains exactly $n+1$ binary symbols. Rules[1] 0, 7.*, 11.*, some of 2.*, some of 8.* and some of 12.* are not applied in this sequence.

These rules reduce the number of binary symbols; after this reduction the total number of binary symbols would have decreased, and no further infinite sequence can exist by the induction hypotheses. Hence, these rules can be ignored.

We will now consider the position at which the outermost binary function symbol of a term, or a sequence of terms, occurs. We will call this position the *obp* (for outermost binary position).

6. Lemma
In the sequence $s_1 \rightarrow s_2 \rightarrow \ldots$, infinitely many reductions must take place at the *obp*.

Suppose after a finite number of k reductions no further reductions occur at *obp* in $s_{k+1}, s_{k+2}, \ldots$. Hence, all reductions occur either on unary minus symbols above *obp* or in one of its arguments. Only finitely many unary minus symbols occur above *obp* in s_k, so infinitely many reductions occur inside either (or both) arguments of *obp*. This is in conflict with the induction hypothesis.

We will now consider how application of a rule can affect *obp*. Three categories of rules can be identified.

[1]we use the notation $n.*$ to indicate all rules whose index have prefix n.

I If *obp* is the juxtaposition operator, rules mapping it to the same symbol: 1.*, (the remaining rules in) 2.*, 3.* and 4. Rules 2.* concern terms with a single binary operator. When considering *obp*, they can be disregarded, since *obp* is the root of a term of $n+1>1$ binary symbols.

II If *obp* is plus or (binary) minus, rules mapping it again to plus or minus: 10.* and 14.*.

III Rules mapping an outermost plus or minus to an outermost juxtaposition operator: (the remaining rules in) 8.* and 12.*, and 13.* and 9.*.

7. Lemma
If, at some point, *obp* is juxtaposition, it will always continue to be juxtaposition.

8. Lemma
A rule in category III can only be applied once to *obp*.

Proof: both lemmata are trivial.

9. Corollary
In the sequence $s_1 \to s_2 \to \ldots$ there is an index m such that one of the following situations occurs:

- After index m *obp* is always juxtaposition
- After index m *obp* is always plus or minus

Proof: follows immediately from lemmata 7 and 8.

Before continuing we will first introduce the notion of *permutation* of reduction sequences.

10. Definition
Two (finite or infinite) reduction sequences $\alpha_0 \to \alpha_1 \to \ldots$ and $\beta_0 \to \beta_1 \to \ldots$ are *alternatives* if there exists an n such that $\forall i \neq n : \alpha_i = \beta_i$. Note that this implies that all rules being applied are identical (or have the same effect) with the exception of those just before and just after α_n.

In addition, two sequences are alternatives if a third sequence exists which is an alternative of both.

We now return to our proof, and we will consider the (infinitely) many (interspersed) reductions that must take place at *obp* after index m. If *obp* is $+$ or $-$, these reductions exclusively involve rules in category II (10.* and 14.*).

11. Lemma
Only finitely many reductions of class II can occur consecutively.

These rules require and remove an outermost unary minus symbol from one of their arguments. Without interleaved reductions in these arguments this can happen only finitely often.

12. Lemma
The reduction sequence $\ldots \to \alpha_{i-1} \overset{obp}{\to} \alpha \overset{\text{non}\,obp}{\to} \alpha_{i+1} \to \ldots$ and $\ldots \to \alpha_{i-1} \overset{\text{non}\,obp}{\to} \alpha' \overset{obp}{\to} \alpha_{i+1} \to \ldots$, where a reduction at *obp* by a class II rule and a subsequent reduction in one of the arguments are exchanged, are alternatives.

Proof: trivial.

13. Corollary
For any number n there exists an alternative of the sequence $s_{m'} \to \ldots$ such that at least n consecutive reductions not involving obp occur.

By Lemma 11 we have infinitely many non-obp reductions, possibly interspersed with finite sequences of obp reductions. Consider the last obp reduction before the n-th non-obp reduction. For each non-obp reduction after this reduction, up until the n-th, Lemma 12 defines an alternative in which this non-obp reduction occurs before that obp reduction. Transitively, an alternative is defined in which this obp reduction occurs beyond the n-th non-obp reduction. This can be repeated for every obp reduction before the n-th non-obp reduction.

14. Corollary
No reduction sequence $s_{m'} \to \ldots$ in which obp is always plus or minus exists.

Proof: trivial.

If, after m, obp is the juxtaposition operator, all reductions at obp are of class I (1.*, 3.* and 4).

15. Lemma
Rules 1.* can only be applied finitely often.

Rules 1.* reduce the number of binary symbols in the right-hand argument of obp. No rule increases this number.

We will consider the reduction sequence from $s_{m'}$, for some $m' > m$, in which only reductions for rules 3.* and 4 take place at obp.

16. Lemma
Between two applications of rules 3.*, rule 4 must be applied at least once.

When a rule 3.* is applied, the right-hand argument of obp must be of the form $-d$, where d is some non-zero digit. After application of the rule, the right-hand argument becomes d. Rules 3.* and 4 are the only rules applied at obp. Hence, rule 4 is applied at least once.

17. Lemma
The sequence of class I reductions contains infinitely many reductions for rules 3.*.

Otherwise infinitely many reductions of rule 4 occur, which are not interleaved with reductions of rules 3.*. But rule 4 removes a unary minus from its left-hand argument without otherwise altering that term. This leads to a contradiction.

18. Corollary
When we consider position obp, we can characterize the reduction sequence $s_{m'} \to s_{m'+1} \to \ldots$ as follows (here, a_i, b_i, c_i and d_i are digits, g_i and h_i are terms without binary operators, and t_i, u_i and v_i are arbitrary terms. Note that $t_i g_i = u_i^0$ and $c_i = v_i^0$):

$$\ldots \twoheadrightarrow t_k a_k(-b_k) \xrightarrow{3.*} t_k g_k c_k \twoheadrightarrow (-t_k^1)u_k^1 \xrightarrow[\text{note1}]{4}$$

$$\xrightarrow[\text{note1}]{4} t_k^1(-u_k^1) \twoheadrightarrow \ldots\ \ldots\ \ldots \twoheadrightarrow (-t_k^{n_k})u_k^{n_k} \xrightarrow[\text{note1}]{4}$$

$$\xrightarrow[\text{note1}]{4} t_k^{n_k}(-u_k^{n_k}) = t_{k+1}a_{k+1}(-b_{k+1}) \xrightarrow{3.*} \ldots$$

Not shown in this sequence are the applications of rule 5 in the terms in the right-hand argument of *obp*, if rule 4 is applied more than once (which means that at some point more than one unary minus symbols occur in that argument).

19. Lemma
The reduction sequence segment shown, contains an alternative of the segment $t_k g_k \twoheadrightarrow (-.. - t_{k+1})a_{k+1}$, where the number of minus signs is identical to the number of applications of rule 4.

The application of rule 4 are independent of those inside the arguments. Hence, these reductions (as well as those of rule 5, which are not shown) can be permuted to the end of each cycle.

To recapitulate, the existence of the infinite reduction sequence $s_{m'} \rightarrow ...$ implies the existence of an alternative which contains the infinite cascade of finite reduction segments $t_k g_k \twoheadrightarrow (-.. - t_{k+1})a_{k+1}$, where the number of minus symbols is at least one. It is important to observe that in these reduction segment, the t_i contain precisely $n-1$ binary symbols, where $n+1$ is the number of binary symbols we are considering in our induction hypothesis.

We will now concentrate on the *obp* of these finite reduction segments. These segments start with juxtaposition at *obp*, and, as mentioned, will therefore always have juxtaposition at *obp*. The rules that can be applied at *obp* are 1.*, 2.*, 3.* and 4.

20. Lemma
Rules 1.* are not applied in these segments.

These rules require a juxtaposition symbol in their right-hand argument. This symbol is initially not present, and there exists no rule which can introduce such a symbol.

21. Lemma
Rules 2.* are not applied in these segments.

After application of a rule 2.* at *obp* (say, in segment k), the term must have the form $-.. - d_1 d_2$ or $-.. - d$ for certain digits d_1, d_2 or d; where the number of minus signs is zero or more. Since this term is a normal form (modulo rule 5), it must be the final term in segment k. Hence, it must contain at least one minus sign and at least one juxtaposition operator, and t_{k+1} is a digit. The next segment starts off with a term of the form dg. This term cannot reduce to a term with an outermost unary minus, since only rules 2.* and 5 can be applicable. Segment $k+1$ cannot produce the required result, and rules 2.* are apparently never applied at *obp*.

Note that Lemma 17 remains valid in the current context, even though separate reduction segments are considered rather than one sequence. This is due to the fact that the terms t_i occur in segments i and $i+1$.

Now consider a segment l in which a rule 3.* is applied at *obp* at least once, and consider the tail of this segment after the last application of such a rule therein: $-.. - t_l^\alpha g_l^\alpha c_l^\alpha \twoheadrightarrow t_{l+1} d_{l+1}$. Also consider the first segment l' after l in which a rule 3.* is applied at *obp*, and consider the head of this segment before the first application of such a rule: $t_{l'} g_{l'} \twoheadrightarrow -.. - t_{l'}^\beta a_{l'}^\beta b_{l'}^\beta$.

22. Lemma
The indicated segment contains an alternative of the segment $t_l^\alpha g_l^\alpha \twoheadrightarrow \underline{t}_{l'}\,\underline{a}_{l'}$.

In the two indicated segments, and in all segments between l and l', only rule 4 is applied at *obp*. As mentioned earlier, applications of this rule, and of rule 5, are irrelevant (modulo alternative) when considering rules which apply below *obp*.

[1]Outermost unary minus signs are ignored here, since we look at *obp*

23. Corollary
There exists an infinite cascade of reduction segments of the form $t'_l g'_l \twoheadrightarrow (- .. - t'_{l+1}) a'_{l+1}$, where the number of minus symbols is at least one.

Above we established the existence of one such segment. As mentioned, rules 3.* must be applied infinitely often, and hence infinitely many such segments must exist. Lemma 16 is valid in the current context, and guarantees that rule 4 is applied at least once between applications of rules 3.*.

Observe that in these reduction segments, the terms t'_i contain precisely $n - 2$ binary symbols.

24. Corollary
The existence of a cascade as described, with terms of k binary symbols implies the existence of a similar cascade with terms of $k - 1$ symbols.

25. Corollary
No reduction sequence $s_{m'} \rightarrow ...$ in which obp is always juxtaposition, exists.

Proof: both corollaries are trivial.

We have now completed the proof of proposition 4 and have consequently established termination of our rewrite system.

Confluence

26. Lemma
A terminating TRS with the unique-normal-form property is confluent.

This is a standard result.

3. Conclusions

We have shown confluence and termination of our rewrite system for the integers. This rewrite system uses the same notation as ordinary decimal integer arithmetic.

Apart from the notational convenience, decimal representation has more efficient space and time requirements than the often-used *successor-zero* notation (logarithmic versus linear). It is easy to see that many computations based on our rewrite system also have logarithmic complexity, but we have not investigated this in detail.

In our opinion, our rewrite system compares favourably to that presented in [CW91], which uses the artificial constant MIN to represent the constant -1, and defines base 4 arithmetic (this latter fact is probably not significant). In addition, that rewrite system relies on AC matching, which requires more powerful implementations than ordinary matching, and which negatively influences the expected execution speed. This latter fact puts a shadow on the suggested gained efficiency due to logarithmic behaviour.

It must be mentioned, though, that in that rewrite system multiplication is defined, and term rewriting for open terms is considered. We have yet been unable to prove termination for our rewrite system extended with multiplication, although such a proof can be given for innermost reduction strategies, which guarantees termination in practice. We have not investigated rewriting open terms.

References

[BW89] L.G. Bouma and H.R. Walters. Implementing algebraic specifications. In J.A. Bergstra, J. Heering, and P. Klint, editors, *Algebraic Specification*, ACM Press Frontier Series, pages 199–282. The ACM Press in co-operation with Addison-Wesley, 1989. Chapter 5.

[CW91] D. Cohen and P. Watson. An efficient representation of arithmetic for term rewriting. In R. Book, editor, *Proceeding of the Fourth International Conference on Rewriting Techniques and Application (Como, Italy)*, LNCS 488, pages 240–251. Springer Verlag, Berlin, 1991.

[DJ90] N. Dershowitz and J.-P Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol B.*, pages 243–320. Elsevier Science Publishers, 1990.

[DJK93] N. Dershowitz, J.-P. Jouannaud, and J.W. Klop. More Problems in Rewriting. In C.Kirchner, editor, *Proceeding of the Fifth International Conference on Rewriting Techniques and Application (Montreal, Canada)*, LNCS 690. Springer Verlag, Berlin, 1993.

[Klo91] J.W. Klop. Term rewriting systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science, Vol II.* Oxford University Press, 1991.

[Wal91] H.R. Walters. *On Equal Terms, Implementing Algebraic Specifications.* PhD thesis, University of Amsterdam, 1991. Available by *ftp* from ftp.cwi.nl:/pub/gipe as Wal91.ps.Z.