Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

Rewrite systems for integer arithmetic

H.R. Walters and H. Zantema

Computer Science/Department of Software Technology

CS-R9576 1995

Report CS-R9576
ISSN 0169-118X

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

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

Rewrite Systems for Integer Arithmetic

H.R.Walters

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

H.Zantema

University of Utrecht
P.O.Box 80.089, 3508 TB Utrecht, The Netherlands

Abstract

We present three term rewrite systems for integer arithmetic with addition, multiplication, and, in two cases, subtraction. All systems are ground confluent and terminating; termination is proved by semantic labelling and recursive path order.

The first system represents numbers by successor and predecessor. In the second, which defines non-negative integers only, digits are represented as unary operators. In the third, digits are represented as constants. The first and the second system are complete; the second and the third system have logarithmic space and time complexity, and are parameterized for an arbitrary radix (binary, decimal, or other radices). Choosing the largest machine representable single precision integer as radix, results in unbounded arithmetic with machine efficiency.

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.

Note: Partial support received from the Foundation for Computer Science Research in the Netherlands (SION) under project 612-17-418, "Generic Tools for Program Analysis and Optimization".

Note: This report was also published as [WZ95].

1. Introduction

In [CW91] a term rewrite system is presented of base four integers with addition and multiplication. This rewrite system is proved to be locally confluent modulo associative-commutative multiplication and addition. Termination is not established and seems to be very hard to prove; it is listed as an open problem in [DJK93]. The system does not easily generalize to arbitrary number base. In [BW89] a rewrite system is presented of non-negative binary integers with addition, which is shown to be ground confluent and terminating. Both systems have logarithmic space and time complexity. Here, by space complexity we mean the space required to store a number as a function of its absolute value, and by time complexity we mean the number of steps required to reduce the addition, subtraction or multiplication of two such numbers, to normal form.

In this article we discuss term rewrite systems for integer arithmetic, where we desire (ground) confluence, termination, logarithmic complexity, a one-one correspondence between integers and ground normal forms, good readability of expressions, a minimal equational setting (non AC) and an arbitrary number base.

Unfortunately we shall not attain all these properties in a single rewrite system; we will present three rewrite systems which have most of the properties. All systems are pure term rewriting systems, unconditional and not taking terms modulo equations. All systems are ground confluent (which follows from having unique ground normal forms) and terminating (which is proved by recursive path order ([Der87]) and, where appropriate, semantic labelling ([Zan93])).

The first system is the common *successor-zero* system for natural numbers, extended with a *predecessor* function. This *successor-predecessor* system is referred to as SP. This system hardly exhibits the desired properties (no number base, bad readability, linear complexity); it is included only because it is well known, and commonly used to define integer arithmetic.

Confluence and termination are easily established. Drawbacks of this system are, firstly, that it has linear space and time complexity, or worse, in the case of multiplication, and secondly, that the successor-zero notation of numbers is hardly palatable to the human eye and is therefore mainly of theoretical importance.

The second system (DA, digit application) concerns non-negative integers only, but it does have all desired properties. It is confluent and terminating without assuming commutative-associativity of addition or multiplication; it has logarithmic complexity; and features human readable syntax of expressions (that is, the TRS itself is rather involved, but arithmetic expressions are simpele and easy to read).

In this system the digits occur as unary operators, written in postfix notation. An (invisible) constant of value zero is needed to represent numbers. This corresponds to the usual human syntax except for the number 0 which is represented by the invisible constant. The system needs some auxiliary functions.

The third system (JP, juxtaposition) defines integers consisting of sequences of digits, where digits are constants. It is ground confluent, but not confluent. It has logarithmic complexity, does not rely on auxiliary functions and features human-readable syntax of expressions. The system is terminating; the termination proof is quite involved. It is given in two levels: first the system obtained from ignoring all unary minus-signs is proved terminating, second the remaining rules for which this ignorance yields equality are proved terminating. Both of these termination proofs are given by transforming the system to an infinite labelled system by the technique of semantic labelling and proving termination of the labelled system by recursive path order (RPO) with status.

The systems DA and JP are presented for an arbitrary radix (number base); only the digits 0, and in one case 1, have special significance. All rules containing other digits are represented using *rule schemata*, which we will introduce in Section 3.

Apart from the formal logarithmic complexity, our rewrite systems have very good practical efficiency. For example, considering JP, discussed in Section 5:

- The binary version has 30 equations; the measured number of reduction steps for performing multiplication of two positive integers of b digits seems to be of the order of $b^2 \log b$.
- The decimal version of our rewrite system has essentially the same 30 equations, only some of them are parametrized. For instance, instead of the 81 rules for non-zero digit multiplication, a single rule schema is given. One of the instances is, for example, $7 * 8 \to 56$. The efficiency of multiplication is of the same order as in the binary version.
- Using the largest machine-representable single precision integer as radix, results in an implementation which provides unbounded integers at near-machine efficiency. This is discussed in Section 6.

Hence this system can be used as the basis for an implementation of integer arithmetic. Its efficiency is comparable with that of the more usual implementations. A major advantage of our approach is its extensibility: extension of the implementation by new functions like exponentiation can be done simply by only adding a few rewrite rules to the system.

In Section 2 we present the system SP. In an intermezzo in Section 3 we present rule schemata. Then, in Section 4 we present the system DA and in Section 5 the system JP. In Section 7 we present an overview of the various rewrite systems, and we discuss some conclusions.

Our notation and terminology are consistent with [Klo92]. We consider ordinary (non-AC) operators, and we are interested exclusively in rewriting finite terms.

2. Successor-predecessor integers

In the following rewrite system 0 is the constant *zero*, s and p are the functions representing successor (plus one) and predecessor (minus one), $+$, $*$ and $-$ have their usual meaning in arithmetic, and x, y and z are variables.

$\langle 1 \rangle$	$x+0$	$\rightarrow$	x	$\langle 10 \rangle$	$s(p(x))$	$\rightarrow$	x
$\langle 2 \rangle$	$x+s(y)$	$\rightarrow$	$s(x+y)$	$\langle 11 \rangle$	$p(s(x))$	$\rightarrow$	x
$\langle 3 \rangle$	$x+p(y)$	$\rightarrow$	$p(x+y)$				
$\langle 4 \rangle$	$x-0$	$\rightarrow$	x	$\langle 12 \rangle$	$(x-y)+y$	$\rightarrow$	x
$\langle 5 \rangle$	$x-s(y)$	$\rightarrow$	$p(x-y)$	$\langle 13 \rangle$	$(x+y)-y$	$\rightarrow$	x
$\langle 6 \rangle$	$x-p(y)$	$\rightarrow$	$s(x-y)$	$\langle 14 \rangle$	$s(x)+y$	$\rightarrow$	$s(x+y)$
$\langle 7 \rangle$	$x*0$	$\rightarrow$	0	$\langle 15 \rangle$	$s(x)-y$	$\rightarrow$	$s(x-y)$
$\langle 8 \rangle$	$x*s(y)$	$\rightarrow$	$(x*y)+x$	$\langle 16 \rangle$	$p(x)+y$	$\rightarrow$	$p(x+y)$
$\langle 9 \rangle$	$x*p(y)$	$\rightarrow$	$(x*y)-x$	$\langle 17 \rangle$	$p(x)-y$	$\rightarrow$	$p(x-y)$

The system SP

Termination of the TRS SP is established with recursive path ordering (RPO, [Der87]), taking the following precedence of operators: $* > \{+,-\} > \{s,p\}$.

Local confluence is verified using, for example, the Larch prover ([GG91]). Note that rules 12—17 are required only to establish local confluence; they are not required for ground confluence. Confluence follows from local confluence and termination.

One easily checks that 0, $s^n(0)$, $p^n(0)$ for $n = 1, 2, 3, \ldots$ are the ground normal forms. Hence ground normal forms correspond bijectively to integers.

The space complexity of this rewrite system is linear. That is, to represent a number n, a term of size $O[n]$ is required at least. Addition of two numbers, in absolute value not exceeding n, requires $O[n]$ reductions (worst case); multiplication requires $O[n^2]$ reductions. For example, the multiplication of 5 and 6 requires 46 reductions using the left-most inner-most reduction strategy.

3. Rule schemata

In the sequel we will consider rewrite systems in which non-zero digits occur, but we will do so for an arbitrary radix. The rules concerning these digits are very regular, and are similar for each radix.

In our system we express this fact by presenting such rules with *rule schemata.* A rule schema is a notational device defining a family of rules at once. The left-hand side is a term containing *schema variables* (e.g., δ); the right-hand side contains schema variables and *schema diagrams* representing common arithmetic operations (e.g., $\oplus$).

The meaning of a rule schema is the set of rules obtained by ranging all schema variables over, as the case may be, all digits or the non-zero digits, and by using a predetermined interpretation for the schema diagrams. We reserve the notation $\overset{\circ}{\delta}$, $\overset{\circ}{\delta}_1$, etc., for schema variables that range from 0 to the largest digit, and δ, δ_1, etc., for digits that range from 1 to the largest digit. That is, using $\mathcal{R}$ to represent the radix, we have $0 \leq \overset{\circ}{\delta} \leq \mathcal{R}-1$ and $1 \leq \delta \leq \mathcal{R}-1$.

Note that schema diagrams are meta notation and do not occur in the signature. We will discuss the individual schema diagrams where we use them. There, we will also discuss the specific properties we use to establish (ground) completeness.

4. Natural numbers with digits as unary operators

In our second rewrite system DA all digits are interpreted as unary postfix operators (hence, digit application). That is, a natural number followed by a digit is again a natural number. The base of this induction is a constant of value zero represented by the empty string. In the sequel we will make this string explicit – by surrounding it with parentheses, – whenever this is necessary for clarity. For example, we will write $x + () \rightarrow x$. A minor inconvienience is the fact that the number 0 is now represented as the empty string. Note that the *digit* 0 still appears as a postfix operator.

We will use the following three schema diagrams: $\oplus$, for addition modulo $\mathcal{R}$, $\copyright$ for addition carry, and $\odot$ for digit multiplication. Since digits are functions themselves, the meaning of schema diagrams is taken appropriately. We enclose schema diagrams that signify function applications rather than proper sub-terms in angle brackets.

For addition we define $\langle \delta^{\circ}_1 \oplus \delta^{\circ}_2 \rangle$ to be the postfix operator corresponding to last digit of $\delta^{\circ}_1 + \delta^{\circ}_2$. For example, $(x)\langle 5 \oplus 2\rangle$ denotes $(x)7$ and $(x)\langle 7 \oplus 8\rangle$ denotes $(x)5$. We denote this schema as a postfix operator.

In complement, the carry schema defines the addition carry. If $\delta^{\circ}_1 + \delta^{\circ}_2 < \mathcal{R}$ then $(x)\langle \delta^{\circ}_1 \copyright \delta^{\circ}_2\rangle$ signifies x; if $\delta^{\circ}_1 + \delta^{\circ}_2 \geq \mathcal{R}$ then $(x)\langle \delta^{\circ}_1 \copyright \delta^{\circ}_2\rangle$ signifies $s(x)$. Here s is an auxiliary function symbol defined in the system. For clarity, this schema is also denoted as a postfix operator, even though it signifies a function application, or no function application at all.

For multiplication, the schema diagram $\odot$ is introduced giving the result of multiplication of two digits. For example, $2 \odot 4$ denotes 8, while $5 \odot 6$ denotes 30.

In addition to the function s, the system defines one auxiliary unary operator for every digit. We will write these functions as $\star_{\delta^{\circ}}$. The meaning of $\star_{\delta^{\circ}}(x)$ is $\delta^{\circ} * x$. Note that each $\star_{\delta^{\circ}}$ is an auxiliary function; not a schema diagram.

$$
\begin{array}{lrcl}
\langle \mathit{1} \rangle & ()0 & \rightarrow & () \\
\langle \mathit{2} \rangle & () + x & \rightarrow & x \\
\langle \mathit{3} \rangle & x + () & \rightarrow & x \\
\langle \mathbf{4} \rangle & x\delta^{\circ}_1 + y\delta^{\circ}_2 & \rightarrow & (x+y)\langle \delta^{\circ}_1 \copyright \delta^{\circ}_2\rangle\langle \delta^{\circ}_1 \oplus \delta^{\circ}_2\rangle \\
\langle \mathit{5} \rangle & () * x & \rightarrow & () \\
\langle \mathbf{6} \rangle & x\delta^{\circ} * y & \rightarrow & (x * y)0 + \star_{\delta^{\circ}}(y) \\
 & & & \\
\langle \mathit{7} \rangle & s() & \rightarrow & 1 \\
\langle \mathbf{8} \rangle & s(x\delta^{\circ}) & \rightarrow & (x)\langle 1 \copyright \delta^{\circ}\rangle\langle 1 \oplus \delta^{\circ}\rangle \\
\langle \mathbf{9} \rangle & \star_{\delta^{\circ}}() & \rightarrow & () \\
\langle \mathbf{10} \rangle & \star_{\delta^{\circ}_1}(x\delta^{\circ}_2) & \rightarrow & (\star_{\delta^{\circ}_1}(x))0 + \delta^{\circ}_1 \odot \delta^{\circ}_2
\end{array}
$$

The system DA

Ground normal forms: The set of ground normal forms of this rewrite system is: $\mathcal{N} = \{()\} \cup \Phi$, where Φ is the smallest set satisfying $\Phi = \{()\delta\} \cup \{\alpha\delta^{\circ} | \alpha \in \Phi\}$. This fact is easily verified. Clearly ground normal forms correspond bijectively to non-negative integers.

Correctness: The natural numbers are a model for this rewrite system under the given interpretation; it is easily verified that all equations hold.

Termination: Termination is proved simply by recursive path order choosing the precedence

$$* > \{\star_{\delta^\circ}\} > + > \boldsymbol{s} > \{\overset{\circ}{\delta}\} > ().$$

Confluence: The only overlap between the left hand sides of the rules is between the rules 2 and 3, only giving the trivial critical pair $((),())$. Hence the system is locally confluent, and hence, by termination, confluent.

5. Integers with digits as constants

A natural next step is to extend the system of the last section to integers by adding rules for the unary and binary minus operator. Both for termination and for confluence this gives some complications. For instance, one expects a rule $(-x)5 \rightarrow -(p(x))5$ for a predecessor operator p for which there is a rule $p() \rightarrow -()1$. This yields the self-embedding reduction $(-())5 \rightarrow -(p())5 \rightarrow -(-()1)5$. Similar problems arise for choosing other representations for predecessor. Even if termination can be achieved, then confluence is a big problem. Adding rules like $- - x \rightarrow x$ yields many critical pairs that turn out to be not convergent.

From now on we leave the requirement of confluence, and only require ground confluence. For many applications, for example the implementation of ordinary arithmetic, ground confluence is sufficient.

We also leave the representation of digits being unary symbols; we choose digits to be constants and introduce a juxtaposition 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. Because of this juxtaposition operator this final system is called JP[1].

There are several reasons to change the representation. One reason is that it becomes even closer to human representation of integers. A second reason is that no auxiliary functions are needed any more. A third reason is that the tables for addition, subtraction and multiplication of two digits can be directly considered as rewite rules. For example, instead of the rule

$$\star_7(x8) \quad \rightarrow \quad (\star_7(x))0 + ((()5)6$$

we simply have the rule $7 * 8 \rightarrow 56$.

The interpretation of the juxtaposition operator is defined as follows: $[\![xy]\!] = \mathcal{R}*[\![x]\!]+[\![y]\!]$, where $\mathcal{R}$ is the radix. The operator is left-associative. For example, in base ten, $[\![123]\!] = [\![(12)3]\!] = 10*[\![12]\!]+[\![3]\!] = 10 * (10 * [\![1]\!] + [\![2]\!]) + [\![3]\!] = 100 * [\![1]\!] + 10 * [\![2]\!] + [\![3]\!] = 123$. For our rewrite system we will keep this interpretation in mind, but the multiplication operator $*$ is not strictly needed as an auxiliary function. In normal forms, the right-hand argument of the juxtaposition operator is always a digit, but non-digits may occur in intermediate terms.

The schema diagrams we will use are all similar: they represent the normal form of a simple function applied to one or two digits, resulting in a digit, or two juxtaposed digits, possibly with a minus sign. The schemas are: $\oplus$ for addition, $\ominus$ for subtraction, $\odot$ for multiplication, $\tilde{}$ for $\mathcal{R}$ complement (i.e., $\tilde{\delta} = \mathcal{R} - \delta$) and $\check{}$ for predecessor (i.e., $\check{\delta} = \delta - 1$). For example, the decimal meaning of $5 \odot 6$ is 30, and the meaning of $\tilde{3}$ is 7. If δ_1 is 1, it is understood that $\check{\delta_1}\tilde{\delta_2}$ signifies $\tilde{\delta_2}$ rather than $0\tilde{\delta_2}$.

To avoid parentheses we introduce priorities: juxtaposition has highest priority; then unary minus; then multiplication, and then then addition and subtraction. Where necessary parentheses have been added to improve readability. Rule schemata have bold indices.

[1] The decimal version of this system was already presented in [Wal94]. There, an *ad-hoc* termination proof was given.

$\langle \mathit{1} \rangle$	$0x$	$\rightarrow$	x	$\langle \mathit{17} \rangle$	$0 - x$	$\rightarrow$	$-x$
$\langle \mathit{2} \rangle$	$x(yz)$	$\rightarrow$	$(x + y)z$	$\langle \mathit{18} \rangle$	$x - 0$	$\rightarrow$	x
$\langle \mathit{3} \rangle$	$x(-(yz))$	$\rightarrow$	$-((y - x)z)$	$\langle \mathbf{19} \rangle$	$\delta_1 - \delta_2$	$\rightarrow$	$\delta_1 \ominus \delta_2$
$\langle \mathbf{4} \rangle$	$\delta_1(-\delta_2)$	$\rightarrow$	$\check{\delta}_1\tilde{\delta}_2$	$\langle \mathit{20} \rangle$	$xy - z$	$\rightarrow$	$x(y - z)$
$\langle \mathbf{5} \rangle$	$x0(-\delta)$	$\rightarrow$	$x(-1)\tilde{\delta}$	$\langle \mathit{21} \rangle$	$x - yz$	$\rightarrow$	$-(y(z - x))$
$\langle \mathbf{6} \rangle$	$x\delta_1(-\delta_2)$	$\rightarrow$	$x\check{\delta}_1\tilde{\delta}_2$	$\langle \mathit{22} \rangle$	$x - -y$	$\rightarrow$	$x + y$
$\langle \mathit{7} \rangle$	$(-x)y$	$\rightarrow$	$-(x(-y))$	$\langle \mathit{23} \rangle$	$-x - y$	$\rightarrow$	$-(x + y)$
$\langle \mathit{8} \rangle$	$- - x$	$\rightarrow$	x				
$\langle \mathit{9} \rangle$	-0	$\rightarrow$	0				
$\langle \mathit{10} \rangle$	$0 + x$	$\rightarrow$	x	$\langle \mathit{24} \rangle$	$0 * x$	$\rightarrow$	0
$\langle \mathit{11} \rangle$	$x + 0$	$\rightarrow$	x	$\langle \mathit{25} \rangle$	$x * 0$	$\rightarrow$	0
$\langle \mathbf{12} \rangle$	$\delta_1 + \delta_2$	$\rightarrow$	$\delta_1 \oplus \delta_2$	$\langle \mathbf{26} \rangle$	$\delta_1 * \delta_2$	$\rightarrow$	$\delta_1 \odot \delta_2$
$\langle \mathit{13} \rangle$	$x + yz$	$\rightarrow$	$y(x + z)$	$\langle \mathit{27} \rangle$	$x * yz$	$\rightarrow$	$(x * y)(x * z)$
$\langle \mathit{14} \rangle$	$xy + z$	$\rightarrow$	$x(y + z)$	$\langle \mathit{28} \rangle$	$xy * z$	$\rightarrow$	$(x * z)(y * z)$
$\langle \mathit{15} \rangle$	$x + -y$	$\rightarrow$	$x - y$	$\langle \mathit{29} \rangle$	$x * -y$	$\rightarrow$	$-(x * y)$
$\langle \mathit{16} \rangle$	$-x + y$	$\rightarrow$	$y - x$	$\langle \mathit{30} \rangle$	$-x * y$	$\rightarrow$	$-(x * y)$

The system JP

Ground normal forms: The set of ground normal forms of the system JP is: $\mathcal{N} = \{0\} \cup \Phi \cup \Phi^-$, where Φ is defined as the smallest set satisfying $\Phi = \Delta \cup \{\alpha\beta | \alpha \in \Phi \wedge \beta \in \{0\} \cup \Delta\}$, with Δ being the set of non-zero digits, and $\Phi^- = \{-\phi | \phi \in \Phi\}$. Hence ground normal forms correspond bijectively to integers.

This fact is easily verified. We will discuss the only non-trivial case. Consider a term of the form $\alpha\beta$.

- if α is 0 or of the form $-\eta$, the term can be reduced by rules 1 and 7, respectively.
- suppose α is a non-zero digit. Consider β:
  - if $\beta \in \{0\} \cup \Delta$, we have $\alpha\beta \in \Phi$.
  - if β is of the form $\sigma\tau$. Rule 2 applies.
  - suppose β is of the form $-\rho$. If $\rho \in \Delta$ rule 4 applies. If ρ is 0 or of the form $\sigma\tau$ or $-\sigma$, the term can be reduced with rule 9, 3 or 8, respectively.
- suppose α is of the form $\varepsilon\eta$. If β is a digit, or has the form $\sigma\tau$, the argument immediately above applies. Otherwise β is of the form $-\rho$. Again, if ρ is 0 or of the form $\sigma\tau$ or $-\sigma$ the term can be reduced as mentioned above. Hence ρ is a non-zero digit.

  To recapitulate: the remaining case is where $\alpha\beta$ is of the form $(\varepsilon\eta)(-\rho)$, where $\rho \in \Delta$.

  The left-hand side of this term has the same form as $\alpha\beta$, and our entire argumentation can be used recursively.

We see that the only irreducible terms not in $\mathcal{N}$ are of the form α_0, where $\alpha_n = \alpha_{n+1}(-d_n)$ and where each d_i is a non-zero digit. This is impossible for finite terms.

Correctness: The integer numbers are a model for this rewrite system under the given interpretation; it is easily verified that all equations hold.

Ground confluence: Observe that no two distinct normal forms have the same value, in this model. From this and the previous observation we can conclude that every term has a unique normal form. But then ground confluence is established if termination can be established; this will be shown in the next section. The system is not confluent, for example we have

$$x(\delta_1(-\delta_2)) \;\rightarrow\; x(\check{\delta_1}\tilde{\delta_2}) \;\rightarrow\; (x+\check{\delta_1})\tilde{\delta_2})$$

and

$$x(\delta_1(-\delta_2)) \;\rightarrow\; (x+\delta_1)(-\delta_2)$$

without having a common reduct. Also associativity and commutativity of + appear as critical pairs, hence the only chance to achieve confluence is by taking the + as an AC-operator. However, Knuth-Bendix completion of this system yields hundreds of critical pairs and seems not to terminate.

Restricting to natural numbers, i.e., taking only the rules in which no −-sign occurs, the system is locally confluent if + is taken as an AC-operator. However, modulo AC the termination proof is much more complicated, while for only the naturals we already had a confluent and terminating system DA without taking + modulo AC.

Termination: Termination of JP is not trivial. Standard techniques like recursive path order do not provide a termination proof. Termination of only the rules 2, 12 and 13 (these rules also occur in the system of [CW91]) can be proved by the Knuth-Bendix order, but this order can not handle duplicating rules like 27 and 28.

In the next section we give a termination proof of the full system using the technique of *semantic labelling* ([Zan93]).

5.1 Termination of JP

Outline of the proof In the proof no distinction is made between the binary plus and the binary minus, and no distinction is made between distinct digits. The proof is given in two levels: first we ignore the unary minus sign and prove termination of the rules for which the left hand side is different from the right hand side (R_1), next we prove termination of the rules for which equality is obtained (R_2). More precisely, termination of the full system JP follows from termination of both R_1 and R_2 and the observation

$$t \rightarrow_{\mathsf{JP}} u \;\Longrightarrow\; \phi(t) \rightarrow_{R_1} \phi(u) \;\vee\; (\phi(t) = \phi(u) \;\wedge\; \psi(t) \rightarrow_{R_2} \psi(u))$$

Here, ϕ and ψ are defined inductively by

$$\begin{array}{cc}
\phi(\overset{\circ}{\delta}) = 0 & \psi(\overset{\circ}{\delta}) = 0 \\
\phi(x) = x & \psi(x) = x \\
\phi(-t) = \phi(t) & \psi(-t) = -\psi(t) \\
\phi(tu) = \phi(t)\phi(u) & \psi(tu) = \psi(t)\psi(u) \\
\phi(t+u) = \phi(t-u) = \phi(t)+\phi(u) & \psi(t+u) = \psi(t-u) = \psi(t)+\psi(u) \\
\phi(t*u) = \phi(t)*\phi(u) & \psi(t*u) = \psi(t)*\psi(u)
\end{array}$$

for all variables x, digits $\overset{\circ}{\delta}$ and terms t, u. The systems R_1 and R_2 are constructed in such a way that the above property holds. For concluding termination of the full system JP it suffices to prove termination of both R_1 and R_2.

$\langle 1\rangle$	$0x$	$\rightarrow$	x
$\langle 2\rangle$	$x(yz)$	$\rightarrow$	$(x+y)z$
$\langle 3\rangle$	$x(yz)$	$\rightarrow$	$(y+x)z$
$\langle 10,17\rangle$	$0+x$	$\rightarrow$	x
$\langle 11,18\rangle$	$x+0$	$\rightarrow$	x
$\langle 12,19\rangle$	$0+0$	$\rightarrow$	0
$\langle 12\rangle$	$0+0$	$\rightarrow$	00
$\langle 13\rangle$	$x+yz$	$\rightarrow$	$y(x+z)$
$\langle 14,20\rangle$	$xy+z$	$\rightarrow$	$x(y+z)$
$\langle 21\rangle$	$x+yz$	$\rightarrow$	$y(z+x)$
$\langle 24\rangle$	$0*x$	$\rightarrow$	0
$\langle 25\rangle$	$x*0$	$\rightarrow$	0
$\langle 26\rangle$	$0*0$	$\rightarrow$	00
$\langle 26\rangle$	$0*0$	$\rightarrow$	0
$\langle 27\rangle$	$x*yz$	$\rightarrow$	$(x*y)(x*z)$
$\langle 28\rangle$	$xy*z$	$\rightarrow$	$(x*z)(y*z)$

The system R_1

$\langle 4\rangle$	$0(-0)$	$\rightarrow$	00
$\langle 5\rangle$	$x0(-0)$	$\rightarrow$	$x(-0)0$
$\langle 6\rangle$	$x0(-0)$	$\rightarrow$	$x00$
$\langle 7\rangle$	$(-x)y$	$\rightarrow$	$-(x(-y))$
$\langle 8\rangle$	$--x$	$\rightarrow$	x
$\langle 9\rangle$	-0	$\rightarrow$	0
$\langle 15,22\rangle$	$x+-y$	$\rightarrow$	$x+y$
$\langle 16\rangle$	$-x+y$	$\rightarrow$	$y+x$
$\langle 23\rangle$	$-x+y$	$\rightarrow$	$-(x+y)$
$\langle 29\rangle$	$x*(-y)$	$\rightarrow$	$-(x*y)$
$\langle 30\rangle$	$(-x)*y$	$\rightarrow$	$-(x*y)$

The system R_2

For the termination proofs for both R_1 and R_2 we shall use the technique of semantic labelling. The version we need is described in the next section.

Semantic labelling This technique makes use of the fact that a TRS with some semantics can be transformed into another (labelled) TRS such that the original TRS terminates if and only if the labelled TRS terminates. The termination proof is then given by proving termination of the labelled TRS, which is often done by recursive path order.

Let R be any term rewrite system over a signature $\mathcal{F}$ and a set $\mathcal{X}$ of variable symbols. Let $\mathcal{M}=(M,\{f_{\mathcal{M}}\}_{f\in\mathcal{F}})$ be an $\mathcal{F}$-algebra. Let $\geq$ be any partial order on M for which all operations $f_{\mathcal{M}}$ are weakly monotone in all coordinates.

For $\sigma:\mathcal{X}\rightarrow M$ the term evaluation $[\sigma]:\mathcal{T}(\mathcal{F},\mathcal{X})\rightarrow M$ is defined inductively by

$$\begin{aligned}[\sigma](x) &= x^{\sigma},\\ [\sigma](f(t_1,\ldots,t_n)) &= f_{\mathcal{M}}([\sigma](t_1),\ldots,[\sigma](t_n))\end{aligned}$$

for $x\in\mathcal{X}, f\in\mathcal{F}, t_1,\ldots,t_n\in\mathcal{T}(\mathcal{F},\mathcal{X})$.

We require that $\mathcal{M}$ is a *quasi-model* for R, i.e., $[\sigma](l)\geq[\sigma](r)$ for all $\sigma:\mathcal{X}\rightarrow M$ and all rules $l\rightarrow r$ of R (if we have $[\sigma](l)=[\sigma](r)$ then it is called a *model*).

Next we introduce labelling of operation symbols: choose for every $f\in\mathcal{F}$ a corresponding non-empty set S_f of labels. Now the new signature $\overline{\mathcal{F}}$ is defined by

$$\overline{\mathcal{F}}=\{f_s|f\in\mathcal{F},s\in S_f\},$$

where the arity of f_s is defined to be the arity of f. An operation symbol f is called *labelled* if S_f contains more than one element. For unlabelled f the set S_f containing only one element can be left implicit; in that case we write f instead of f_s. Note that $\overline{\mathcal{F}}$ can be infinite, even if $\mathcal{F}$ is finite.

We assume that every set S_f is provided with a well-founded partial order $\geq$. Choose for every $f\in\mathcal{F}$ a map $\pi_f:M^n\rightarrow S_f$, where n is the arity of f. We require π_f to be weakly monotone in all coordinates. It describes how a function symbol is labelled depending on the values of its arguments as interpreted in $\mathcal{M}$. For unlabelled f this function π_f can be left implicit. We extend the labelling of operation symbols to a labelling of terms by defining $\mathsf{lab}:\mathcal{T}(\mathcal{F},\mathcal{X})\times M^{\mathcal{X}}\rightarrow\mathcal{T}(\overline{\mathcal{F}},\mathcal{X})$ inductively by

$$\begin{aligned}\mathsf{lab}(x,\sigma) &= x,\\ \mathsf{lab}(f(t_1,\ldots,t_n),\sigma) &= f_{\pi_f([\sigma](t_1),\ldots,[\sigma](t_n))}(\mathsf{lab}(t_1,\sigma),\ldots,\mathsf{lab}(t_n,\sigma))\end{aligned}$$

for $x \in \mathcal{X}, \sigma : \mathcal{X} \to M, f \in \mathcal{F}, t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$.

Now $\overline{R}$ is defined to be the TRS over $\overline{\mathcal{F}}$ consisting of the rules

$$\mathsf{lab}(l, \sigma) \to \mathsf{lab}(r, \sigma)$$

for all $\sigma : \mathcal{X} \to M$ and all rules $l \to r$ of R. Note that this system can be infinite, even if R is finite.

Finally the TRS Decr over $\overline{\mathcal{F}}$ is defined to consist of the rules

$$f_s(x_1, \ldots, x_n) \to f_{s'}(x_1, \ldots, x_n)$$

for all $f \in \mathcal{F}$ and all $s, s' \in S_f$ satisfying $s > s'$. Here $>$ denotes the strict part of $\geq$. Now we are ready to state the main result of semantic labelling:

Theorem

Let $\mathcal{M}$ be a quasi-model for a TRS R over $\mathcal{F}$. Let $\overline{R}$ and Decr be as above for any choice of S_f and π_f. Then R is terminating if and only if $\overline{R} \cup \mathsf{Decr}$ is terminating.

For the proof we refer to [Zan93].

For both R_1 and R_2 we shall give a quasi-model $\mathcal{M}$, and S_f and π_f such that the corresponding infinite system $\overline{R} \cup \mathsf{Decr}$ can be proved terminating by using RPO with status over a well-founded precedence.

Termination of R_1 We denote the juxtaposition operator by '$\cdot$', for clarity.

As the quasi-model $\mathcal{M}$ and the sets $S.$ and S_+ we choose the strictly positive integers with the usual order. We choose

$$0_{\mathcal{M}} = 1, \;\; x \cdot_{\mathcal{M}} y = x +_{\mathcal{M}} y = x + y, \;\; x *_{\mathcal{M}} y = x * y$$

for all $x, y \in M$. One easily checks that indeed $[\sigma](l) \geq [\sigma](r)$ for all $\sigma : \mathcal{X} \to M$ and all rules $l \to r$ of R_1, hence indeed $\mathcal{M}$ is a *quasi-model* for R_1. Next we choose

$$\pi.(x, y) = \pi_+(x, y) = x + y$$

for all $x, y \in M$. All functions involved are weakly monotone in all coordinates. Now the infinite system $\overline{R_1} \cup \mathsf{Decr}$ consists of the rules

$\langle 1 \rangle$	$0 \cdot_i x$	$\to$	x	for all $i > 1$
$\langle 2 \rangle$	$x \cdot_i (y \cdot_j z)$	$\to$	$(x +_k y) \cdot_i z$	for $j < i$ and $k < i$
$\langle 3 \rangle$	$x \cdot_i (y \cdot_j z)$	$\to$	$(y +_k x) \cdot_i z$	for $j < i$ and $k < i$
$\langle 10, 17 \rangle$	$0 +_i x$	$\to$	x	for all $i > 1$
$\langle 11, 18 \rangle$	$x +_i 0$	$\to$	x	for all $i > 1$
$\langle 12, 19 \rangle$	$0 +_2 0$	$\to$	0	
$\langle 12 \rangle$	$0 +_2 0$	$\to$	$0 \cdot_2 0$	
$\langle 13 \rangle$	$x +_i (y \cdot_j z)$	$\to$	$y \cdot_i (x +_k z)$	for $j < i$ and $k < i$
$\langle 14, 20 \rangle$	$(x \cdot_j y) +_i z$	$\to$	$x \cdot_i (y +_k z)$	for $j < i$ and $k < i$
$\langle 21 \rangle$	$x +_i (y \cdot_j z)$	$\to$	$y \cdot_i (z +_k x)$	for $j < i$ and $k < i$
$\langle 24 \rangle$	$0 * x$	$\to$	0	
$\langle 25 \rangle$	$x * 0$	$\to$	0	
$\langle 26 \rangle$	$0 * 0$	$\to$	$0 \cdot_2 0$	
$\langle 26 \rangle$	$0 * 0$	$\to$	0	
$\langle 27 \rangle$	$x * (y \cdot_i z)$	$\to$	$(x * y) \cdot_j (x * z)$	where j is a multiple of i
$\langle 28 \rangle$	$(x \cdot_i y) * z$	$\to$	$(x * z) \cdot_j (y * z)$	where j is a multiple of i
	$x \cdot_i y$	$\to$	$x \cdot_j y$	for $j < i$
	$x +_i y$	$\to$	$x +_j y$	for $j < i$.

For instance, for any $\sigma : \mathcal{X} \to M$ we obtain from rule 2 a labelled rule $x \cdot_i (y \cdot_j z) \to (x +_k y) \cdot_i z$ where $i = [\sigma](x) + [\sigma](y) + [\sigma](z)$, $j = [\sigma](y) + [\sigma](z)$, $k = [\sigma](x) + [\sigma](y)$, indeed satisfying $j < i$ and $k < i$.

This infinite system $\overline{R_1} \cup \mathsf{Decr}$ is proved terminating by RPO with status by choosing the well-founded precedence

$$* > \cdots > +_i > \cdot_i > +_{i-1} > \cdot_{i-1} > \cdots.$$

Here $\cdot_i$ has the lexicographic status from right to left.

Termination of R_2 As the quasi-model $\mathcal{M}$ and the set $S.$ we again choose the positive integers with the usual order. Only juxtaposition will be labelled, the other operation symbols remain unlabelled. We choose

$$0_{\mathcal{M}} = 1, \;\; -_{\mathcal{M}}x = x + 1, \;\; x \cdot_{\mathcal{M}} y = x, \;\; x +_{\mathcal{M}} y = x *_{\mathcal{M}} y = x + y$$

for all $x, y \in M$. One easily checks that indeed $[\sigma](l) \geq [\sigma](r)$ for all $\sigma : \mathcal{X} \to M$ and all rules $l \to r$ of R_2, hence indeed $\mathcal{M}$ is a *quasi-model* for R_2. Next we choose

$$\pi.(x, y) = x$$

for all $x, y \in M$. All functions involved are weakly monotone in all coordinates. Now the infinite system $\overline{R_2} \cup \mathsf{Decr}$ consists of the rules

⟨*4*⟩	$0 \cdot_1 (-0)$	$\to$	$0 \cdot_1 0$	
⟨*5*⟩	$x \cdot_i 0 \cdot_i (-0)$	$\to$	$x \cdot_i (-0) \cdot_i 0$	for all i
⟨*6*⟩	$x \cdot_i 0 \cdot_i (-0)$	$\to$	$x \cdot_i 0 \cdot_i 0$	for all i
⟨*7*⟩	$(-x) \cdot_{i+1} y$	$\to$	$-(x \cdot_i (-y))$	for all i
⟨*8*⟩	$- - x$	$\to$	x	
⟨*9*⟩	-0	$\to$	0	
⟨*15*, *22*⟩	$x + -y$	$\to$	$x + y$	
⟨*16*⟩	$-x + y$	$\to$	$y + x$	
⟨*23*⟩	$-x + y$	$\to$	$-(x + y)$	
⟨*29*⟩	$x * (-y)$	$\to$	$-(x * y)$	
⟨*30*⟩	$(-x) * y$	$\to$	$-(x * y)$	
	$x \cdot_i y$	$\to$	$x \cdot_j y$	for $j < i$.

This system is proved terminating by RPO with status by choosing any precedence satisfying

$$\cdots > \cdot_{i+1} > \cdot_i > \cdot_{i-1} > \cdots > \cdot_1 > - \quad \text{and} \quad + > - \quad \text{and} \quad * > -.$$

For $\cdot_i$ we choose the lexicographic status from right to left and for $+$ we choose the multiset status.

5.2 Complexity

The common complexity measure for binary multiplication is the number of bit operations as a function of the number of bits in both arguments. If we focus on the number of digit multiplications we have to count the number of applications of rule 26. One can show that multiplication of two numbers of b_1 and b_2 digits respectively takes $b_1 * b_2$ of these digit multiplications. More precisely, any reduction of $\langle n_1 \rangle * \langle n_2 \rangle$ to normal form, where $\langle n_i \rangle$ represents the normal form of a positive number of b_i digits, takes $b_1 * b_2$ applications of rule 26. This coincides with common algorithms ([AHU84] p. 62).

In [AHU84] two algorithms are given with better asymptotical behavior. Firstly, the *divide-and-conquer* algorithm ([AHU84] pp. 62—65) is $O[b^{\log 3}]$. Detailed inspection of the algorithm, however, reveals that the implicit constants render our specification and the divide and conquer algorithm comparable for $b \lesssim 32$ bits. Secondly, the Shönhage-Strassen algorithm, which is based on fast Fourier transform, has complexity $O[b \cdot \log b \cdot \log \log b]$. However, the constant in this complexity result is so unwieldy that this result is of no practical use. Note that these complexity results are in the number b of bits, which is logarithmic in the value n of the argument.

In the context of term rewriting, the number of reductions is a more appropriate measure. A worst-case bound for this number could perhaps be established, but this bound would also regard exotic, irregular reduction strategies. Most implementations however, have a fixed, regular reduction strategy, which could have a better asymptotical behavior.

Considering the left-most innermost strategy only, our multiplication algorithm can be shown to be $O[\log^3 n] = O[b^3]$. Experimental measurements suggest however that the actual number of rewrite steps used for binary multiplication is close to $b^2 \log b$, which is for practical use comparable with the common algorithms. For example, the multiplication of two 10-bit numbers requires in the order of 500 reduction steps; that of two ten digit numbers in the order of 1000 reduction steps.

5.3 Discussion

The rewrite system JP is an efficient, practical system for integers arithmetic. Its human-readable syntax makes it easy to use, and the absence of auxiliary functions make it esthetically interesting.

There are two flaws to this system: firstly, it is not confluent and secondly, the fact that compound terms occur as the right-hand argument of the juxtaposition operator requires the intuitively unexpected rules 1—3.

6. Implementation of rule schemata

Rule schemata can be implemented in three ways:

1. Literal inclusion. This is in fact the method by which humans implement arithmetic: tables for multiplication and addition are committed to memory, together with various distributive laws.

   This method is suitable for small radices, but becomes impractical as the number of rules rises (in JP, base ten already requires 414 rules for all tables).

2. Auxiliary functions. The introduction (in JP) of successor and predecessor functions *on digits* allows basic operations to be specified using $O[\mathcal{R}]$ rules and only affects the complexity within a constant. This method decreases the practical efficiency, and renders the system less clear. Nevertheless, if 1 is impractical and 3 is impossible, it is the only alternative.

3. External functions. Every computer supports fixed precision integer arithmetic, which could be used to compute all needed instances of rule schemata on the fly. Clearly, this requires additional capabilities in the implementation. One framework for the correct use of external functions in term rewrite systems is discussed in [Wal90].

We will discuss this third alternative in some more detail.

Using built-in integers is profitable (if it is at all possible) under one condition: the time to apply a single rewrite rule should be in the same order as that to do the actual calculation (this is usually the case) and that to translate digits from their TRS representation to the machine representation, and to translate the result back (this depends on the implementation).

An interesting idea[2] is the following. If $\mathcal{R}$ is chosen to be the largest single precision integer plus one, then all schema diagrams and rule schemata can be implemented using double precision arithmetic. The TRS uses the machine integers directly for all practical values, but as soon as values larger than $\mathcal{R}$-1 occur, they are represented using the juxtaposition operator, or nested postfix application. This results in a highly efficient implementation of unbounded integers: *every* calculation on integers in the single-precision range (typically $[-2^{15}, 2^{15}-1]$, or $[-2^{31}, 2^{31}-1]$) requires a single reduction step and an amount of time comparable to only a few reduction steps in order to allow for conversions, and calculations on larger values require a number of steps which is cubic in the logarithm of the arguments, but now this logarithm is base 2^{15}, or even base 2^{31}. For example, if $\mathcal{R} = 2^{15}$ the number 9876543210 is represented as (9)(6496)(5866); the multiplication 9876543210$*$1234567890 is computed using in the order of 100 reduction steps.

[2] Thanks to J.F.Th.Kamperman.

7. Conclusions

In the table below, we have listed the three presented rewrite systems with all discussed properties. We have included the system from [CW91] since the four systems are to some degree complementary. The indication of readability refers to readability of *expressions*, and reflects the authors' opinions. The indication of complexity refers to the order of number of rewrite steps required for the multiplication of two numbers not exceeding n.

The entry marked ‡, relating to the confluence of the system [CW91], must be interpreted as follows: if the system is indeed terminating (which is not yet proved) then it is confluent modulo associativity and commutativity of $*$ and $+$.

system	range	conf.	term.	compl.	readability	radix
[CW91]	Int	‡	unproved	$\log n$	reasonable	4
SP	Int	yes	yes	n	poor	–
DA	Nat	yes	yes	$\log n$	good	any
JP	Int	ground	yes	$\log n$	good	any

We have presented three rewrite systems for integer arithmetic with addition, multiplication, and in two cases subtraction. We have shown ground confluence and termination.

In the last system, the common, human readable notation for arithmetic can be used. The latter two systems have logarithmic complexity and can be used for any radix.

Termination of the last system turned out to be non-trivial. We gave a proof in which the system was split up into two levels. For both remaining systems we gave a termination proof first by transforming it to an infinite labelled system by the technique of semantic labelling and finally proving termination of the labelled system by recursive path order (RPO) with status.

In our opinion, our second rewrite system is perfect for natural number arithmetic, and our final rewrite system is perfect for ground integer arithmetic. Richard Kennaway (personal communication) proposed a confluent and terminating rewrite system for integer arithmetic. Although it can be used for implementing addition, subtraction and multiplication it does not meet the basic requirement that ground normal forms correspond bijectively to integers. The construction of a complete rewrite system with logarithmic complexity satisfying this basic requirement remains an open problem.

References

[AHU84] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison Wesley, 1984.

[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.

[Der87] N. Dershowitz. Termination of rewriting. *J. Symbolic Computation*, 3(1&2):69–115, Feb./April 1987. Corrigendum: 4, 3, Dec. 1987, 409-410.

[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.

[GG91] S.J. Garland and J.V. Guttag. *A Guide to LP, The Larch Prover.* MIT, November 1991.

[Klo92] J.W. Klop. Term rewriting systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science, Volume 2.*, pages 1–116. Oxford University Press, 1992.

[Wal90] H.R. Walters. Hybrid implementations of algebraic specifications. In H. Kirchner and W. Wechler, editors, *Proceedings of the Second International Conference on Algebraic and Logic Programming*, volume 463 of *Lecture Notes in Computer Science*, pages 40–54. Springer-Verlag, 1990.

[Wal94] H.R. Walters. A complete term rewriting system for decimal integer arithmetic. Technical Report CS-9435, Centrum voor Wiskunde en Informatica, 1994. Available by *ftp* from ftp.cwi.nl:/pub/gipe as Wal94.ps.Z.

[WZ95] H.R. Walters and H. Zantema. Rewrite systems for integer arithmetic. In Jieh Hsiang, editor, *Rewriting Techniques and Applications*, number 914 in Lecture Notes in Computer Science, pages 324—338. Springer-Verlag, 1995.

[Zan93] H. Zantema. Termination of term rewriting by semantic labelling. Technical Report RUU-CS-93-24, Utrecht University, July 1993. *Accepted for publication in Fundamenta Informaticae.*