Basic arithmetic by rewriting and its complexity

Hans Zantema

December, 2003

Constructors

There are three basic number types: pos representing positive integers, nat representing non-negative integers, and int representing all integers.

For the positive integers there are three constructor symbols, having the following types.

$$\begin{array}{lcllll} 1 & : & & \to & \mathsf{pos} & \\ \mathbf{.0} & : & \mathsf{pos} & \to & \mathsf{pos} & (\lambda x \cdot 2x) \\ \mathbf{.1} & : & \mathsf{pos} & \to & \mathsf{pos} & (\lambda x \cdot 2x+1) \end{array}$$

Every positive integers can be uniquely represented as a ground term over these three symbols. This representation corresponds to the usual binary notation in which a postfix notation for **.0** and **.1** is used. For instance, the number 29 is 11101 in binary notation, and is written as 1**.1.1.0.1** as a postfix ground term. Since we will also use symbols for which prefix notation is most convenient, and mixing postfix and prefix notation does not increase readability, we will always use prefix notation, in which 29 is written as **.1**(**.0**(**.1**(**.1**(1)))).

For the non-negative integers there are two constructor symbols, having the following types.

$$\begin{array}{lcllll} 0 & : & & \to & \mathsf{nat} & \\ c_{pn} & : & \mathsf{pos} & \to & \mathsf{nat} & (\lambda x \cdot x) \end{array}$$

Hence the number 0 is written as 0, and every other non-negative integer is written as $c_{pn}(t)$ where t is the pos-representation of the number. The operation c_{pn} is the conversion from positive integers to non-negative integers.

For the integers there are two constructor symbols, having the following types.

$$\begin{array}{lcllll} c_{ni} & : & \mathsf{nat} & \to & \mathsf{int} & (\lambda x \cdot x) \\ - & : & \mathsf{pos} & \to & \mathsf{int} & (\lambda x \cdot -x) \end{array}$$

Hence an integer is written as $c_{ni}(t)$ where t is a ground term of type nat if its value is ≥ 0, and it is written as $-(t)$ where t is a ground term of type pos if it is negative.

For all three basic number types there is a bijective correspondence between the constructor ground terms and the set that the basic type intends to represent. This was one main requirement of the integer representation that we are looking for.

The conversion operations c_{pn} and c_{ni} may be left implicit if for all symbols involved the types are known. For instance, if $f : \mathsf{pos} \to \mathsf{pos}$ and $g : \mathsf{int} \to \mathsf{int}$, then $g(f(\cdots))$ should be read as $g(c_{ni}(c_{pn}(f(\cdots))))$. Although these conversion operations may be left implicit we shall always write them explicitly.

In order to be able to use equalities and inequalities between numbers, we also need the basic type bool for booleans. For the booleans we only have two constants:

$$\begin{array}{lcl} \mathbf{F} & : \to & \mathsf{bool} \\ \mathbf{T} & : \to & \mathsf{bool} \end{array}$$

intended to represent false and true, respectively.

Summarizing, written in BNF we have the following constructor ground terms:

$$\begin{array}{lcl} \mathsf{pos} & ::= & 1 \mid \mathbf{.0}(\mathsf{pos}) \mid \mathbf{.1}(\mathsf{pos}) \\ \mathsf{nat} & ::= & 0 \mid c_{pn}(\mathsf{pos}) \\ \mathsf{int} & ::= & c_{ni}(\mathsf{nat}) \mid -(\mathsf{pos}) \\ \mathsf{bool} & ::= & \mathbf{F} \mid \mathbf{T}. \end{array}$$

Defined symbols and rewrite rules

Apart from the constructor symbols we introduced until now there are other symbols called defined symbols. For all defined symbols that we introduce we give rewrite rules is such a way that the term rewrite system consisting of all of these rules have the following convenient properties:

- For every rule the root of the left hand side is a defined symbol, and all other symbols of the left hand side are constructor symbols.
- One every ground term containing at least one defined symbol a rewrite rule is applicable.
- If on a term for which all non-root symbols are constructor symbols two rules are applicable, then the result of the corresponding one-step reductions is equal.
- Every rule is *left-linear*, i.e, in the left hand side every variable occurs at most once.
- The term rewrite system is *terminating*, i.e., it does not allow infinite reductions. Even more, we want bounds on the lengths of the reductions corresponding to the complexity of usual integer computations.

One consequence of these properties is that the term rewrite system is weakly orthogonal and hence confluent, not only for ground terms but also for open terms. Hence every term t rewrites to its unique *normal form* $N(t)$, the term obtained form t by applying rewriting as long as possible. Due to termination such a normal form will always be obtained, and due to confluence it is unique, i.e., it does not depend on choices among different possible rewrite steps. In particular for every ground term its normal form is a ground constructor term: the unique ground constructor term representing the same value as the original ground term.

We will use $x_p, y_p, \ldots$ for variables of type pos, $x_n, y_n, \ldots$ for variables of type nat and $x_i, y_i, \ldots$ for variables of type int.

For all of the basic operations we introduce a version for each of the basic types as far as they are relevant. Sometimes we need a few auxiliary symbols.

Successor and predecessor

Symbols:

$$\begin{array}{lcllll} \mathsf{succ}_p & : & \mathsf{pos} & \to & \mathsf{pos} & (\lambda x \cdot x + 1) \\ \mathsf{succ}_n & : & \mathsf{nat} & \to & \mathsf{pos} & (\lambda x \cdot x + 1) \\ \mathsf{succ}_i & : & \mathsf{int} & \to & \mathsf{int} & (\lambda x \cdot x + 1) \\ \mathsf{pred}_i & : & \mathsf{int} & \to & \mathsf{int} & (\lambda x \cdot x - 1) \end{array}$$

Auxiliary symbol:

$$\mathsf{pred}_p \ : \ \mathsf{pos} \ \to \ \mathsf{pos} \quad (\lambda x \cdot \max(1, x - 1))$$

Rewrite rules for positive successor:

$$\begin{array}{rcl} \mathsf{succ}_p(1) & \to & \mathbf{.0}(1) \\ \mathsf{succ}_p(\mathbf{.0}(x_p)) & \to & \mathbf{.1}(x_p) \\ \mathsf{succ}_p(\mathbf{.1}(x_p)) & \to & \mathbf{.0}(\mathsf{succ}_p(x_p)) \end{array}$$

Rewrite rules for natural successor:

$$\begin{array}{rcl} \mathsf{succ}_n(0) & \to & 1 \\ \mathsf{succ}_n(c_{pn}(x_p)) & \to & \mathsf{succ}_p(x_p) \end{array}$$

Rewrite rules for integer successor:

$$\begin{array}{rcl} \mathsf{succ}_i(c_{ni}(x_n)) & \to & c_{ni}(c_{pn}(\mathsf{succ}_n(x_n))) \\ \mathsf{succ}_i(-(1)) & \to & c_{ni}(0) \\ \mathsf{succ}_i(-(\mathbf{.0}(x_p))) & \to & -(\mathsf{pred}_p(\mathbf{.0}(x_p))) \\ \mathsf{succ}_i(-(\mathbf{.1}(x_p))) & \to & -(\mathsf{pred}_p(\mathbf{.1}(x_p))) \end{array}$$

Rewrite rules for positive predecessor:

$$\begin{aligned}
\mathsf{pred}_p(1) &\to 1\\
\mathsf{pred}_p(\mathbf{.0}(1)) &\to 1\\
\mathsf{pred}_p(\mathbf{.0}(\mathbf{.0}(x_p))) &\to \mathbf{.1}(\mathsf{pred}_p(\mathbf{.0}(x_p)))\\
\mathsf{pred}_p(\mathbf{.0}(\mathbf{.1}(x_p))) &\to \mathbf{.1}(\mathsf{pred}_p(\mathbf{.1}(x_p)))\\
\mathsf{pred}_p(\mathbf{.1}(x_p)) &\to \mathbf{.0}(x_p)
\end{aligned}$$

Rewrite rules for integer predecessor:

$$\begin{aligned}
\mathsf{pred}_i(c_{ni}(0)) &\to -(1)\\
\mathsf{pred}_i(c_{ni}(c_{pn}(1))) &\to c_{ni}(0)\\
\mathsf{pred}_i(c_{ni}(c_{pn}(\mathbf{.0}(x_p)))) &\to c_{ni}(c_{pn}(\mathsf{pred}_p(\mathbf{.0}(x_p))))\\
\mathsf{pred}_i(c_{ni}(c_{pn}(\mathbf{.1}(x_p)))) &\to c_{ni}(c_{pn}(\mathbf{.0}(x_p)))\\
\mathsf{pred}_i(-(x_p)) &\to -(\mathsf{succ}_p(x_p))
\end{aligned}$$

Addition, subtraction and absolute value

Symbols:

$$\begin{array}{lcllll}
\mathsf{plus}_p & : & \mathsf{pos} \times \mathsf{pos} & \to & \mathsf{pos} & (\lambda xy \cdot x + y)\\
\mathsf{plus}_n & : & \mathsf{nat} \times \mathsf{nat} & \to & \mathsf{nat} & (\lambda xy \cdot x + y)\\
\mathsf{plus}_i & : & \mathsf{int} \times \mathsf{int} & \to & \mathsf{int} & (\lambda xy \cdot x + y)\\
\mathsf{min}_b & : & \mathsf{int} \times \mathsf{int} & \to & \mathsf{int} & (\lambda xy \cdot x - y)\\
\mathsf{min}_u & : & \mathsf{int} & \to & \mathsf{int} & (\lambda x \cdot -x)\\
\mathsf{abs} & : & \mathsf{int} & \to & \mathsf{nat} & (\lambda x \cdot |x|)
\end{array}$$

Auxiliary symbols:

$$\begin{array}{lcllll}
\mathsf{double} & : & \mathsf{int} & \to & \mathsf{int} & (\lambda x \cdot 2x)\\
\mathsf{min}_{pi} & : & \mathsf{pos} \times \mathsf{pos} & \to & \mathsf{int} & (\lambda xy \cdot x - y)
\end{array}$$

Rewrite rules for positive addition:

$$\begin{aligned}
\mathsf{plus}_p(1, x_p) &\to \mathsf{succ}_p(x_p)\\
\mathsf{plus}_p(\mathbf{.0}(x_p), 1) &\to \mathsf{succ}_p(\mathbf{.0}(x_p))\\
\mathsf{plus}_p(\mathbf{.1}(x_p), 1) &\to \mathbf{.0}(\mathsf{succ}_p(x_p))\\
\mathsf{plus}_p(\mathbf{.0}(x_p), \mathbf{.0}(y_p)) &\to \mathbf{.0}(\mathsf{plus}_p(x_p, y_p))\\
\mathsf{plus}_p(\mathbf{.0}(x_p), \mathbf{.1}(y_p)) &\to \mathbf{.1}(\mathsf{plus}_p(x_p, y_p))\\
\mathsf{plus}_p(\mathbf{.1}(x_p), \mathbf{.0}(y_p)) &\to \mathbf{.1}(\mathsf{plus}_p(x_p, y_p))\\
\mathsf{plus}_p(\mathbf{.1}(x_p), \mathbf{.1}(y_p)) &\to \mathbf{.0}(\mathsf{succ}_p(\mathsf{plus}_p(x_p, y_p)))
\end{aligned}$$

Rewrite rules for natural addition:

$$\begin{aligned}
\mathsf{plus}_n(0, x_n) &\to x_n\\
\mathsf{plus}_n(x_n, 0) &\to x_n\\
\mathsf{plus}_n(c_{pn}(x_p), c_{pn}(y_p)) &\to c_{pn}(\mathsf{plus}_p(x_p, y_p))
\end{aligned}$$

Rewrite rules for integer addition:

$$\begin{aligned}
\mathsf{plus}_i(c_{ni}(x_n), c_{ni}(y_n)) &\to c_{ni}(\mathsf{plus}_n(x_n, y_n)) \\
\mathsf{plus}_i(c_{ni}(0), -(x_p)) &\to -(x_p) \\
\mathsf{plus}_i(c_{ni}(c_{pn}(x_p)), -(y_p)) &\to \mathsf{min}_{pi}(x_p, y_p) \\
\mathsf{plus}_i(-(x_p), c_{ni}(0)) &\to -(x_p) \\
\mathsf{plus}_i(-(x_p), c_{ni}(c_{pn}(y_p))) &\to \mathsf{min}_{pi}(y_p, x_p) \\
\mathsf{plus}_i(-(x_p), -(y_p)) &\to -(\mathsf{plus}_p(x_p, y_p))
\end{aligned}$$

Rewrite rules for the auxiliary symbol min_{pi}:

$$\begin{aligned}
\mathsf{min}_{pi}(1, 1) &\to c_{ni}(0) \\
\mathsf{min}_{pi}(\mathbf{.0}(x_p), 1) &\to c_{ni}(c_{pn}(\mathsf{pred}_p(\mathbf{.0}(x_p)))) \\
\mathsf{min}_{pi}(\mathbf{.1}(x_p), 1) &\to c_{ni}(c_{pn}(\mathbf{.0}(x_p))) \\
\mathsf{min}_{pi}(1, \mathbf{.0}(x_p)) &\to -(\mathsf{pred}_p(\mathbf{.0}(x_p))) \\
\mathsf{min}_{pi}(1, \mathbf{.1}(x_p)) &\to -(\mathbf{.0}(x_p)) \\
\mathsf{min}_{pi}(\mathbf{.0}(x_p), \mathbf{.0}(y_p)) &\to \mathsf{double}(\mathsf{min}_{pi}(x_p, y_p)) \\
\mathsf{min}_{pi}(\mathbf{.1}(x_p), \mathbf{.0}(y_p)) &\to \mathsf{succ}_i(\mathsf{double}(\mathsf{min}_{pi}(x_p, y_p))) \\
\mathsf{min}_{pi}(\mathbf{.0}(x_p), \mathbf{.1}(y_p)) &\to \mathsf{pred}_i(\mathsf{double}(\mathsf{min}_{pi}(x_p, y_p))) \\
\mathsf{min}_{pi}(\mathbf{.1}(x_p), \mathbf{.1}(y_p)) &\to \mathsf{double}(\mathsf{min}_{pi}(x_p, y_p))
\end{aligned}$$

Rewrite rules for the auxiliary symbol double:

$$\begin{aligned}
\mathsf{double}(c_{ni}(0)) &\to c_{ni}(0) \\
\mathsf{double}(c_{ni}(c_{pn}(x_p))) &\to c_{ni}(c_{pn}(\mathbf{.0}(x_p))) \\
\mathsf{double}(-(x_p)) &\to -(\mathbf{.0}(x_p))
\end{aligned}$$

Rewrite rules for unary minus min_u:

$$\begin{aligned}
\mathsf{min}_u(c_{ni}(0)) &\to c_{ni}(0) \\
\mathsf{min}_u(c_{ni}(c_{pn}(x_p))) &\to -(x_p) \\
\mathsf{min}_u(-(x_p)) &\to c_{ni}(c_{pn}(x_p))
\end{aligned}$$

Rewrite rule for subtraction min_b:

$$\mathsf{min}_b(x_i, y_i) \to \mathsf{plus}_i(x_i, \mathsf{min}_u(y_i))$$

Rewrite rules for absolute value:

$$\begin{aligned}
\mathsf{abs}(c_{ni}(x_n)) &\to x_n \\
\mathsf{abs}(-(x_p)) &\to c_{pn}(x_p)
\end{aligned}$$

Properties of the additive rewrite system

Let $\mathbf{R}_{\mathsf{add}}$ be the rewrite system consisting of all rules until now.

Theorem 1 *The rewrite system $\mathbf{R}_{\mathsf{add}}$ has the following properties:*

1. $\mathbf{R}_{\mathsf{add}}$ *is terminating;*

2. $\mathbf{R}_{\mathsf{add}}$ *is confluent;*

3. *all reductions from a term to its normal form have the same length;*

4. *a ground term is a normal form if and only if it is a constructor term;*

Proof:

1. Termination of $\mathbf{R}_{\mathsf{add}}$ is proved by recursive path order, for instance by choosing the precedence

$$\mathsf{min}_b > \mathsf{plus}_i > \mathsf{plus}_n > \mathsf{plus}_p > \mathsf{min}_u > \mathsf{min}_{pi} > \mathsf{double} > \mathsf{abs} >$$

$$\mathsf{succ}_i > \mathsf{pred}_i > \mathsf{succ}_n > \mathsf{succ}_p > \mathsf{pred}_p > - > c_{ni} > c_{pn} > \mathbf{.1} > \mathbf{.0} > 1 > 0.$$

2. $\mathbf{R}_{\mathsf{add}}$ is confluent since it is weakly orthogonal.

3. All reductions from a term to its normal form have the same length since $\mathbf{R}_{\mathsf{add}}$ satisfies the diamond property. This hold since $\mathbf{R}_{\mathsf{add}}$ is weakly orthogonal and for every rule every occurring variable occurs both exactly once in the left hand side and exactly once in the right hand side.

4. Every constructor term is a normal form since every left hand side of a rule contains a defined symbol. Conversely for every ground term containing a defined symbol a rule is applicable.

□

The next theorem gives a linear bound on reduction lengths of terms containing only one defined symbol. To stress the subtility of this theorem we first give an example showing that this property does not hold for terms containing an arbitrary number of defined symbols. For $n \geq 0$ the term

$$(\mathsf{succ}_p\mathsf{pred}_p)^n(\mathbf{.0})^n(1)$$

consisting of $3n + 1$ symbols allows the following reduction

$$\begin{array}{lll}
(\mathsf{succ}_p\mathsf{pred}_p)^n(\mathbf{.0})^n(1) & = & (\mathsf{succ}_p\mathsf{pred}_p)^{n-1}(\mathsf{succ}_p(\mathsf{pred}_p((\mathbf{.0})^n(1)))) \\
& \to^n & (\mathsf{succ}_p\mathsf{pred}_p)^{n-1}(\mathsf{succ}_p((\mathbf{.1})^{n-1}(1))) \\
& \to^n & (\mathsf{succ}_p\mathsf{pred}_p)^{n-1}((\mathbf{.0})^n(1)) \\
& \to^{2n} & (\mathsf{succ}_p\mathsf{pred}_p)^{n-2}((\mathbf{.0})^n(1)) \\
\cdots & \cdots & \cdots \\
& \to & (\mathbf{.0})^n(1)
\end{array}$$

of $2n^2$ steps.

We write $|t|$ for the size of a term t, i.e., $|t|$ is the number of symbols (both constructor symbols and defined symbols) in t.

Theorem 2 *The length of an* $\mathbf{R}_{\mathsf{add}}$*-reduction of a term* t *containing exactly one defined symbol is linear in* $|t|$*.*

Proof: A basic idea of getting bounds on reduction lengths is the following. Define a weight function W to the natural numbers such that the weight of a term strictly decreases by every rewrite step. Then the length of a reduction starting in t is at most $W(t)$. In our case we want a linear bound, so we want $W(t)$ to be linear in $|t|$. This is achieved if we give every symbol f a weight w_f and define $W(t)$ of simply to be the sum of all weights of the symbols in t, i.e., we define inductively

$$W(f(t_1, \ldots, t_n)) = w_f + \sum_{i=1}^{n} W(t_i).$$

Unfortunately it is not possible to define a uniform weight for all symbols, since in the above example we saw a linear size term having a reduction of quadratic length. The reason for this quadratic behavior was in the mixture of successor and predecessor symbols. Therefore we will prove linear bounds on reduction lengths by the weight approach for reductions in which predecessor symbols do not occur and also for reductions in which successor symbols do not occur. Finally we will prove that every reduction of the shape that we consider can be polished in such a way that after a linear number of steps it is in one of these two classes.

> **Claim 1.** Let t be a term composed from constructors and one or more from the defined symbols succ_p, succ_n, plus_p, plus_n, min_u, abs and double. Then the length of an $\mathbf{R}_{\mathsf{add}}$-reduction of t is linear in $|t|$.

The proof of this claim is easily given by the weight approach: define

$$w_0 = w_1 = w_{\mathbf{.0}} = w_{-} = w_{c_{ni}} = w_{c_{pn}} = 1 \text{ and } w_{\mathbf{.1}} = w_f = 2$$

for all defined symbols f. Then it is easily seen from the rewrite rules that in every step in the $\mathbf{R}_{\mathsf{add}}$-reduction of t only symbols occur from the given set, and the weight strictly decreases. This gives a linear reduction length, more precisely, the redcution length of t is at most $2|t|$.

> **Claim 2.** Let t be a term composed from constructors and one or more from the defined symbols pred_p, min_u, abs and double. Then the length of an $\mathbf{R}_{\mathsf{add}}$-reduction of t is linear in $|t|$.

The proof of this claim is again given by the weight approach: now define

$$w_0 = w_1 = w_{\mathbf{.1}} = w_{-} = w_{c_{ni}} = w_{c_{pn}} = 1 \text{ and } w_{\mathbf{.0}} = w_f = 2$$

for all defined symbols f. Again it is easily seen from the rewrite rules that in every step in the $\mathbf{R}_{\mathsf{add}}$-reduction of t only symbols occur from the given set, and

the weight strictly decreases in every rewrite step, hence again giving a linear reduction length.

Note that for Claim 1 it is essential that the weight of $\mathbf{.0}$ is smaller than the weight of $\mathbf{.1}$, while the reverse holds for Claim 2.

It remains to prove the theorem for the case that the single defined symbol in the term t is succ_i, pred_i, plus_i, min_{pi} or min_b. By substitution of variables by the constants 0 and 1, and removing everything above the defined symbols, without loss of generality we may assume that t is a ground term of which the root is one of the given defined symbols.a

If the root of t is succ_i then either $t = \mathsf{succ}_i(c_{ni}(u))$ or $t = \mathsf{succ}_i(-(u))$ for some constructor term u. In the first case after one reduction step t is transformed to $\mathsf{succ}_n(u)$, on which Claim 1 can be applied; in the second case after one reduction step t is transformed to a term on which Claim 2 can be applied, in both cases yielding a linear reduction length.

If the root of t is pred_i then either $t = \mathsf{pred}_i(c_{ni}(u))$ or $t = \mathsf{pred}_i(-(u))$ for some constructor term u, again yielding a term after one reduction step on which Claim 1 or Claim 2 can be applied.

Now assume that the root of t is min_{pi}. We will construct a particular reduction of t. First apply the last four rules for min_{pi} as long as possible, yielding a reduction $t \to^* C[\mathsf{min}_{pi}(t_1, t_2)]$ in which the context C consists of the symbols succ_i, pred_i and double, and in which $t_1 = 1$ or $t_2 = 2$ (or both). Next reduce $\mathsf{min}_{pi}(t_1, t_2)$ to its normal form u; after one step this takes at most a linear number of reduction steps due to Claim 2. Next apply the three rules for double, the four rules for succ_i and the five rules for pred_i as long as possible. Since in every step here at least one of the symbols double, succ_i or pred_i disappears, this takes a linear number of steps. Since the argument of every symbol succ_i and pred_i has always double as its root, due to the way C was constructed, and double applied on a constructor term always rewrites to a term of the shape $c_{ni}(0)$, $c_{ni}(c_{pn}(\mathbf{.0}(\cdots)))$ or $-(\mathbf{.0}(\cdots))$, always a rule for succ_i or pred_i is applicable. Hence this reduction ends in a term of the shape $c_{ni}(0)$ or $D[v]$ where D is either $c_{ni}(c_{pn}(\Box))$ or $-(\Box)$, and v is a ground term constructed from 1, $\mathbf{.0}$, $\mathbf{.1}$, succ_p and pred_p. Since every succ_p-symbol in v originates from a succ_i-symbol or a pred_i-symbol in C where its argument has double as its root, in v every succ_p-symbol has $\mathbf{.0}$ as its root. Now each of these succ_p-symbols can be removed by applying the rule $\mathsf{succ}_p(\mathbf{.0}(x_p)) \to \mathbf{.1}(x_p)$, yielding a term on which Claim 2 can be applied, yielding a normal form of $D[v]$ in a linear number of steps. In all cases we found a reduction of t to its normal form of which the length is linear in $|t|$. Due to part 3 of Theorem 1 this holds for every reduction of t.

Next assume that the root of t is plus_i. From the rules for plus_i it is easily seen that after one reduction step is transformed either in a term in which Claim 1 is applicable, or it is of the shape $\mathsf{min}_{pi}(t_1, t_2)$. In both cases we already obtained a linear reduction length.

Finally assume that the root of t is min_b. Then after one reduction step the term reads $\mathsf{plus}_i(t_1, \mathsf{min}_u(t_2))$. Due to Claim 1 $\mathsf{min}_u(t_2)$ reduces to a normal form u in a linear number of steps, hence $\mathsf{plus}_i(t_1, \mathsf{min}_u(t_2))$ reduces to $\mathsf{plus}_i(t_1, u)$, which reduces to normal form in a linear number of steps as we saw above. Hence we have a reduction from t to normal form in a linear number of steps; due to part 3 of Theorem 1 this holds for every reduction of t. □

Multiplication

Symbols:

$$\begin{array}{rcrcll} \mathsf{mult}_p & : & \mathsf{pos} \times \mathsf{pos} & \to & \mathsf{pos} & (\lambda xy \cdot x * y) \\ \mathsf{mult}_n & : & \mathsf{nat} \times \mathsf{nat} & \to & \mathsf{nat} & (\lambda xy \cdot x * y) \\ \mathsf{mult}_i & : & \mathsf{int} \times \mathsf{int} & \to & \mathsf{int} & (\lambda xy \cdot x * y) \end{array}$$

Rewrite rules for positive multiplication:

$$\begin{array}{rcl} \mathsf{mult}_p(1, x_p) & \to & x_p \\ \mathsf{mult}_p(\mathbf{.0}(x_p), y_p) & \to & \mathbf{.0}(\mathsf{mult}_p(x_p, y_p)) \\ \mathsf{mult}_p(\mathbf{.1}(x_p), y_p) & \to & \mathsf{plus}_p(\mathbf{.0}(\mathsf{mult}_p(x_p, y_p)), y_p) \end{array}$$

Rewrite rules for natural multiplication:

$$\begin{array}{rcl} \mathsf{mult}_n(0, x_n) & \to & 0 \\ \mathsf{mult}_n(x_n, 0) & \to & 0 \\ \mathsf{mult}_n(c_{pn}(x_p), c_{pn}(y_p)) & \to & c_{pn}(\mathsf{mult}_p(x_p, y_p)) \end{array}$$

Rewrite rules for integer multiplication:

$$\begin{array}{rcl} \mathsf{mult}_i(c_{ni}(x_n), c_{ni}(y_n)) & \to & c_{ni}(\mathsf{mult}_n(x_n, y_n)) \\ \mathsf{mult}_i(c_{ni}(0), -(y_p)) & \to & c_{ni}(0) \\ \mathsf{mult}_i(c_{ni}(c_{pn}(x_p)), -(y_p)) & \to & -(\mathsf{mult}_p(x_p, y_p)) \\ \mathsf{mult}_i(-(x_p), c_{ni}(0)) & \to & c_{ni}(0) \\ \mathsf{mult}_i(-(x_p), c_{ni}(c_{pn}(y_p))) & \to & -(\mathsf{mult}_p(x_p, y_p)) \\ \mathsf{mult}_i(-(x_p), -(y_p)) & \to & c_{ni}(c_{pn}(\mathsf{mult}_p(x_p, y_p))) \end{array}$$

Let $\mathbf{R}_{\mathsf{mult}}$ be the rewrite system consisting of all rules until now. We want to analyze which of the properties of Theorem 1 hold for $\mathbf{R}_{\mathsf{mult}}$, and what can be said about the lengths of reductions. It turns out that the third property of Theorem 1 does not hold in general. For instance, the term $\mathsf{mult}_p(\mathbf{.1}(1), \mathsf{succ}_p(1))$ reduces in two steps to $\mathsf{plus}_p(\mathbf{.0}(\mathsf{mult}_p(1, \mathbf{.0}(1)), \mathbf{.0}(1))$ if first $\mathsf{succ}_p(1)$ is reduced to $\mathbf{.0}(1)$, but reduces in three steps to the same term if first the rule $\mathsf{mult}_p(\mathbf{.1}(x_p), y_p) \to \mathsf{plus}_p(\mathbf{.0}(\mathsf{mult}_p(x_p, y_p)), y_p)$ is applied. This gives rise to reductions to normal form of different lengths.

In real computations however normal forms are computed of terms in which the arguments of defined symbols are constructor terms. For such terms we will show that this undesired behavior does not occur.

We define a *multiplication term* to be a term in which there is at most one occurrence of the symbols mult_p, mult_n and mult_i, and if it occurs then its arguments are constructor terms.

For any term t we write $N(t)$ for the normal form of t w.r.t. $\mathbf{R}_{\mathsf{mult}}$.

Theorem 3 *The rewrite system $\mathbf{R}_{\mathsf{mult}}$ has the following properties:*

1. *$\mathbf{R}_{\mathsf{mult}}$ is terminating and confluent, and a ground term is a normal form if and only if it is a constructor term;*
2. *$|t| \geq |N(t)|$ for every ground term t;*
3. *all reductions from a multiplication term to its normal form have the same length;*
4. *the length of a reduction from a multiplication term to its normal form is at most quadratic in the size of the term.*

Proof: Part 1 is similar to parts 1, 2 an 4 of Theorem 1, for termination by recursive path order the precedence has to be extended by

$$\mathsf{mult}_i > \mathsf{mult}_n > \mathsf{mult}_p > \mathsf{plus}_p.$$

For part 2 we apply induction on $|t|$. If t is a constant then the property holds since t is a normal form. If $t = f(u)$ for any unary symbol f then $|u| \geq |N(u)|$ by the induction hypothesis, and $N(f(u)) = N(f(N(u)))$ by uniqueness of normal form. For all choices of f we check that $|v| + 1 \geq |N(f(v)|$ for every ground constructor term v by counting the number of digits of the value of $f(v)$. For instance for $f = \mathsf{succ}_p$ then $|v|+1 \geq |N(f(v)|$ follows from $2x \geq x+1$ for positive values of x. Since $N(u)$ is a ground constructor term we obtain

$$|t| = |u| + 1 \geq |N(u)| + 1 \geq |N(f(N(u))| = |N(f(u))| = |N(t)|.$$

Similarly if $t = f(t_1, t_2)$ for any binary symbol f we obtain

$$|t| = |t_1| + |t_2| + 1 \geq |N(t_1)| + |N(t_2)| + 1$$

$$\geq |N(f(N(t_1), N(t_2))| = |N(f(t_1, t_2))| = |N(t)|;$$

here for all binary symbols f we have to check that $|v_1|+|v_2|+1 \geq |N(f(v_1, v_2)|$ for all ground constructor terms v_1, v_2 by counting the number of digits of the values, for instance for $f = \mathsf{mult}_p$ this follows from the observation that the positive ground constructor term v_i represents a number n_i satisfying $2^{|v_i|-1} \leq n_i < 2^{|v_i|}$ for $i = 1, 2$, and $n_1 * n_2 < 2^{|v_1|+|v_2|}$, hence $n_1 * n_2$ will be represented by a ground constructor term of size at most $|v_1| + |v_2|$. This concludes the proof of part 2.

For part 3 we apply induction on the length of the reduction to normal form. If none of the symbols mult_p, mult_n or mult_i occurs then part 3 of Theorem 1

applies. In the remaining case let $t = C[m(t_1, t_2)]$ be a multiplication term in which m is one of the symbols mult_p, mult_n or mult_i, and assume that $t \to u_i \to^* n$ are two reductions from t to its normal form n for $i = 1, 2$. Due to the shape of the rules we conclude that u_1 and u_2 are multiplication terms. By the induction hypothesis we may assume that every reduction from u_i to n has length n_i form $i = 1, 2$; we have to prove that $n_1 = n_2$. Since t_i is a constructor term and hence a normal form, we have either $u_i = D_i[m(t_1, t_2)]$ for a context D_i satisfying $C[x] \to D_i[x]$, or $u_i = C[v_i]$ for $m(t_1, t_2) \to v_i$, for $i = 1, 2$. We distinguish three cases:

1. Let $u_i = D_i[m(t_1, t_2)]$ and $C[x] \to D_i[x]$ for $i = 1, 2$. Then a context E exists for which $D_i[x] \to E[x]$ for $i = 1, 2$, since no multiplication symbol occurs in C and for every rule for any other defined symbol every occurring variable occurs exactly once in both the left hand side and the right hand side. Let $w = E[m(t_1, t_2)]$, then $u_i \to w$ for $i = 1, 2$, hence $n_1 = 1 + k = n_2$, where k is the length of a reduction from w to n.

2. Let $u_i = D_i[m(t_1, t_2)]$ and $C[x] \to D_i[x]$ and $u_j = C[v_j]$ for $m(t_1, t_2) \to v_j$ for $i \neq j$. Let $w = D_i[v_j]$, then $u_i \to w$ and $u_j \to w$, hence $n_1 = 1 + k = n_2$, where k is the length of a reduction from w to n.

3. Let $u_i = C[v_i]$ for $m(t_1, t_2) \to v_i$ for $i = 1, 2$. Then $v_1 = v_2$ since only one rule is applicable on $m(t_1, t_2)$. Hence $u_1 = C[v_1] = C[v_2] = u_2$.

In all cases we proved that $n_1 = n_2$, concluding the proof of part 3.

For part 4 we define $d(t)$ to be the number of defined symbols in a term t, and we choose c to be a constant such that every $\mathbf{R}_{\mathsf{add}}$-reduction of a term t containing exactly one defined symbol has length at most $c|t|$; this exists due to Theorem 2. We will prove by induction on the reduction length that for every ground multiplication term t there is a reduction to normal form of length at most

$$c * d(t) * |t| + 2cM^2$$

where $M = 0$ if none of the symbols mult_p, mult_n or mult_i occurs in t, and $M = |m(t_1, t_2)|$ if $t = C[m(t_1, t_2)]$ where m is one of the symbols mult_p, mult_n or mult_i. Clearly this bound is quadratic in $|t|$. By substitution of variables by constants, and observing that reductions to normal form then never will be shorter, we see that the required property for general terms follows from this property for ground terms. By part 3 we conclude that the property holds for all reductions of t to normal form if it has been proven to hold for a single one.

First consider the case that none of the symbols mult_p, mult_n or mult_i occurs in t. If t contains no defined symbol then t is a normal form and the property trivially holds; if t contains a defined symbol then it can be written as $t = C[u]$ such that u contains exactly one defined symbol. Then there is a reduction from u to $N(u)$ of at most $c|u|$ steps, giving rise to a reduction from $t = C[u]$ to

$C[N(u)]$ of the same length. Now $|u| \geq |N(u)|$ by part 2, hence $|t| \geq |C[N(u)]|$. Applying the hypothesis on $C[N(u)]$ yields a reduction to normal form of length at most

$$c * d(C[N(u)]) * |C[N(u)]| = c * (d(t) - 1) * |C[N(u)]| \leq c * (d(t) - 1) * |t|,$$

hence the total length of the reduction to normal form of t is at most

$$c|u| + c * (d(t) - 1) * |t| \leq c|t| + c * (d(t) - 1) * |t| = c * d(t) * |t|,$$

which we had to prove.

In the remaining case we have $t = C[m(t_1, t_2)]$ where m is one of the symbols mult_p, mult_n or mult_i, and t_1, t_2 are constructor terms. We concentrate on the case that $m = \mathsf{mult}_p$ and $t_1 = \mathbf{.1}(u)$, for all other cases the proof is similar but much simpler, and left to the reader. Write $v = \mathsf{plus}_p(\mathbf{.0}(\mathsf{mult}_p(u, t_2), t_2)$. Now $t = C[\mathsf{mult}_p(\mathbf{.1}(u), t_2)]$ rewrites in one step to $C[v]$. Applying the induction hypothesis on v yields a reduction from v to $N(v)$ of at most $c * 2 * |v| + 2c(M-1)^2$ steps. Since $|v| \leq 2M - 1$ this number is at most $2cM^2$. This gives rise to a reduction of $C[v]$ to $C[N(v)]$ of at most $2cM^2$ steps. Since $N(v) = N(\mathsf{mult}_p(\mathbf{.1}(u), t_2))$ we obtain from part 2 that $|C[N(v)]| \leq |C[\mathsf{mult}_p(\mathbf{.1}(u), t_2)| = |t|$. Hence applying the induction hypothesis on $C[N(v)]$ yields a reduction from $C[N(v)]$ to normal form of length at most $c * d(C[N(v)) * |C[N(v)| \leq c * (d(t) - 1) * |t|$. Combining all reduction steps yields a reduction form t to normal form of length at most

$$1 + 2cM^2 + c * (d(t) - 1) * |t| \leq c * d(t) * |t| + 2cM^2,$$

concluding the proof of part 4. □

Now we show that some conditions in Theorem 3 are essential. In part 2 it is essential to restrict to ground terms. For instance, the normal form of the open term $\mathsf{mult}_p(\mathbf{.1}(x_p), y_p)$ is the bigger term $\mathsf{plus}_p(\mathbf{.0}(\mathsf{mult}_p(x_p, y_p)), y_p)$.

We already gave an example showing that for part 3 it is essential to restrict to multiplication terms. The same holds for part 4. For instance, for $n \geq 0$ the term

$$\mathsf{mult}_p((\mathbf{.1})^n(1), (\mathsf{succ}_p\mathsf{pred}_p)^n(\mathbf{.0})^n(1))$$

reduces by n applications of the rule $\mathsf{mult}_p(\mathbf{.1}(x_p), y_p) \to \mathsf{plus}_p(\mathbf{.0}(\mathsf{mult}_p(x_p, y_p)), y_p)$ to a term having $n + 1$ occurrences of the subterm $(\mathsf{succ}_p\mathsf{pred}_p)^n(\mathbf{.0})^n(1)$. Since this subterm admits a reduction of quadratic length, the original term admits a reduction of cubic length.

Exponentiation

Symbols:

$$\begin{array}{lcllll} \mathsf{pow}_p & : & \mathsf{pos} \times \mathsf{pos} & \to & \mathsf{pos} & (\lambda xy \cdot x^y) \\ \mathsf{pow}_n & : & \mathsf{nat} \times \mathsf{pos} & \to & \mathsf{nat} & (\lambda xy \cdot x^y) \\ \mathsf{pow}_i & : & \mathsf{int} \times \mathsf{pos} & \to & \mathsf{int} & (\lambda xy \cdot x^y) \end{array}$$

Rewrite rules for positive exponentiation:

$$\begin{aligned}
\mathsf{pow}_p(x_p, 1) &\rightarrow x_p \\
\mathsf{pow}_p(x_p, \mathbf{.0}(y_p)) &\rightarrow \mathsf{pow}_p(\mathsf{mult}_p(x_p, x_p), y_p) \\
\mathsf{pow}_p(x_p, \mathbf{.1}(y_p)) &\rightarrow \mathsf{mult}_p(x_p, \mathsf{pow}_p(\mathsf{mult}_p(x_p, x_p), y_p))
\end{aligned}$$

Rewrite rules for natural exponentiation:

$$\begin{aligned}
\mathsf{pow}_n(0, x_p) &\rightarrow 0 \\
\mathsf{pow}_n(c_{pn}(x_p), y_p) &\rightarrow c_{pn}(\mathsf{pow}_p(x_p, y_p))
\end{aligned}$$

Rewrite rules for integer exponentiation:

$$\begin{aligned}
\mathsf{pow}_i(c_{ni}(x_n), y_p) &\rightarrow c_{ni}(\mathsf{pow}_n(x_n, y_p)) \\
\mathsf{pow}_i(-(x_p), 1) &\rightarrow -(x_p) \\
\mathsf{pow}_i(-(x_p), \mathbf{.0}(y_p)) &\rightarrow c_{ni}(c_{pn}(\mathsf{pow}_p(x_p, \mathbf{.0}(y_p)))) \\
\mathsf{pow}_i(-(x_p), \mathbf{.1}(y_p)) &\rightarrow -(\mathsf{pow}_p(x_p, \mathbf{.1}(y_p)))
\end{aligned}$$

Boolean operations

We will use $x_b, y_b, \ldots$ for variables of type bool.

Symbols:

$$\begin{array}{lcll}
\mathsf{not} & : & \mathsf{bool} & \rightarrow \mathsf{bool} \\
\mathsf{and} & : & \mathsf{bool} \times \mathsf{bool} & \rightarrow \mathsf{bool} \\
\mathsf{or} & : & \mathsf{bool} \times \mathsf{bool} & \rightarrow \mathsf{bool} \\
\mathsf{eq}_p & : & \mathsf{pos} \times \mathsf{pos} & \rightarrow \mathsf{bool} \\
\mathsf{eq}_n & : & \mathsf{nat} \times \mathsf{nat} & \rightarrow \mathsf{bool} \\
\mathsf{eq}_i & : & \mathsf{int} \times \mathsf{int} & \rightarrow \mathsf{bool} \\
\mathsf{gr}_p & : & \mathsf{pos} \times \mathsf{pos} & \rightarrow \mathsf{bool} \\
\mathsf{gr}_n & : & \mathsf{nat} \times \mathsf{nat} & \rightarrow \mathsf{bool} \\
\mathsf{gr}_i & : & \mathsf{int} \times \mathsf{int} & \rightarrow \mathsf{bool} \\
\mathsf{if}_p & : & \mathsf{bool} \times \mathsf{pos} \times \mathsf{pos} & \rightarrow \mathsf{pos} \\
\mathsf{if}_n & : & \mathsf{bool} \times \mathsf{nat} \times \mathsf{nat} & \rightarrow \mathsf{nat} \\
\mathsf{if}_i & : & \mathsf{bool} \times \mathsf{int} \times \mathsf{int} & \rightarrow \mathsf{int}
\end{array}$$

Rewrite rules for basic boolean operations:

$$\begin{aligned}
\mathsf{not}(\mathbf{F}) &\rightarrow \mathbf{T} \\
\mathsf{not}(\mathbf{T}) &\rightarrow \mathbf{F} \\
\mathsf{and}(x_b, \mathbf{T}) &\rightarrow x_b \\
\mathsf{and}(x_b, \mathbf{F}) &\rightarrow \mathbf{F} \\
\mathsf{and}(\mathbf{T}, x_b) &\rightarrow x_b \\
\mathsf{and}(\mathbf{F}, x_b) &\rightarrow \mathbf{F} \\
\mathsf{or}(x_b, \mathbf{T}) &\rightarrow \mathbf{T} \\
\mathsf{or}(x_b, \mathbf{F}) &\rightarrow x_b \\
\mathsf{or}(\mathbf{T}, x_b) &\rightarrow \mathbf{T} \\
\mathsf{or}(\mathbf{F}, x_b) &\rightarrow x_b
\end{aligned}$$

Rewrite rules for positive equality:

$$\begin{aligned}
\mathsf{eq}_p(1,1) &\to \mathbf{T}\\
\mathsf{eq}_p(1,\mathbf{.0}(x_p)) &\to \mathbf{F}\\
\mathsf{eq}_p(1,\mathbf{.1}(x_p)) &\to \mathbf{F}\\
\mathsf{eq}_p(\mathbf{.0}(x_p),1) &\to \mathbf{F}\\
\mathsf{eq}_p(\mathbf{.0}(x_p),\mathbf{.0}(y_p)) &\to \mathsf{eq}_p(x_p,y_p)\\
\mathsf{eq}_p(\mathbf{.0}(x_p),\mathbf{.1}(y_p)) &\to \mathbf{F}\\
\mathsf{eq}_p(\mathbf{.1}(x_p),1) &\to \mathbf{F}\\
\mathsf{eq}_p(\mathbf{.1}(x_p),\mathbf{.0}(y_p)) &\to \mathbf{F}\\
\mathsf{eq}_p(\mathbf{.1}(x_p),\mathbf{.1}(y_p)) &\to \mathsf{eq}_p(x_p,y_p)
\end{aligned}$$

Rewrite rules for natural equality:

$$\begin{aligned}
\mathsf{eq}_n(0,0) &\to \mathbf{T}\\
\mathsf{eq}_n(0,c_{pn}(x_p)) &\to \mathbf{F}\\
\mathsf{eq}_n(c_{pn}(x_p),0) &\to \mathbf{F}\\
\mathsf{eq}_n(c_{pn}(x_p),c_{pn}(y_p)) &\to \mathsf{eq}_p(x_p,y_p)
\end{aligned}$$

Rewrite rules for integer equality:

$$\begin{aligned}
\mathsf{eq}_i(c_{ni}(x_n),c_{ni}(y_n)) &\to \mathsf{eq}_n(x_n,y_n)\\
\mathsf{eq}_i(c_{ni}(x_n),-(y_p)) &\to \mathbf{F}\\
\mathsf{eq}_i(-(x_p),c_{ni}(y_n)) &\to \mathbf{F}\\
\mathsf{eq}_i(-(x_p),-(y_p)) &\to \mathsf{eq}_p(x_p,y_p)
\end{aligned}$$

Rewrite rules for positive inequality:

$$\begin{aligned}
\mathsf{gr}_p(1,1) &\to \mathbf{F}\\
\mathsf{gr}_p(1,\mathbf{.0}(x_p)) &\to \mathbf{F}\\
\mathsf{gr}_p(1,\mathbf{.1}(x_p)) &\to \mathbf{F}\\
\mathsf{gr}_p(\mathbf{.0}(x_p),1) &\to \mathbf{T}\\
\mathsf{gr}_p(\mathbf{.0}(x_p),\mathbf{.0}(y_p)) &\to \mathsf{gr}_p(x_p,y_p)\\
\mathsf{gr}_p(\mathbf{.0}(x_p),\mathbf{.1}(y_p)) &\to \mathsf{gr}_p(x_p,y_p)\\
\mathsf{gr}_p(\mathbf{.1}(x_p),1) &\to \mathbf{T}\\
\mathsf{gr}_p(\mathbf{.1}(x_p),\mathbf{.0}(y_p)) &\to \mathsf{not}(\mathsf{gr}_p(y_p,x_p))\\
\mathsf{gr}_p(\mathbf{.1}(x_p),\mathbf{.1}(y_p)) &\to \mathsf{gr}_p(x_p,y_p)
\end{aligned}$$

Rewrite rules for natural inequality:

$$\begin{aligned}
\mathsf{gr}_n(0,0) &\to \mathbf{F}\\
\mathsf{gr}_n(0,c_{pn}(x_p)) &\to \mathbf{F}\\
\mathsf{gr}_n(c_{pn}(x_p),0) &\to \mathbf{T}\\
\mathsf{gr}_n(c_{pn}(x_p),c_{pn}(y_p)) &\to \mathsf{gr}_p(x_p,y_p)
\end{aligned}$$

Rewrite rules for integer inequality:

$$\begin{array}{rcl} \mathsf{gr}_i(c_{ni}(x_n), c_{ni}(y_n)) & \to & \mathsf{gr}_n(x_n, y_n) \\ \mathsf{gr}_i(c_{ni}(x_n), -(y_p)) & \to & \mathbf{T} \\ \mathsf{gr}_i(-(x_p), c_{ni}(y_n)) & \to & \mathbf{F} \\ \mathsf{gr}_i(-(x_p), -(y_p)) & \to & \mathsf{gr}_p(y_p, x_p) \end{array}$$

Rewrite rules for selection:

$$\begin{array}{rcl} \mathsf{if}_p(\mathbf{T}, x_p, y_p) & \to & x_p \\ \mathsf{if}_p(\mathbf{F}, x_p, y_p) & \to & y_p \\ \mathsf{if}_n(\mathbf{T}, x_n, y_n) & \to & x_n \\ \mathsf{if}_n(\mathbf{F}, x_n, y_n) & \to & y_n \\ \mathsf{if}_i(\mathbf{T}, x_i, y_i) & \to & x_i \\ \mathsf{if}_i(\mathbf{F}, x_i, y_i) & \to & y_i \end{array}$$

In order to improve readability we write the three arguments of these operators on separate lines if these arguments are big terms.

Div and mod

Symbols:

$$\begin{array}{lcllll} \mathsf{mod} & : & \mathsf{nat} \times \mathsf{pos} & \to & \mathsf{nat} & (\lambda xy \cdot x \mathsf{mod} y) \\ \mathsf{div} & : & \mathsf{nat} \times \mathsf{pos} & \to & \mathsf{nat} & (\lambda xy \cdot x \mathsf{div} y) \end{array}$$

Auxiliary symbols:

$$\begin{array}{lcll} f & : & \mathsf{pos} \times \mathsf{pos} & \to \mathsf{pos} \\ g & : & \mathsf{pos} \times \mathsf{pos} \times \mathsf{pos} & \to \mathsf{pos} \end{array}$$

The intended meaning of f and g is the following. For $x \geq y > 0$ we have $f(x, y) = 2^i * y$ for the unique positive integer i satisfying

$$2^i * y \leq x < 2^{i+1} * y.$$

For $x \geq y > 0, z > 0$ we have $g(x, y, z) = 2^i * z$ for the same value i, i.e., $g(x, y, z)$ is defined by

$$g(x, y, z) * y = f(x, y) * z.$$

Rewrite rules:

$$\begin{array}{rcl}
\mathsf{mod}(0, y_p) & \to & 0 \\
\mathsf{mod}(c_{pn}(x_p), y_p) & \to & \mathsf{if}_n(\mathsf{gr}_p(y_p, x_p), \\
 & & \quad c_{pn}(x_p), \\
 & & \quad \mathsf{mod}(\mathsf{abs}(\mathsf{min}_{pi}(x_p, f(x_p, y_p))), y_p)) \\
f(x_p, y_p) & \to & \mathsf{if}_p(\mathsf{gr}_p(\mathbf{.0}(y_p), x_p), \\
 & & \quad y_p, \\
 & & \quad f(x_p, \mathbf{.0}(y_p))) \\
\mathsf{div}(0, y_p) & \to & 0 \\
\mathsf{div}(c_{pn}(x_p), y_p) & \to & \mathsf{if}_n(\mathsf{gr}_p(y_p, x_p), \\
 & & \quad 0, \\
 & & \quad \mathsf{plus}_n(c_{pn}(g(x_p, y_p, 1)), \mathsf{div}(\mathsf{abs}(\mathsf{min}_{pi}(x_p, f(x_p, y_p))), y_p))) \\
g(x_p, y_p, z_p) & \to & \mathsf{if}_p(\mathsf{gr}_p(\mathbf{.0}(y_p), x_p), \\
 & & \quad z_p, \\
 & & \quad g(x_p, \mathbf{.0}(y_p), \mathbf{.0}(z_p)))
\end{array}$$

Note that these rewrite rules are not terminating in the general sense since we have a rule of the shape $f(x_p, y_p) \to C[f(x_p, y_p)\sigma]$ for a context C and a substitution σ. However, if we disallow rewriting inside the second or third argument of an if-symbol (this is a particular case of context-sensitive rewriting), then we may expect that termination can be proved, or even better a bound on the length of reductions.

An approach to achieve termination without restricting to context-sensitive rewriting is the following. For every if-symbol in the above rewrite rules we introduce a fresh symbol IF_i, for a number i, and replace the rule of the shape

$$t \to \mathsf{if}(C, u, v)$$

for terms t, u, v and boolean term C by the three rules

$$\begin{array}{rcl}
t & \to & \mathsf{IF}_i(C, x_1, \ldots, x_n) \\
\mathsf{IF}_i(\mathbf{T}, x_1, \ldots, x_n) & \to & u \\
\mathsf{IF}_i(\mathbf{F}, x_1, \ldots, x_n) & \to & v
\end{array}$$

where $x_1, \ldots, x_n$ are the variables occurring in u or v. Applying this approach transforms the above rules for mod and div to the following rules, using four auxiliary symbols.

Auxiliary symbols:

$$\begin{array}{lcll}
\mathsf{IF}_1 & : & \mathsf{bool} \times \mathsf{pos} \times \mathsf{pos} & \to \mathsf{nat} \\
\mathsf{IF}_2 & : & \mathsf{bool} \times \mathsf{pos} \times \mathsf{pos} & \to \mathsf{pos} \\
\mathsf{IF}_3 & : & \mathsf{bool} \times \mathsf{pos} \times \mathsf{pos} & \to \mathsf{nat} \\
\mathsf{IF}_4 & : & \mathsf{bool} \times \mathsf{pos} \times \mathsf{pos} \times \mathsf{pos} & \to \mathsf{pos}
\end{array}$$

Rewrite rules:

$$
\begin{aligned}
\mathsf{mod}(0, y_p) &\to 0 \\
\mathsf{mod}(c_{pn}(x_p), y_p) &\to \mathsf{IF}_1(\mathsf{gr}_p(y_p, x_p), x_p, y_p) \\
\mathsf{IF}_1(\mathbf{T}, x_p, y_p) &\to c_{pn}(x_p) \\
\mathsf{IF}_1(\mathbf{F}, x_p, y_p) &\to \mathsf{mod}(\mathsf{abs}(\mathsf{min}_{pi}(x_p, f(x_p, y_p))), y_p) \\
f(x_p, y_p) &\to \mathsf{IF}_2(\mathsf{gr}_p(\mathbf{.0}(y_p), x_p), x_p, y_p) \\
\mathsf{IF}_2(\mathbf{T}, x_p, y_p) &\to y_p \\
\mathsf{IF}_2(\mathbf{F}, x_p, y_p) &\to f(x_p, \mathbf{.0}(y_p)) \\
\mathsf{div}(0, y_p) &\to 0 \\
\mathsf{div}(c_{pn}(x_p), y_p) &\to \mathsf{IF}_3(\mathsf{gr}_p(y_p, x_p), x_p, y_p) \\
\mathsf{IF}_3(\mathbf{T}, x_p, y_p) &\to 0 \\
\mathsf{IF}_3(\mathbf{F}, x_p, y_p) &\to \mathsf{plus}_n(c_{pn}(g(x_p, y_p, 1)), \mathsf{div}(\mathsf{abs}(\mathsf{min}_{pi}(x_p, f(x_p, y_p))), y_p)) \\
g(x_p, y_p, z_p) &\to \mathsf{IF}_4(\mathsf{gr}_p(\mathbf{.0}(y_p), x_p), x_p, y_p, z_p) \\
\mathsf{IF}_4(\mathbf{T}, x_p, y_p, z_p) &\to z_p \\
\mathsf{IF}_4(\mathbf{F}, x_p, y_p, z_p) &\to g(x_p, \mathbf{.0}(y_p), \mathbf{.0}(z_p))
\end{aligned}
$$