Rewrite systems for natural, integral, and rational arithmetic*

Evelyne Contejean, Claude Marché and Landy Rabehasaina

LRI, CNRS URA 410
Bât. 490, Université Paris-Sud, Centre d'Orsay
91405 Orsay Cedex, France
Phone:+33 1 69 15 64 85, Fax: +33 1 69 15 65 86
Email: {contejea,marche}@lri.fr

Abstract. We give algebraic presentations of the sets of natural numbers, integers, and rational numbers by convergent rewrite systems which moreover allow efficient computations of arithmetical expressions. We then use such systems in the general normalised completion algorithm, in order to compute Gröbner bases of polynomial ideals over $\mathbb{Q}$.

1 Introduction

1.1 Motivations

In the literature one can find several papers [10, 13, 24] which present rewrite systems for evaluation of arithmetical expressions on natural numbers, integers or even real numbers with finite expansion in a given base. The motivation of these papers was to have useful rewrite systems in a context of algebraic specifications and automated theorem proving. Thus, the goal was to obtain *convergent* (modulo associativity and commutativity of addition and multiplication) rewrite systems, to have existence and uniqueness of normal forms; and also *efficient* rewrite systems, in the sense that the number of reductions to get the normal form of an expression has to be as small as possible w.r.t the size of the expression. Practically speaking, this second requirement amounts to have a binary representation of numbers (or in any base $b \geq 2$) and not the classical Peano representation by the constant 0 and the successor function.

Our motivation is somewhat different : we are interested in computing *Gröbner bases of polynomial ideals* over the field $\mathbb{Q}$ of rational numbers. It has been remarked by Loos [20] and Buchberger [6] that Buchberger's algorithm for computing Gröbner bases [4, 5] proceeds in a very similar way as the equational completion algorithm of Knuth and Bendix [18]. Since that time, many authors have tried to "unify" these algorithms [1, 7, 8, 16, 19, 22], that is to view them as two instances of a general completion algorithm. In this paper, we are interested in the work of Marché [21] where he presented the *S-normalised completion*

* This research was supported in part by the EWG CCL II, the HCM Network CONSOLE, and the "GDR de programmation du CNRS".

algorithm — where S is an arbitrary convergent rewrite system — which specialises to Knuth-Bendix completion when choosing $S = \emptyset$ and to a Gröbner bases computation algorithm of integral polynomial ideals when choosing for S a convergent rewrite system for commutative rings. To use this completion algorithm for computing Gröbner bases over $\mathbb{Q}$, one has to use for S a convergent system for rational numbers, and that was left as an open problem in [21]. That's the reason why we were interested in the works mentioned at the beginning.

Thus, our motivation was to find a convergent and efficient rewrite system for rational numbers. While achieving this goal we also obtain improvements to the rewrite systems for natural numbers and integers so we are going first to present these improvements.

1.2 State of the Art

We briefly give the state of the art on this subject, for more detail we refer to [23].

As far as we know, the first paper concerning efficient rewrite presentations of arithmetic is the one of Cohen and Watson [10], where they gave a rewrite system for natural numbers and integers. They represent numbers in *positional notation* (or *place* notation), that is numbers are represented as strings of digits, using a concatenation operator. For example, 13 is represented in base 2 as 1.1.0.1. Implicitly, concatenation associates to the left, that is 13 is in fact denoted ((1.1).0).1. This allows a natural interpretation of the operator dot : $x.y$ represents $2x + y$ (or more generally $bx + y$ in base b). They present their system for integers in base 4 although one can generalise this to any base $b \geq 3$ (but surprisingly not to base 2, see [23]) This system has been proved locally confluent but the authors were not able to prove its termination (it is still a conjecture).

Later, Walters [24] and then De Vries and Yamada [13] presented other systems for integers, and they succeeded in proving their termination. Their termination proof is far from trivial because it uses a quite complicated reduction ordering called *semantic labelling* due to Zantema [25].

De Vries and Yamada [13] were also interested in what they call "real numbers" which are in fact numbers with a finite expansion in the given base. This is not sufficient for our purpose since of course in any given base, there are rational numbers with an infinite expansion.

Concerning computation of Gröbner bases with equational completion, we have to mention the work of Bündgen [7,8] where he shows how it is possible to compute Gröbner bases of polynomials with coefficients in $\mathbb{Z}$, in finite fields, and also in $\mathbb{Q}$ but for this last case he considers an infinite signature and an infinite rewrite system so in practice the completion procedure has to be modified in order to add, at run-time on each examples, the symbols that are really in use. In this paper, we show how this can be done with a completion procedure without any ad-hoc modification. About computation of Gröbner bases on $\mathbb{Z}$ and finite fields by normalised completion, we refer to [21].

The main results of this paper are :

- nicer presentations of natural numbers and integers than the ones mentioned above, with a simpler termination proof, and allowing slightly more efficient reductions;
- the first (as far as we know) presentation of $\mathbb{Q}$ with a convergent and efficient rewrite system;
- computation of Gröbner bases over $\mathbb{Q}$ by use of a general purpose equational completion procedure. We are going to present practical examples obtained with our completion system C*i*ME [11]

1.3 Overview of our Approach

The representation by strings of digits needs some structural rules such as $x.(y.z) \to x.(y+z)$, and we dislike such rules (for proving termination in particular) since they make appear the defined operator $+$. For this reason, our approach starts by simplifying the former notation a bit : instead of writing numbers as strings of digits, we consider digits as unary operators, written in postfix notation : for example the representation of 13 is now $((((\#)1)1)0)1$, where $\#$ is some initial "mark". With this notation, $(x)1$ represents $2x+1$ and $(x)0$ represents $2x$. This method has the advantage of avoiding all cases where in the former approach there was something other than a digit on the right of the dot. In Section 3, we will present a convergent rewrite systems for natural numbers using this notation, with AC operations $+$ and $\times$. Termination is insured by a polynomial interpretation of terms, hence a quite simple termination proof, a simpler proof than all termination proofs of the systems mentioned above.

The next step, in Section 4, is to deal with integers. The first idea we add is to use (in base 2) 2-complement notation. Unfortunately, this presentation leads to a non-terminating rewrite system. To solve this problem, we use a *balanced* notation that can be applied only when the base b is odd (we explain why in Section 4). We were able to exhibit a convergent rewrite system but unfortunately we did not find a simple termination proof like for natural numbers : most of the rules are made decreasing with respect to a polynomial interpretation, but three of them have to be proved terminating by an ad-hoc reasoning.

We then consider rational numbers in Section 5. The harder part is to get reduction rules in order to put a given fraction into a irreducible form, addition and multiplication are then defined easily. The termination proof of this system uses polynomial and exponential interpretations.

Last, in Section 6, we show how to use the previous rewrite system for computing Gröbner bases of polynomials over $\mathbb{Q}[X_1, \ldots, X_n]$, thus solving the open problem mentioned before.

2 Rewrite Presentations of Arithmetic

In the following, we denote by $\mathbb{N}$ the set of natural numbers, by $\mathbb{Z}$ the set of integers, and by $\mathbb{Q}$ the set of rational numbers. We follow the notations for rewriting given by Dershowitz and Jouannaud [14]. In the following, let N be either $\mathbb{N}$, $\mathbb{Z}$ or $\mathbb{Q}$.

Definition 1. A pair (Σ, R) where Σ is a signature and R a rewrite system is said to be a *rewrite presentation* of N if and only if

1. Σ contains two binary symbols $+_N$ and $\times_N$, and also, if N is $\mathbb{Z}$ or $\mathbb{Q}$, a binary symbol $-_N$
2. R is convergent modulo $AC(+_N)$ and $AC(\times_N)$
3. there is a mapping φ from the set of ground terms $T(\Sigma)$ into N, such that for all $t_1,\ t_2$ in $T(\Sigma)$:
   (a) $\varphi(t_1 +_N t_2) = \varphi(t_1) + \varphi(t_2)$
   (b) $\varphi(t_1 \times_N t_2) = \varphi(t_1) \times \varphi(t_2)$
   (c) if N is $\mathbb{Z}$ or $\mathbb{Q}$: $\varphi(t_1 -_N t_2) = \varphi(t_1) - \varphi(t_2)$
   (d) if $t_1 =_R t_2$ then $\varphi(t_1) = \varphi(t_2)$
   (e) φ is one-to-one from the set of R-normal forms $T(\Sigma)/_{\downarrow_R}$ onto N.

We also look for efficient rewrite presentations in the sense that the number of steps of reductions to reduce $t_1 +_N t_2$ and $t_1 \times_N t_2$ to their normal forms should be bounded by a polynomial function of $\log(\varphi(t_1))$ and $\log(\varphi(t_2))$ of degree 2, as the usual algorithms one learns at primary school for decimal numbers.

3 Presentations of Natural Numbers

The classical Peano presentation of natural numbers by the constant 0 and the successor can be easily made a convergent rewrite system. Unfortunately, it leads to very inefficient reductions, thus we are going to use a positional notation.

We give a presentation "parametrised" by the base $b \geq 2$. The signature $\Sigma_{\mathrm{nat}}(b)$ contains

$$0 : Nat \qquad 0_{\mathbb{N}}, 1_{\mathbb{N}}, \ldots, (b-1)_{\mathbb{N}} : Nat \to Nat \qquad +_{\mathbb{N}}, \times_{\mathbb{N}} : Nat, Nat \to Nat$$

$+_{\mathbb{N}}$ and $\times_{\mathbb{N}}$ are AC. The intended meaning of constructors $0, 0_{\mathbb{N}}, 1_{\mathbb{N}}, \ldots, (b-1)_{\mathbb{N}}$ is $\varphi(0) = 0$ and $\varphi((x)d_{\mathbb{N}}) = b\varphi(x) + d$ for each d, $0 \leq d \leq b-1$. In order to have an exact presentation of $\mathbb{N}$ we need the rule $(0)0_{\mathbb{N}} \to 0$ which removes leading zeros. The rewrite system $R_{\mathrm{nat}}(b)$ contains three rules and three families of rules:

$$\begin{array}{l}
(0)0_{\mathbb{N}} \to 0 \qquad x +_{\mathbb{N}} 0 \to x \qquad x \times_{\mathbb{N}} 0 \to 0 \\
(x)d_{\mathbb{N}} +_{\mathbb{N}} (y)d'_{\mathbb{N}} \to (x +_{\mathbb{N}} y)d''_{\mathbb{N}} \quad \text{if } d'' = d + d' \leq b - 1 \\
(x)d_{\mathbb{N}} +_{\mathbb{N}} (y)d'_{\mathbb{N}} \to (x +_{\mathbb{N}} y +_{\mathbb{N}} (0)1_{\mathbb{N}})d''_{\mathbb{N}} \quad \text{if } d + d' \geq b, d'' = d + d' - b \\
x \times_{\mathbb{N}} (y)d_{\mathbb{N}} \to (x \times_{\mathbb{N}} y)0_{\mathbb{N}} +_{\mathbb{N}} \underbrace{x +_{\mathbb{N}} \ldots +_{\mathbb{N}} x}_{d \text{ times}}
\end{array}$$

It can be checked by hand that for an arbitrary b, $R_{\mathrm{nat}}(b)$ is locally confluent, and termination can be proved thanks to the lexicographic combination of the following polynomial interpretations:

$$\begin{array}{ll}
[\![0]\!]_1 = 2 & [\![0]\!]_2 = 2 \\
[\![d_{\mathbb{N}}]\!]_1(x) = x + d & [\![d_{\mathbb{N}}]\!]_2(x) = x + 1 \\
[\![+_{\mathbb{N}}]\!]_1(x, y) = x + y - 2 & [\![+_{\mathbb{N}}]\!]_2(x, y) = x + y \\
[\![\times_{\mathbb{N}}]\!]_1(x, y) = x.y & [\![\times_{\mathbb{N}}]\!]_2(x, y) = x.y
\end{array}$$

Moreover, the systems $R_{\mathrm{nat}}(b)$ have been automatically proved convergent modulo $\mathrm{AC}(+_{\mathbb{N}}) \cup \mathrm{AC}(\times_{\mathbb{N}})$ by C*i*ME in the cases where $b = 2, 3, 4, 5$.

4 Presentations of Integers

4.1 Idea of the Presentation

The idea we had for generalising the presentation of natural numbers to integers was to use the well-known 2-complement notation. Reformulated in a rewriting framework, this means that integers are generated by two constants, say 0 and m, denoting respectively 0 and -1; and two unary functions, say $0_{\mathbb{Z}}$ and $1_{\mathbb{Z}}$, whose semantics is given by $\varphi((x)0_{\mathbb{Z}}) = 2\varphi(x)$ and $\varphi((x)1_{\mathbb{Z}}) = 2\varphi(x) + 1$ (to represent exactly the set $\mathbb{Z}$, we have the two additional relations $(0)0_{\mathbb{Z}} = 0$ and $(m)1_{\mathbb{Z}} = m$). This is a very nice point of view since with such a presentation, rules for addition are exactly the same as for natural numbers, plus the rules:

$$m +_{\mathbb{Z}} m \to (m)0_{\mathbb{Z}} \qquad m +_{\mathbb{Z}} (x)0_{\mathbb{Z}} \to (x +_{\mathbb{Z}} m)1_{\mathbb{Z}} \qquad m +_{\mathbb{Z}} (x)1_{\mathbb{Z}} \to (x)0_{\mathbb{Z}}$$

However, the situation gets very bad for multiplication: natural rules for $\times_{\mathbb{Z}}$ are

$$\begin{array}{rl} x \times_{\mathbb{Z}} 0 \to 0 & \qquad x \times_{\mathbb{Z}} (y)0_{\mathbb{Z}} \to (x \times_{\mathbb{Z}} y)0_{\mathbb{Z}} \\ m \times_{\mathbb{Z}} m \to (0)1_{\mathbb{Z}} & \qquad x \times_{\mathbb{Z}} (y)1_{\mathbb{Z}} \to (x \times_{\mathbb{Z}} y)0_{\mathbb{Z}} +_{\mathbb{Z}} x \end{array}$$

But this is not anymore confluent because $x \times_{\mathbb{Z}} (m)1_{\mathbb{Z}}$ can be rewritten both to $x \times_{\mathbb{Z}} m$ and $(x \times_{\mathbb{Z}} m)0_{\mathbb{Z}} +_{\mathbb{Z}} x$ and this critical pair cannot be turned into a new rewrite rule: from left to right it does not terminate, from right to left it then generates infinitely many other critical pairs.

We strongly conjecture that set of rules given above is in fact confluent on ground terms, hence it could certainly be useful for several specification or theorem proving purposes, but in Section 6, for normalised completion we need a convergent system for open terms also.

So the idea is then to use a presentation which is *balanced* w.r.t. positive and negative numbers, in the sense that for each constructor f there is a corresponding constructor f' such that $f(-x) = -f'(x)$. We want such a symmetry property because if one can defined a convergent rewrite system on natural numbers, then he has immediately a rewrite system on integers by adding the symmetric rules.

Strangely, if we want to construct $\mathbb{Z}$ by such a balanced presentation, it appears that we have to use a base b where b is odd. In the following we present such a construction in base 3, although we believe that it could easily be generalised to any other odd base. Although we first designed this arithmetic notation by our own, it appears[1] that such a notation has been known and rediscovered several times in the literature, as the so-called *balanced ternary numbers*. We refer to Knuth [17], pages 190–192, who listed a large set of references on that notation, who seems to have been used first in India already 1000 years ago! Let us now present this notation on a rewriting point a view.

[1] We thank the anonymous referee who pointed this to us

4.2 Balanced Presentation of $\mathbb{Z}$ in Base 3

The signature Σ_{int} is given by the constructors 0, $0_{\mathbb{Z}}$, $1_{\mathbb{Z}}$, $j_{\mathbb{Z}}$, the AC operators $+_{\mathbb{Z}}$ and $\times_{\mathbb{Z}}$, the binary operator $-_{\mathbb{Z}}$ and the unary operator opp for computing the opposite of a number:

$$0 : Int \quad 0_{\mathbb{Z}}, 1_{\mathbb{Z}}, j_{\mathbb{Z}} : Int \to Int \quad +_{\mathbb{Z}}, \times_{\mathbb{Z}}, -_{\mathbb{Z}} : Int, Int \to Int \quad \mathrm{opp} : Int \to Int$$

The intended meaning of constructors is $\varphi(0) = 0$, $\varphi((x)0_{\mathbb{Z}}) = 3\varphi(x)$, $\varphi((x)1_{\mathbb{Z}}) = 3\varphi(x)+1$ and $\varphi((x)j_{\mathbb{Z}}) = 3\varphi(x)-1$. $1_{\mathbb{Z}}$ and $j_{\mathbb{Z}}$ are symmetric and the sense above, 0 and $0_{\mathbb{Z}}$ are their own symmetric. In order to have a exact presentation of $\mathbb{Z}$ we need the relation $(0)0_{\mathbb{Z}} = 0$.

The idea is to add numbers like the way we learnt in primary school, using the following basic rules:

$$\begin{array}{r} 0 \\ + \quad 0 \\ \hline = 0\,0 \end{array} \qquad \begin{array}{r} 0 \\ + \quad 1 \\ \hline = 0\,1 \end{array} \qquad \begin{array}{r} 0 \\ + \quad j \\ \hline = 0\,j \end{array} \qquad \begin{array}{r} 1 \\ + \quad 1 \\ \hline = 1\,j \end{array} \qquad \begin{array}{r} j \\ + \quad j \\ \hline = j\,1 \end{array} \qquad \begin{array}{r} 1 \\ + \quad j \\ \hline = 0\,0 \end{array}$$

For example, to add 17 and 8, whose representations are $((((0)1_{\mathbb{Z}})j_{\mathbb{Z}})0_{\mathbb{Z}})j_{\mathbb{Z}}$ and $(((0)1_{\mathbb{Z}})0_{\mathbb{Z}})j_{\mathbb{Z}}$, or to add -17 and 8 :

$$\begin{array}{l} \quad {\scriptstyle 0\;0\;j} \\ \quad 1\,j\,0\,j \\ + \quad\;\; 1\,0\,j \\ \hline = 1\,0\,j\,1 = 25 \end{array} \qquad\qquad \begin{array}{l} \quad {\scriptstyle 1\;0\;0} \\ \quad j\,1\,0\,1 \\ + \quad\;\; 1\,0\,j \\ \hline = 0\,j\,0\,0 = -9 \end{array}$$

Thus one can see that addition and subtraction can be computed exactly in the same way (for subtraction it is even not necessary to check which operand has the greatest absolute value, like the usual algorithm).

Let R_{int} be the following set of rules

$$\begin{array}{rlrl}
(0)0_{\mathbb{Z}} \to 0 & (1) & \mathrm{opp}(0) \to 0 & (8) \\
x +_{\mathbb{Z}} 0 \to x & (2) & \mathrm{opp}((x)0_{\mathbb{Z}}) \to (\mathrm{opp}(x))0_{\mathbb{Z}} & (9) \\
(x)0_{\mathbb{Z}} +_{\mathbb{Z}} (y)0_{\mathbb{Z}} \to (x +_{\mathbb{Z}} y)0_{\mathbb{Z}} & (3) & \mathrm{opp}((x)1_{\mathbb{Z}}) \to (\mathrm{opp}(x))j_{\mathbb{Z}} & (10) \\
(x)0_{\mathbb{Z}} +_{\mathbb{Z}} (y)1_{\mathbb{Z}} \to (x +_{\mathbb{Z}} y)1_{\mathbb{Z}} & (4) & \mathrm{opp}((x)j_{\mathbb{Z}}) \to (\mathrm{opp}(x))1_{\mathbb{Z}} & (11) \\
(x)0_{\mathbb{Z}} +_{\mathbb{Z}} (y)j_{\mathbb{Z}} \to (x +_{\mathbb{Z}} y)j_{\mathbb{Z}} & (5) & x \times_{\mathbb{Z}} 0 \to 0 & (12) \\
(x)1_{\mathbb{Z}} +_{\mathbb{Z}} (y)j_{\mathbb{Z}} \to (x +_{\mathbb{Z}} y)0_{\mathbb{Z}} & (6) & x \times_{\mathbb{Z}} (y)0_{\mathbb{Z}} \to (x \times_{\mathbb{Z}} y)0_{\mathbb{Z}} & (13) \\
x -_{\mathbb{Z}} y \to x +_{\mathbb{Z}} \mathrm{opp}(y) & (7) & x \times_{\mathbb{Z}} (y)1_{\mathbb{Z}} \to x +_{\mathbb{Z}} (x \times_{\mathbb{Z}} y)0_{\mathbb{Z}} & (14)
\end{array}$$

$$(x)1_{\mathbb{Z}} +_{\mathbb{Z}} (y)1_{\mathbb{Z}} \to (x +_{\mathbb{Z}} y +_{\mathbb{Z}} (0)1_{\mathbb{Z}})j_{\mathbb{Z}} \tag{15}$$

$$(x)j_{\mathbb{Z}} +_{\mathbb{Z}} (y)j_{\mathbb{Z}} \to (x +_{\mathbb{Z}} y +_{\mathbb{Z}} (0)j_{\mathbb{Z}})1_{\mathbb{Z}} \tag{16}$$

$$x \times_{\mathbb{Z}} (y)j_{\mathbb{Z}} \to (x \times_{\mathbb{Z}} y)0_{\mathbb{Z}} +_{\mathbb{Z}} \mathrm{opp}(x) \tag{17}$$

Let R^2_{int} be the set of rules $\{(2), (15), (16)\}$ and R^1_{int} be the others.

4.3 Termination of R_{int}

Definition 2. Let I_0 be the lexicographic combination of the following three polynomial interpretations :

$[\![0]\!]_1 = 2$	$[\![0]\!]_2 = 2$	$[\![0]\!]_3 = 2$
$[\![0_{\mathbb{Z}}]\!]_1(x) = x + 1$	$[\![0_{\mathbb{Z}}]\!]_2(x) = x$	$[\![0_{\mathbb{Z}}]\!]_3(x) = x + 1$
$[\![1_{\mathbb{Z}}]\!]_1(x) = x + 1$	$[\![1_{\mathbb{Z}}]\!]_2(x) = x + 1$	$[\![1_{\mathbb{Z}}]\!]_3(x) = x + 1$
$[\![j_{\mathbb{Z}}]\!]_1(x) = x + 1$	$[\![j_{\mathbb{Z}}]\!]_2(x) = x + 1$	$[\![j_{\mathbb{Z}}]\!]_3(x) = x + 1$
$[\![+_{\mathbb{Z}}]\!]_1(x, y) = x + y - 2$	$[\![+_{\mathbb{Z}}]\!]_2(x, y) = x + y - 2$	$[\![+_{\mathbb{Z}}]\!]_3(x, y) = x + y - 2$
$[\![\mathrm{opp}]\!]_1(x) = x$	$[\![\mathrm{opp}]\!]_2(x) = x$	$[\![\mathrm{opp}]\!]_3(x) = 2.x$
$[\![-_{\mathbb{Z}}]\!]_1(x, y) = x + y$	$[\![-]\!]_2(x, y) =$ not needed	$[\![-]\!]_3(x, y) =$ not needed
$[\![\times_{\mathbb{Z}}]\!]_1(x, y) = xy - x - y + 2$	$[\![\times_{\mathbb{Z}}]\!]_2(x, y) = xy$	$[\![\times_{\mathbb{Z}}]\!]_3(x, y) = xy$

Lemma 3. *I_0 is AC-compatible. The rules of R^1_{int} decrease w.r.t I_0. The rules of R^2_{int} keep I_0 the same.*

Proof. Verification can be made by hand and has also been proved by C*i*ME.

Definition 4. Let $\mathrm{dig}(t)$ be the number of digits of a term t, defined inductively by $\mathrm{dig}(0) = 0$, $\mathrm{dig}((t)0_{\mathbb{Z}}) = \mathrm{dig}((t)1_{\mathbb{Z}}) = \mathrm{dig}((t)j_{\mathbb{Z}}) = 1 + \mathrm{dig}(t)$ and $\mathrm{dig}(f(t_1, \dots, t_n)) = \mathrm{dig}(t_1) + \cdots + \mathrm{dig}(t_n)$ for any other operator f. If t is a term, we denote $\overline{t}$ the term obtained by *flattening* t w.r.t to AC symbols. Let $I_1(t) = J(\overline{t})$ be the multiset of pairs defined inductively by

- $J(0) = \{\}$, $J((t)0_{\mathbb{Z}}) = J((t)1_{\mathbb{Z}}) = J((t)j_{\mathbb{Z}}) = J(t)$
- $J(t_1 +_{\mathbb{Z}} \cdots +_{\mathbb{Z}} t_n) = \bigcup_{i=1}^n J(t_i) \cup \{(\sum_{i=1}^n \mathrm{dig}(t_i), n)\}$
- $J(f(t_1, \dots, t_n)) = \bigcup_{i=1}^n J(t_i)$ for any other operator f.

Lemma 5. *For any flat terms $s = s_1 +_{\mathbb{Z}} \cdots +_{\mathbb{Z}} s_n$ and $t = t_1 +_{\mathbb{Z}} \cdots +_{\mathbb{Z}} t_m$ (with $n \geq m \geq 1$) such that $\mathrm{dig}(s) = \mathrm{dig}(t)$, and for any u such that $\mathrm{Head}(u) \neq +_{\mathbb{Z}}$, $J(s) > J(t)$ implies $J(\overline{s +_{\mathbb{Z}} u}) > J(\overline{t +_{\mathbb{Z}} u})$.*

Proof. See [12] for the detailed proof.

Lemma 6. *For any term s and t such that $s \to_{R^2_{\text{int}}} t$ at top, we have $I_1(s) > I_1(t)$ and $\mathrm{dig}(s) = \mathrm{dig}(t)$.*

Proof. We simply have to verify that I_1 decreases for any instance of the three rules of R^2_{int}. We have to take care about flattening: for the first rule we have to distinguish two cases depending on whether the instance of x has a $+_{\mathbb{Z}}$ at top or not :

- if $\mathrm{Head}(s) \neq +_{\mathbb{Z}}$, $I_1(s +_{\mathbb{Z}} 0) = J(s +_{\mathbb{Z}} 0) = J(s) \cup \{(\mathrm{dig}(s), 2)\} > J(s) = I_1(s)$
.

– if $n \geq 2$,

$$I_1(s +_{\mathbb{Z}} 0) = J(s_1 +_{\mathbb{Z}} \cdots +_{\mathbb{Z}} s_n +_{\mathbb{Z}} 0) = \bigcup_{i=1}^{n} J(s_i) \cup \{(\sum_{i=1}^{n} \mathrm{dig}(s_i), n+1)\}$$
$$I_1(s) = J(s_1 +_{\mathbb{Z}} \cdots +_{\mathbb{Z}} s_n) = \bigcup_{i=1}^{n} J(s_i) \cup \{(\sum_{i=1}^{n} \mathrm{dig}(s_i), n)\}$$

hence the former in larger than the latter.

For the second rule :

$$J((s)1_{\mathbb{Z}} +_{\mathbb{Z}} (t)1_{\mathbb{Z}}) = J(s) \cup J(t) \cup \{(\mathrm{dig}(s) + \mathrm{dig}(t) + 2, 2)\}$$
$$J((s +_{\mathbb{Z}} t +_{\mathbb{Z}} (0)1_{\mathbb{Z}})j_{\mathbb{Z}}) = \bigcup_{i=1}^{n} J(s_i) \cup \bigcup_{j=1}^{m} J(t_j) \cup \{(\mathrm{dig}(s) + \mathrm{dig}(t) + 1, n+m+1)\}$$

where $s = s_1 +_{\mathbb{Z}} \cdots +_{\mathbb{Z}} s_n$ and $t = t_1 +_{\mathbb{Z}} \cdots +_{\mathbb{Z}} t_m$ with $n \geq 1$ and $m \geq 1$. But $J(s) \geq \bigcup_{i=1}^{n} J(s_i)$, $J(t) \geq \bigcup_{j=1}^{m} J(t_j)$ and $(\mathrm{dig}(s) + \mathrm{dig}(t) + 2, 2) > (\mathrm{dig}(s) + \mathrm{dig}(t) + 1, n+m+1)$. For the third rule, this is the same as the second, by permuting $1_{\mathbb{Z}}$ and $j_{\mathbb{Z}}$.

Corollary 7. *For any term s and t such that $s \to_{R^2_{\mathrm{int}}} t$ we have $I_1(s) > I_1(t)$ and $\mathrm{dig}(s) = \mathrm{dig}(t)$.*

Proof. Easy induction on the depth of the context, using Lemma 5 in the case where the context operator is $+_{\mathbb{Z}}$.

Corollary 8. *The system R_{int} terminates modulo AC.*

Proof. Suppose R_{int} does not terminate. Then there is an infinite sequence of ground terms $s_1 \to_{R_{\mathrm{int}}} s_2 \to_{R_{\mathrm{int}}} s_3 \to_{R_{\mathrm{int}}} \cdots$. By Lemma 3, $I_0(s_1) \geq I_0(s_2) \geq \cdots$ hence there is an integer n such that $I_0(s_n) = I_0(s_{n+1}) = \cdots$, hence from the step s_n only rules of R^2_{int} are applied, but this contradicts the previous corollary.

4.4 Confluence and Efficiency of R_{int}

Proposition 9. *R_{int} is convergent.*

Proof. We verified with C*i*ME that it is locally confluent modulo AC. For information, this verification requires the computation of 110 AC-critical pairs and is made in 2.43 seconds on a PC Pentium 90; and with *AC-complete ACU unification* [3] it computes 76 critical pairs, in 1.93 seconds.

Proposition 10. *R_{int} is efficient : if one uses an innermost strategy such that, at a given position, the rules for addition which do not introduce a carry have priority over the others, then for each ground terms t_1, t_2 in $T(\Sigma)/\downarrow_R$, the number of reduction steps for normalising $t_1 +_{\mathbb{Z}} t_2$ is bounded by $1 + 2(d_1 + d_2)$; and the number of reduction steps for normalising $t_1 \times_{\mathbb{Z}} t_2$ is bounded by $1 + 4(d_1 + d_2) + d_1^2 + d_2^2 + 5d_1d_2$ where $d_i = \mathrm{dig}(t_i)$.*

Proof. By an easy induction.

5 Presentations of Rational Numbers

A rational number p/q is represented by a pair of two integer numbers: its numerator p and its denominator q. Since we want to check equality by rewriting techniques, a rational number must have a unique normal form. The normal form usually used is when q is a positive integer and p and q have no common divisor. We have add some more operators to the signature Σ_{int} in order to be able to reduce a rational into its normal form:

$$sign, abs : Int \to Int \qquad sign^?, min, min', min'' : Int, Int \to Int$$
$$abs^? : Int, Int, Int \to Int \qquad min^? : Int, Int, Int, Int \to Int$$

The semantics of the new operators is specified as follows:

$$\begin{aligned}
\varphi(sign(x)) &= 0 \text{ if } \varphi(x) = 0,\ 1 \text{ if } \varphi(x) > 0,\ -1 \text{ if } \varphi(x) < 0\\
\varphi(sign^?(x,y)) &= \varphi(y) \text{ if } \varphi(x) = 0,\ 1 \text{ if } \varphi(x) > 0,\ -1 \text{ if } \varphi(x) < 0\\
\varphi(abs(x)) &= |\varphi(x)|\\
\varphi(abs^?(x,y,z)) &= sign(\varphi(x))\varphi(y) \text{ if } \varphi(x) \neq 0,\ sign(\varphi(z))\varphi(y) \text{ otherwise}\\
\varphi(min(x,y)) &= \varphi(x) \text{ if } |\varphi(x)| \leq |\varphi(y)|,\ \varphi(y) \text{ otherwise}\\
\varphi(min'(x,y)) &= \varphi(x) \text{ if } |3\varphi(x)+1| \leq |3\varphi(y)+1|,\ \varphi(y) \text{ otherwise}\\
\varphi(min''(x,y)) &= \varphi(x) \text{ if } |3\varphi(x)-1| \leq |3\varphi(y)-1|,\ \varphi(y) \text{ otherwise}\\
\varphi(min^?(x,y,z,u)) &= \varphi(y) \text{ if } \varphi(x) > 0 \text{ or } (\varphi(x) = 0 \text{ and } \varphi(u) \geq 0),\ \varphi(z) \text{ otherwise}
\end{aligned}$$

In a first step, we add the following set of rules R_{incr} to R_{int}:

$$\begin{aligned}
sign(x) &\to sign^?(x,0)\\
sign^?(0,x) &\to x\\
sign^?((x)0_{\mathbb{Z}},y) &\to sign^?(x,y)\\
sign^?((x)1_{\mathbb{Z}},y) &\to sign^?(x,(0)1_{\mathbb{Z}})\\
sign^?((x)j_{\mathbb{Z}},y) &\to sign^?(x,(0)j_{\mathbb{Z}})
\end{aligned}
\qquad
\begin{aligned}
abs(x) &\to abs^?(x,x,0)\\
abs^?((x)0_{\mathbb{Z}},y,z) &\to abs^?(x,y,z)\\
abs^?((x)1_{\mathbb{Z}},y,z) &\to abs^?(x,y,(0)1_{\mathbb{Z}})\\
abs^?((x)j_{\mathbb{Z}},y,z) &\to abs^?(x,y,(0)j_{\mathbb{Z}})\\
abs^?(0,x,0) &\to 0\\
abs^?(0,x,(0)1_{\mathbb{Z}}) &\to x\\
abs^?(0,x,(0)j_{\mathbb{Z}}) &\to \mathrm{opp}(x)
\end{aligned}$$

$$\begin{aligned}
min(x,y) &\to min^?(abs(y)-abs(x),x,y,0)\\
min'(x,y) &\to min^?(abs((y)1_{\mathbb{Z}})-abs((x)1_{\mathbb{Z}}),x,y,0)\\
min''(x,y) &\to min^?(abs((y)j_{\mathbb{Z}})-abs((x)j_{\mathbb{Z}}),x,y,0)\\
min^?((x)0_{\mathbb{Z}},y,z,u) &\to min^?(x,y,z,u)\\
min^?((x)1_{\mathbb{Z}},y,z,u) &\to min^?(x,y,z,(0)1_{\mathbb{Z}})\\
min^?((x)j_{\mathbb{Z}},y,z,u) &\to min^?(x,y,z,(0)j_{\mathbb{Z}})\\
min^?(0,x,y,0) &\to x\\
min^?(0,x,y,(0)1_{\mathbb{Z}}) &\to x\\
min^?(0,x,y,(0)j_{\mathbb{Z}}) &\to y
\end{aligned}$$

This following lemma establishes the soundness of R_{incr} w.r.t. φ

Lemma 11. *φ remains the same on both sides of any equation in R_{incr}.*

Proof. This is just an easy verification.

5.1 Termination and Confluence of $R_{\text{int}} \cup R_{\text{incr}}$

Definition 12. We extend the interpretation I_0 by adding to the first component:

$$\begin{array}{ll}
[\![abs]\!]_1(x) = 6x & [\![min]\!]_1(x,y) = 16x + 16y \\
[\![abs^?]\!]_1(x,y,z) = 3x + y + z & [\![min']\!]_1(x,y) = 20x + 20y \\
[\![sign]\!]_1(x) = 4x & [\![min'']\!]_1(x,y) = 20x + 20y \\
[\![sign^?]\!]_1(x,y) = 2x + y & [\![min^?]\!]_1(x,y,z,u) = 3x + y + z + u
\end{array}$$

Lemma 13. *The rules of R_{incr} decrease w.r.t. I_0. Consequently, $R_{\text{int}} \cup R_{\text{incr}}$ is terminating.*

Lemma 14. *$R_{\text{int}} \cup R_{\text{incr}}$ is convergent.*

Proof. We verified with C*i*ME that it is locally confluent modulo AC. For information, this verification requires the computation of 157 AC-critical pairs and is made in 9.18 seconds on a PC Pentium 90; and with *AC-complete ACU unification* [3] it computes 123 critical pairs, in 8.71 seconds.

5.2 Computing Irreducible Fractions

A well-known method for reducing a rational number p/q to its normal form is to divide both p and q by their GCD. We do not adopt this method here since we would have to introduce too complicated operators, the first one for the division and the second one for computing the GCD of two integers (it would raise the problem of division by 0). We have introduced an operator called f in the following, which performs the GCD computation at the same time as the division (note that we consider that a GCD is always positive). This operator has the semantics:

$$\varphi(f(x,y)) = \varphi(x)/gcd(\varphi(x), \varphi(y))$$

It can be recursively defined by taking advantage of the representation of the integers in base 3. We adapted the idea of J. Stein referred in [17] which take advantage of the representation of two integers in base 2 for computing their GCD without any division.

Let R_{f} be the following set of rules:

$$\begin{aligned}
f(0,x) &\to 0 \\
f(x,0) &\to sign(x) \\
f((x)0_{\mathbb{Z}},(y)0_{\mathbb{Z}}) &\to f(x,y) \\
f((x)0_{\mathbb{Z}},(y)1_{\mathbb{Z}}) &\to (f(x,(y)1_{\mathbb{Z}}))0_{\mathbb{Z}}
\end{aligned}$$

$$
\begin{aligned}
f((x)0_{\mathbb{Z}}, (y)j_{\mathbb{Z}}) &\to (f(x, (y)j_{\mathbb{Z}}))0_{\mathbb{Z}} \\
f((x)1_{\mathbb{Z}}, (y)0_{\mathbb{Z}}) &\to f((x)1_{\mathbb{Z}}, y) \\
f((x)1_{\mathbb{Z}}, (y)1_{\mathbb{Z}}) &\to (f(x -_{\mathbb{Z}} min'(x, y), (y)1_{\mathbb{Z}}))0_{\mathbb{Z}} +_{\mathbb{Z}} \\
&\qquad f(min((x)1_{\mathbb{Z}}, (y)1_{\mathbb{Z}}), x -_{\mathbb{Z}} y) \\
f((x)1_{\mathbb{Z}}, (y)j_{\mathbb{Z}}) &\to (f(x +_{\mathbb{Z}} min''(\mathrm{opp}(x), y), (y)j_{\mathbb{Z}}))0_{\mathbb{Z}} +_{\mathbb{Z}} \\
&\qquad f(min((x)1_{\mathbb{Z}}, (\mathrm{opp}(y))1_{\mathbb{Z}}), x +_{\mathbb{Z}} y) \\
f((x)j_{\mathbb{Z}}, (y)0_{\mathbb{Z}}) &\to f((x)j_{\mathbb{Z}}, y) \\
f((x)j_{\mathbb{Z}}, (y)1_{\mathbb{Z}}) &\to (f(x +_{\mathbb{Z}} min'(\mathrm{opp}(x), y), (y)1_{\mathbb{Z}}))0_{\mathbb{Z}} +_{\mathbb{Z}} \\
&\qquad f(min((x)j_{\mathbb{Z}}, (\mathrm{opp}(y))j_{\mathbb{Z}}), x +_{\mathbb{Z}} y) \\
f((x)j_{\mathbb{Z}}, (y)j_{\mathbb{Z}}) &\to (f(x -_{\mathbb{Z}} min''(x, y), (y)j_{\mathbb{Z}}))0_{\mathbb{Z}} +_{\mathbb{Z}} \\
&\qquad f(min((x)j_{\mathbb{Z}}, (y)j_{\mathbb{Z}}), x -_{\mathbb{Z}} y)
\end{aligned}
$$

f is well-defined in the sense that φ remains constant for the rules in R_{f}:

Lemma 15. *φ remains the same on both sides of any equation in R_{f}.*

Proof. See [12] for the detailed proof.

Definition 16. We extend again the first lexicographic component of I_0 as follows:

$$[\![f(x, y)]\!]_1 = 50^{|\varphi(x)\varphi(y)|}([\![x]\!]_1 + [\![y]\!]_1)$$

Lemma 17. *The rules of R_{f} decrease w.r.t. I_0.*

Proof. See [12] for the detailed proof.

Lemma 18. *For all terms s, t and context $C[x]_p$ in $\mathcal{T}(\Sigma_{int} \cup \Sigma_{incr} \cup \{f\}, \mathcal{X})$, $\varphi(s) = \varphi(t)$ and $[\![s]\!]_1 < [\![t]\!]_1$ implies $[\![C[s]_p]\!]_1 < [\![C[t]_p]\!]_1$.*

Proof. A straightforward induction on the depth of the context.

Corollary 19. *$R_{\mathrm{int}} \cup R_{\mathrm{incr}} \cup R_{\mathrm{f}}$ is terminating.*

Lemma 20. *$R_{\mathrm{int}} \cup R_{\mathrm{incr}} \cup R_{\mathrm{f}}$ is convergent.*

Proof. We verified with C*i*ME that it is locally confluent modulo AC.

5.3 Presentation of $\mathbb{Q}$ — Termination and Confluence

Now we can give a presentation for the rational numbers with the signature Σ_{rat}:

$$\mathrm{rat}, / : Int, Int \to Rat \qquad\qquad +_{\mathbb{Q}}, \times_{\mathbb{Q}} : Rat, Rat \to Rat$$

The symbol rat denotes any fraction while / denotes irreducible fractions. Notice that division by zero is allowed, but such alien terms can be avoided by introducing a sort for non-null integers, and one can verify that the following rules are then well-sorted. The rationals numbers are presented by R_{rat}:

$$\begin{aligned}
\mathrm{rat}(x,y) &\to (sign(y) \times_{\mathbb{Z}} f(x,y))/(sign(y) \times_{\mathbb{Z}} f(y,x)) \\
(x/y) +_{\mathbb{Q}} (u/v) &\to \mathrm{rat}((x \times_{\mathbb{Z}} v) +_{\mathbb{Z}} (u \times_{\mathbb{Z}} y), y \times_{\mathbb{Z}} v) \\
(x/y) \times_{\mathbb{Q}} (u/v) &\to \mathrm{rat}(x \times_{\mathbb{Z}} u, y \times_{\mathbb{Z}} v)
\end{aligned}$$

Definition 21. Let I_2 be the lexicographic combination of the following two interpretations

$$\begin{array}{ll}
[\![+_{\mathbb{Q}}]\!]_4(x,y) = x+y+1 & [\![\mathrm{rat}]\!]_5(x,y) = x+y+1 \\
[\![\times_{\mathbb{Q}}]\!]_4(x) = x+y+1 & [\![/]\!]_5(x) = x+y \\
[\![g]\!]_4(x_1,\ldots,x_n) = \sum_{i=1}^n x_i & [\![h]\!]_5(x_1,\ldots,x_n) = \sum_{i=1}^n x_i
\end{array}$$

for all $g \notin \{+_{\mathbb{Q}}, \times_{\mathbb{Q}}\}$ and for all $h \notin \{\mathrm{rat}, /\}$

Lemma 22. $R_{\mathrm{int}} \cup R_{\mathrm{incr}} \cup R_{\mathrm{f}} \cup R_{\mathrm{rat}}$ *is terminating on well-typed terms.*

Proof. If a term t has type *Int* then, by Definition 21, $[\![t]\!]_4 = [\![t]\!]_5 = 0$. The rules of R_{rat} strictly decrease w.r.t I_2, and the others rules keep the same interpretation for well-typed terms: $(0,0)$. Together with corollary 19, this gives the result.

Lemma 23. $R_{\mathrm{int}} \cup R_{\mathrm{incr}} \cup R_{\mathrm{f}} \cup R_{\mathrm{rat}}$ *is convergent on well-typed terms.*

Proof. We verified with C*i*ME that it is locally confluent modulo $\mathrm{AC}(+_{\mathbb{Z}})$, $\mathrm{AC}(\times_{\mathbb{Z}})$, $\mathrm{C}(+_{\mathbb{Q}})$ and $\mathrm{C}(\times_{\mathbb{Q}})$. The operators $+_{\mathbb{Q}}$ and $\times_{\mathbb{Q}}$ are only commutative here: if we add associativity, the system is not anymore confluent, and it seems that its completion generates infinitely many critical pairs.

6 Applications to Gröbner Bases

We present now our first experiments of use of normalised completion for computing Gröbner bases of polynomials with rational coefficients.

6.1 Presentation of Polynomials

As in [1, 7], polynomials are presented by the signature:

$$+_{\mathbb{P}}, \times_{\mathbb{P}} : Poly, Poly \to Poly \qquad -_{\mathbb{P}} : Poly \to Poly \qquad 0_{\mathbb{P}}, 1_{\mathbb{P}} : Poly \to Poly$$

satisfying the usual presentation of commutative rings:

$$\begin{array}{rl}
P +_{\mathbb{P}} -_{\mathbb{P}}(P) \to 0_{\mathbb{P}} & \\
-_{\mathbb{P}}(-_{\mathbb{P}}(P)) \to P & P +_{\mathbb{P}} 0_{\mathbb{P}} \to P \\
-_{\mathbb{P}}(P +_{\mathbb{P}} Q) \to -_{\mathbb{P}}(P) +_{\mathbb{P}} -_{\mathbb{P}}(Q) & -_{\mathbb{P}}(0_{\mathbb{P}}) \to 0_{\mathbb{P}} \\
P \times_{\mathbb{P}} (Q +_{\mathbb{P}} R) \to (P \times_{\mathbb{P}} Q) +_{\mathbb{P}} (P \times_{\mathbb{P}} R) & P \times_{\mathbb{P}} 1_{\mathbb{P}} \to P \\
P \times_{\mathbb{P}} -_{\mathbb{P}}(Q) \to -_{\mathbb{P}}(P \times_{\mathbb{P}} Q) & P \times_{\mathbb{P}} 0_{\mathbb{P}} \to 0_{\mathbb{P}}
\end{array}$$

Rational coefficients are embedded into polynomials by a function $\iota : Rat \to Poly$ satisfying:

$$
\begin{aligned}
0_{\mathbb{P}} &\to \iota(0/(0)1_{\mathbb{Z}}) \\
1_{\mathbb{P}} &\to \iota((0)1_{\mathbb{Z}}/(0)1_{\mathbb{Z}}) \\
-_{\mathbb{P}}(P) &\to P \times_{\mathbb{P}} \iota((0)j_{\mathbb{Z}}/(0)1_{\mathbb{Z}}) \\
\iota(x) +_{\mathbb{P}} \iota(y) &\to \iota(x +_{\mathbb{Q}} y) \\
\iota(x) \times_{\mathbb{P}} \iota(y) &\to \iota(x \times_{\mathbb{Q}} y) \\
\iota(0/(0)1_{\mathbb{Z}}) +_{\mathbb{P}} P &\to P \\
\iota(0/(0)1_{\mathbb{Z}}) \times_{\mathbb{P}} P &\to \iota(0/(0)1_{\mathbb{Z}}) \\
\iota((0)1_{\mathbb{Z}}/(0)1_{\mathbb{Z}}) \times_{\mathbb{P}} P &\to P \\
P +_{\mathbb{P}} P &\to \iota(((0)1_{\mathbb{Z}})j_{\mathbb{Z}}/(0)1_{\mathbb{Z}}) \times_{\mathbb{P}} P \\
(\iota(x) \times_{\mathbb{P}} P) +_{\mathbb{P}} P &\to \iota(x +_{\mathbb{Q}} (0)1_{\mathbb{Z}}/(0)1_{\mathbb{Z}}) \times_{\mathbb{P}} P \\
(\iota(x) \times_{\mathbb{P}} P) +_{\mathbb{P}} (\iota(y) \times_{\mathbb{P}} P) &\to \iota(x +_{\mathbb{Q}} y) \times_{\mathbb{P}} P
\end{aligned}
$$

The union of the two previous systems is known to be convergent [1,7]. However, the union of these systems with the systems for integers and rationals is not anymore confluent. To make it confluent, we need to add commutative ring rules for integers and rationals. We have been able to build a locally confluent system and we conjecture that this system is terminating but we do not have a proof. Nevertheless, we tried to use this system for computing Gröbner bases.

6.2 Computation of Gröbner Bases

To use all these rewrite systems inside the normalised completion algorithm, we declare them as the theory S modulo which completion is performed. Moreover, the algorithm allows us to design our own *normalising pair* [21]. In this particular case, we implemented the following *symmetrisation* rule which mimics the behaviour of the classical Gröbner bases algorithms:

$$\iota(x/y) \times_{\mathbb{P}} P = Q \;\vdash\; P = \iota(y/x) \times_{\mathbb{P}} Q \text{ if } P > Q$$

We now describe roughly what happens on an example.

Example 24. To compute Gröbner basis of the ideal of $\mathbb{Q}[X,Y]$ generated by $\{2X^2Y - Y, 3XY^2 - X\}$, we run normalised completion on the initial set of equations

$$
\begin{aligned}
\iota(2_{\mathbb{Q}}) \times_{\mathbb{P}} X \times_{\mathbb{P}} X \times_{\mathbb{P}} Y +_{\mathbb{P}} -_{\mathbb{P}}(Y) = 0_{\mathbb{P}} \\
\iota(3_{\mathbb{Q}}) \times_{\mathbb{P}} X \times_{\mathbb{P}} Y \times_{\mathbb{P}} Y +_{\mathbb{P}} -_{\mathbb{P}}(X) = 0_{\mathbb{P}}
\end{aligned}
$$

where $1_{\mathbb{Q}} \to (0)1_{\mathbb{Z}}/(0)1_{\mathbb{Z}}$, $2_{\mathbb{Q}} \to 1_{\mathbb{Q}} +_{\mathbb{Q}} 1_{\mathbb{Q}}$, $3_{\mathbb{Q}} \to 2_{\mathbb{Q}} +_{\mathbb{Q}} 1_{\mathbb{Q}}$. The result is the following set of three rules

$$
\begin{aligned}
X \times_{\mathbb{P}} X \times_{\mathbb{P}} Y &\to \iota((0)1_{\mathbb{Z}}/((0)1_{\mathbb{Z}})j_{\mathbb{Z}}) \times_{\mathbb{P}} Y \\
Y \times_{\mathbb{P}} Y &\to \iota(((0)1_{\mathbb{Z}})j_{\mathbb{Z}}/((0)1_{\mathbb{Z}})0_{\mathbb{Z}}) \times_{\mathbb{P}} X \times_{\mathbb{P}} X \\
X \times_{\mathbb{P}} X \times_{\mathbb{P}} X &\to \iota((0)1_{\mathbb{Z}}/((0)1_{\mathbb{Z}})j_{\mathbb{Z}}) \times_{\mathbb{P}} X
\end{aligned}
$$

which corresponds to $\{X^2Y \to \frac{1}{2}Y, Y^2 \to \frac{2}{3}Y, X^3 \to \frac{1}{2}X\}$.

On this example, time for computing the basis is quite large : 19.20 seconds (on PC Pentium 90) although only 14 critical pairs are computed. The detailed computation times reveals that 99% of the time is spent in normalisation with respect to the hundreds of rewrite rules we have. Notice that specialised Gröbner bases computation systems can compute this in less than a second. The conclusion is that in practice, to efficiently simulate Gröbner bases computation, we also need to simulate efficiently the polynomial normalisation which is usually built-in in Gröbner bases algorithms.

7 Conclusion

We obtained a significant step towards the goal we would like to achieve. However, there are still many possible improvements, and this work raises interesting questions for future work:

- can we improve the presentation of rational numbers (in particular $+_{\mathbb{Q}}$ and $\times_{\mathbb{Q}}$ should be AC)?
- can we prove termination of R_{int} with a more "standard" technique?
- we need to prove the termination of the rewrite system presenting polynomials with rational coefficients.
- can we design an efficient normalisation algorithm? In particular, in C*i*ME we do not have a many-to-one AC matching [9], that is an AC-matching algorithm that tries to match a term with several rules at a time. This may improve the efficiency of the rewrite procedure, but we would like also to have a kind of compilation of the whole normalisation process.

References

1. L. Bachmair and H. Ganzinger. Buchberger's algorithm: A constraint-based completion procedure. In J.-P. Jouannaud, editor, *First International Conference on Constraints in Computational Logics*, volume 845 of *Lecture Notes in Computer Science*, pages 285–301, München, Germany, Sept. 1994. Springer-Verlag.
2. R. V. Book, editor. *4th International Conference on Rewriting Techniques and Applications*, volume 488 of *Lecture Notes in Computer Science*, Como, Italy, Apr. 1991. Springer-Verlag.
3. A. Boudet, E. Contejean, and C. Marché. AC-complete unification and its application to theorem proving. In Ganzinger [15], pages 18–32.
4. B. Buchberger. *An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-Dimensional Ideal.* PhD thesis, University of Innsbruck, Austria, 1965. (in German).
5. B. Buchberger. History and basic features of the critical pair / completion procedure. *Journal of Symbolic Computation*, 3(1), Feb. 1987.
6. B. Buchberger and R. Loos. Algebraic simplification. In *Computer Algebra, Symbolic and Algebraic Computation. Computing Supplementum 4*. Springer-Verlag, 1982.

7. R. Bündgen. Simulating Buchberger's algorithm by a Knuth-Bendix completion procedure. In Book [2].
8. R. Bündgen. *Term Completion versus Algebraic Completion*. PhD thesis, Universität Tübingen, 1991.
9. T. Chen and S. Anantharaman. Storm: A many-to-one associative-commutative matcher. In J. Hsiang, editor, *6th International Conference on Rewriting Techniques and Applications*, volume 914 of *Lecture Notes in Computer Science*, Kaiserslautern, Germany, Apr. 1995. Springer-Verlag.
10. D. Cohen and P. Watson. An efficient representation of arithmetic for term rewriting. In Book [2], pages 240–251.
11. E. Contejean and C. Marché. CiME: Completion Modulo E. In Ganzinger [15], pages 416–419. System Description available at `http://www.lri.fr/~demons/cime.html`.
12. E. Contejean, C. Marché, and L. Rabehasaina. Rewrite systems for natural, integral, and rational arithmetic. Research report, Laboratoire de Recherche en Informatique, Université de Paris-Sud, Orsay, France, 1997.
13. F.-J. de Vries and J. Yamada. On termination of rewriting with real numbers. In M. Takeichi, editor, *Functional Programming II, JSSST'94*, volume 10 of *Lecture Notes on Software Gaku*. Kindai-kagaku-sya, Tokyo, Nov. 1994.
14. N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–309. North-Holland, 1990.
15. H. Ganzinger, editor. *7th International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, New Brunswick, NJ, USA, July 1996. Springer-Verlag.
16. A. Kandri-Rody, D. Kapur, and F. Winkler. Knuth-Bendix procedure and Buchberger algorithm — a synthesis. In *Proc. of the 20th Int. Symp. on Symbolic and Algebraic Computation, Portland, Oregon*, pages 55–67, 1989.
17. D. E. Knuth. *The art of computer programming*, volume 2. Addison-Wesley, 2nd edition, 1981.
18. D. E. Knuth and P. B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.
19. P. Le Chenadec. *Canonical forms in finitely presented algebras*. Pitman, London, 1986.
20. R. Loos. Term reduction systems and algebraic algorithms. In *Proceedings of the Fifth GI Workshop on Artificial Intelligence*, pages 214–234, Bad Honnef, West Germany, 1981. Available as *Informatik Fachberichte*, Vol. 47.
21. C. Marché. Normalized rewriting: an alternative to rewriting modulo a set of equations. *Journal of Symbolic Computation*, 21(3):253–288, 1996.
22. L. Pottier. Algorithmes de complétion et généralisation en logique du premier ordre. Thèse de doctorat. Université de Nice, 1989.
23. L. Rabehasaina. Systèmes de réécriture pour l'arithmétique. Rapport de stage de magistère, Université Paris-sud, 1996.
24. H. R. Walters. A complete term rewriting system for decimal integer arithmetic. Tech. Report CS-9435, CWI, Amsterdam, 1994.
25. H. Zantema. Termination of term rewriting by semantic labelling. Technical Report 92-38, RUU-CS, University Utrecht, Dec. 1992.