PSEUDO-DIVISION MACHINE (I): A MODEL OF SYMBOLIC COMPUTATION

SHE ZHIKUN, XIA BICAN AND ZHENG ZHIMING

School of Mathematical Sciences, Peking University, Beijing 100871, China
E-mail: zkshe@mail1.math.pku.edu.cn, xbc@math.pku.edu.cn, zzheng@pku.edu.cn

Through careful study of the theory of real Turing machine and the algorithm of pseudo-division, we define a new machine, called pseudo-division machine, which is a formal model of computation over ring. Especially, it applies to symbolic computation and it represents algorithms naturally. Some fundamental concepts are defined and some primary results presented.

1 Introduction

The study of computability and computational complexity originated in the work of logicians such as Gödel, Turing, Church, Kleene, and Post in the 1930s. The model of computation that developed in the following decades, the Turing machine, has been extraordinarily successful in giving the foundations and framework for theoretical computer science [3,6]. Based on the point of view that "the Turing model with its dependence on 0s and 1s is fundamentally inadequate for giving such a foundation to the theory of modern scientific computation, where most of the algorithms are real number algorithms", L. Blum, M. Shub and S. Smale [1,2] presented a formal model of real computation , which is now called "the real Turing machine" or "the BSS machine". Their theory preserves the Turing theory as a special case. Furthermore, by providing a bridge between complexity theory and the fields of analysis, geometry, and topology, the real Turing machine may give us possibilities for new attack on the classical problems of complexity theory, which heretofore have been attacked using only the tools of logic and combinatorics.

The computation on computers includes two aspects: numerical computation and symbolic computation. The study of symbolic computation originated in the 1950s and has gained wide interests since the 1970s. In 1978, Wen-tsün Wu [8,9] proposed a decision procedure for proving geometry theorems of "equality type", i.e. the hypotheses and conclusions of the statements consist of polynomial equations only. This is a very efficient method for mechanically proving elementary geometry theorems (of equality type) [5]. Now, Wu's method has been one of the most famous algorithms in the field of symbolic computation. On the other hand, the complexity of Wu's method has rarely been studied. To discuss the computation complexity of Wu's method,

as well as many other famous algorithms such as Gröbner basis method [4], from the viewpoint of theoretical computer science is our motivation.

The point of view of this paper is that the symbolic computation, which is over the objects having some basic algebraic structures, is quite different from the numerical computation, which is over numbers. Therefore, although the BSS machine, by its definition, would apply to computation over any ring or field, it is not suitable for our study of the complexity of Wu's method. For example, the pseudo-division of two polynomials in several variables with respect to one main variable can not be represented in the frame of the BSS machine naturally. We need a new formal model that would describe the essential properties of symbolic computation better. The goal of our work is to develop a formal theory of symbolic computation so that we can study the complexity of algorithms defined under the framework. The work presented in this paper is our primary results.

2 Pseudo-Division

Let $\mathbb{D}$ be a *unique factorization domain* (UFD), A be a polynomial in $\mathbb{D}[x_1, ..., x_n]$ and x_k a fixed variable. While considered as a polynomial in x_k, A can be written as

$$\begin{array}{l} A = A_0 x_k^a + A_1 x_k^{a-1} + \cdots + A_a, \\ A_i \in \mathbb{D}[x_1, ..., x_{k-1}, x_{k+1}, ..., x_n], \end{array} \tag{1}$$

where a is the degree of A in x_k and denoted by $\deg(A, x_k)$, A_0 is the *leading coefficient* of A in x_k, denoted by $\mathrm{lc}(A, x_k)$.

Let A and B be two polynomials in $\mathbb{D}[x_1, ..., x_n]$ and $B \neq 0, b = \deg(B, x_k)$, $a = \deg(A, x_k)$. For pseudo-dividing A by B, considered as polynomials in x_k, we have a division algorithm as follows [7]. Let $R = A$; repeat the following process until $r = \deg(R, x_k) < b : R \leftarrow B_0 R - R_0 x_k^{r-b} B$, where $R_0 = \mathrm{lc}(R, x_k)$. Finally, one obtains two polynomials Q and R in $\mathbb{D}[x_1, ..., x_n]$ satisfying the relation

$$I^s A = QB + R, \tag{2}$$

where $I = \mathrm{lc}(B, x_k)$, $s = \max(a - b + 1, 0)$, $\deg(R, x_k) < b$. In case $b = 0$, $R = 0$ and $Q = B^a A$.

Remark 1. *Two* standard *ways to represent polynomials are* dense *representation (where every monomial is specified) and* sparse *representations (which disregard terms whose coefficient is* 0*). For pseudo-division, polynomials are regarded as univariate and every term is specified.*

Let A and B be two polynomials in the form of Eq. (1) and denoted by $A = (A_0, ..., A_a)$ and $B = (B_0, ..., B_b)$, respectively. From the viewpoint of fraction-free Gauss elimination, we have a `prem` algorithm as follows. Let $(P, Q, t) \leftarrow (A, B, a - b)$; repeat the following two steps until $t < 0$:

1. $(P, Q, t) \leftarrow (g(P, Q), Q, t - 1)$, where $P \leftarrow g(P, Q)$ means
$$P_i \leftarrow Q_0 P_i - P_0 Q_i,\ 0 \le i \le b + t,\ Q_j = 0 \text{ if } j > b; \tag{3}$$
2. $(P, Q, t) \leftarrow \text{ShiftL}(P, Q, t)$, which means $P_i \leftarrow P_{i+1}\,(0 \le i \le b + t - 1)$, $Q_j \leftarrow Q_j$, $t \leftarrow t$.

Return P while $t < 0$. We may represent the `prem` algorithm schematically as in Fig. (1).

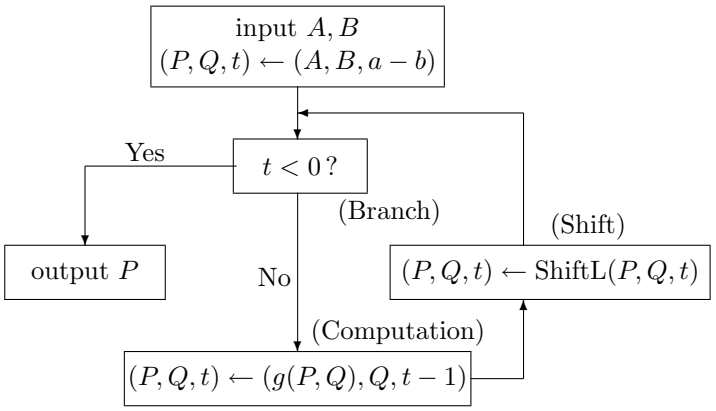

Figure 1. A Machine For Pseudo-Division

3 The Pseudo-Division Machine

Suppose R be a ring. We list some notations defined in [1]. Let

$$R^\infty = \bigsqcup_{n \ge 0} R^n,$$

where for $n > 0$, R^n is the standard n-dimensional space over R and R^0 is the 0-dimensional space with just one point $\mathbf{0}$. For $x \in R^n \subset R^\infty$, we call

n the *length* of x. We denote by R_∞ the *bi-infinite direct sum* space over R. Elements of R_∞ have the form

$$x = (..., x_{-2}, x_{-1}, x_0.x_1, x_2, ...),$$

where $x_i \in R$ for all integers i, $x_k = 0$ for $|k|$ sufficient large, and . is a distinguished marker between x_0 and x_1. In fact, R_∞ can be viewed as a work tape of a machine. Suppose $h: R^m \mapsto R$ is a polynomial (or rational) function of degree d over R. Then h defines a polynomial (or rational) function $\widehat{h}: R_\infty \mapsto R$ on R_∞ of dimension m and degree d by letting $\widehat{h}(x) = h(x_1, ..., x_m)$ for each $x \in R_\infty$. Suppose $g_i: R^m \mapsto R,\ i = 1, ..., m$, are polynomial (or rational) functions of maximum degree d over R. Then the $g_i,\ i = 1, ..., m$, defines a polynomial (or rational) map on R_∞

$$\widehat{g}: R_\infty \mapsto R_\infty$$

of dimension m and degree d by letting $(\widehat{g}(x))_i = \widehat{g_i}(x),\ i = 1, ..., m$ and $(\widehat{g}(x))_i = x_i$ for $i < 1$ or $i > m$. The space R_∞ has natural *shift* operations, *shift left* σ_l and *shift right* σ_r, where

$$\sigma_l(x)_i = x_{i+1} \text{ and } \sigma_r(x)_i = x_{i-1}. \tag{4}$$

In order to define pseudo-division machine, we define the *state space* as $R_\infty \times \mathbb{Z}^{k_M}$, where k_M is a positive integer. We call R_∞ the *computation space* and $\mathbb{Z}^{k_M}$ the *counter space*. That is to say, our machine will have two work tapes. We relate the space R^∞ and the state space by defining maps $I_\infty: R^\infty \mapsto R_\infty \times \mathbb{Z}^{k_M}$ and $O_\infty: R_\infty \times \mathbb{Z}^{k_M} \mapsto R^\infty$ as follows.

$$I_\infty(x) = (..., 0, 0.x_1, ..., x_n, 0, 0, ...) \times (v_1, ..., v_{k_M}) \text{ for } x \in R^n, \tag{5}$$

where every $v_i,\ i = 1, ..., k_M$, is a linear function in n, or, if x belongs to a product space, is a linear function in the dimensions of all spaces in the product space, and

$$O_\infty((..., x_0.x_1, ...) \times (w_1, ..., w_{k_M})) = \begin{cases} \mathbf{0} \in R^0 & \text{if } w_1 = 0 \\ (x_1, ..., x_{w_1}) \in R^{w_1} & \text{otherwise.} \end{cases} \tag{6}$$

Definition 1. *A pseudo-division machine M over R consists of a finite directed connected graph with five types of nodes: input, computation, shift, branch and output. The unique input node has no incoming edges and only one outgoing edge. All other nodes have possibly several incoming edges. Computation and shift nodes have only one outgoing edge, branch nodes exactly two, Yes and No, and output nodes none.*

In addition, the machine has three spaces: an input space $\mathcal{I}_M = R^\infty$, state space $\mathcal{S}_M = R_\infty \times \mathbb{Z}^{k_M}$, and output space $\mathcal{O}_M = R^\infty$, where k_M is a positive

integer. Associated with each node of the graph are maps of these spaces and next node assignments.

I. *Associated with the input node is the map I_∞ defined by Eq. (5) and a unique next node β_1.*

II. *Each computation node η has an associated computation map, $g_\eta : \mathcal{S}_M \longmapsto \mathcal{S}_M$, and a unique next node β_η. The computation map g_η is defined as follows.*

$$\begin{aligned} g_\eta = (p_\eta \circ \pi_1, f_\eta) : R_\infty \times \mathbb{Z}^{k_M} &\longmapsto R_\infty \times \mathbb{Z}^{k_M} \\ y = (x, v) &\mapsto (p_\eta \circ \pi_1(y), f_\eta(y)) = (p_\eta(x), f_\eta(y)) \end{aligned}$$

where $p_\eta(x) : R_\infty \mapsto R_\infty$ is a polynomial (or rational) function on R_∞, f_η is a map from the state space to the counter space and $\pi_1 : \mathcal{S}_M \to R_\infty$ is the projection onto R_∞.

III. *Associated with each shift node η is a map $g_\eta \in \{\widehat{\sigma}_l, \widehat{\sigma}_r\}$ of the state space to itself and a unique next node β_η. The shift maps $\widehat{\sigma}_l$ and $\widehat{\sigma}_r$ are defined as follows. For $(x, v) \in \mathcal{S}_M$,*

$$\widehat{\sigma}_l(x, v) = (\sigma_l(x), v) \text{ and } \widehat{\sigma}_r(x, v) = (\sigma_r(x), v),$$

where σ_l and σ_r are defined by Eq. (4).

IV. *Each branch node η has an associated branching function $h_\eta : \mathcal{S}_M \longmapsto \mathbb{Z}$. The next node along the Yes outgoing edge, β_η^+, is associated with the condition $h_\eta(y) \geq 0$ and the next node along the No outgoing edge, β_η^-, with the condition $h_\eta(y) < 0$.*

V. *Finally, each output node η has an associated map O_∞ defined by Eq. (6).*

Remark 2. *The* dimension K_M *and* degree D_M *of M are, respectively, the maximum of the dimensions and the degrees of all maps associated with its computation and branch nodes.*
Remark 3. *We can test if $h(y) = 0$ by a subroutine that uses the comparisons $h(y) \geq 0$ and $-h(y) \geq 0$ (see Fig. (2))*
Remark 4. *For convenience, we assume all branch nodes are* standard*, that is, that $h_\eta(y) = v_{i_\eta}$ for some $1 \leq i_\eta \leq k_M$ for each branch node η and $y = (x, v) \in \mathcal{S}_M$. It is easy to see that this is without loss of generality of our theory.*
Remark 5. *It is sometimes convenient to define computations that involve non-positive coordinates. And this can be easily achieved by combining some shift operations with computations.*

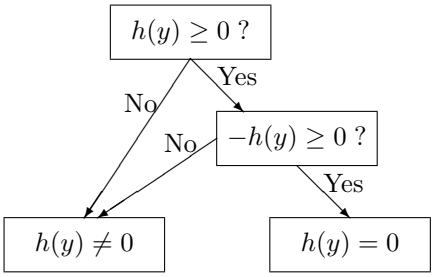

Figure 2. Testing for Equality

Remark 6. *It is often convenient to stipulate that a machine M over R has a unique output node. We then can identify the nodes of the machine with the labels $1, \ldots, N$ where 1 denotes the input node and N the output node.*
Remark 7. *It's not difficult to see that a classical Turing machine is a pseudo-division machine over $\mathbb{Z}_2$.*

4 Relations to the BSS Machines

In this section, we will demonstrate by two examples that the BSS machine can be simulated by the pseudo-division machine defined in last section. For all the definitions and notations related to the BSS machine, please refer to [1].
Example 1. *Let R be $\mathbb{R}$ or $\mathbb{Q}$, that is, the field of the real numbers or rational numbers. Let $M^{(1)}$ be a BSS machine in its* normal form *over R. We construct a pseudo-division machine $M^{(2)}$ over R as follows.*

I. *The input spaces are both R^∞. Let $k_{M^{(2)}} = 1$. For any $x \in R^n \subseteq R^\infty$, the input map of $M^{(1)}$ is $I^{(1)}_\infty(x) = (..., 0, 0, \overbrace{1, ..., 1}^{n}.x_1, ..., x_n, 0, 0, ...)$. Let the input map of $M^{(2)}$ be $I^{(2)}_\infty(x) = ((..., 0.x_1, ..., x_n, 0, 0, ...), n)$. Furthermore, let the input node of $M^{(2)}$ be followed immediately by a computation node η with $g_\eta((..., 0.x_1, ..., x_n, 0, 0, ...), n) = ((..., 0, 0, \overbrace{1, ..., 1}^{n}.x_1, ..., x_n, 0, 0, ...), n)$.*

II. *For each computation node $\eta^{(1)}$ in $M^{(1)}$, suppose $g_{\eta^{(1)}}$ be its associated polynomial (or rational) map, construct a computation node $\eta^{(2)}$ in $M^{(2)}$ with associated map $g_{\eta^{(2)}}$ defined as $g_{\eta^{(2)}}(x, v) = (g_{\eta^{(1)}}(x), v)$.*

III. *For each shift node in $M^{(1)}$, construct a shift node in $M^{(2)}$ with the same*

shift operation.

IV. *For each branch node $\eta^{(1)}$ in $M^{(1)}$, we construct a computation node $\eta_1^{(2)}$ and a branch node $\eta_2^{(2)}$ in $M^{(2)}$ as follows. Suppose $h_{\eta^{(1)}}$, a polynomial map, be the branching function associated with $\eta^{(1)}$, we define the associated map with $\eta_1^{(2)}$ to be $g_{\eta_1^{(2)}}(x,v) = (x, \mathrm{sign}(h_{\eta^{(1)}}(x)))$, where, for any $a \in R$, $\mathrm{sign}(a)$ is 1 if $a > 0$; 0 if $a = 0$; -1 if $a < 0$. And we define the next node to be $\eta_2^{(2)}$. As to the branch node $\eta_2^{(2)}$, let the associated branching function $h_{\eta_2^{(2)}}(y) = v_1$.*

V. *For the output node $N^{(1)}$ in $M^{(1)}$, we construct a computation node $\eta_{N^{(2)}}$ and the output node $N^{(2)}$ in $M^{(2)}$ as follows. $g_{\eta_{N^{(2)}}} : (x,v) \mapsto (x, f(x))$, where $f(x) = \min_{i\geq 0}\{x_{-i} = 0\}$. And the next node is $N^{(2)}$. The output map of $N^{(2)}$ is defined by Eq. (6).*

It's easy to see that $M^{(2)}$ simulates $M^{(1)}$.

Remark 8. *If R is an ordered ring and $M^{(1)}$ a BSS machine over R with branching condition $h(x) < 0$, we can construct a pseudo-division machine over R which simulates $M^{(1)}$ by doing the same procedure as above and letting* $\mathrm{sign}(a) : R \mapsto \mathbb{Z}$ *to be defined as above.*

Example 2. *Let R be $\mathbb{C}$ or $\mathbb{Z}_p$ where p is a positive integer , that is, the field of the complex numbers or the ring of integers modulo p. Let $M^{(1)}$ be a BSS machine over R with the branching condition $h(x) = 0$, We can construct a pseudo-division machine $M^{(2)}$ over R to simulate $M^{(1)}$ by almost the same procedure as in last example except replacing* $\mathrm{sign}(a)$ *by $h^B : R \mapsto \mathbb{Z}$, where $h^B(a)$ is 0 if $a = 0$; 1 otherwise.*

For $m > 0$ let $\mathcal{S}_m = \{((x_{-m+1}, ..., x_0, x_1, ..., x_m), v) \mid x_i \in R \text{ and } v \in \mathbb{Z}^{k_M}\} = R^{2m} \times \mathbb{Z}^{k_M}$. Naturally, we have the injections $\widetilde{i} : \mathcal{S}_m \longmapsto \mathcal{S}$ and $\widetilde{i_1} : R^{2m} \longmapsto R_\infty$, and the projections $\widetilde{\pi} : \mathcal{S} \longmapsto \mathcal{S}_m$, $\widetilde{\pi_1} : \mathcal{S}_m \longmapsto R^{2m}$ and $\widetilde{\pi_1'} : R_\infty \longmapsto R^{2m}$, defined, respectively, by

$$\widetilde{i}((x_{-m+1}, ..., x_0, x_1, ..., x_m), v) = ((..., 0, x_{-m+1}, ..., x_0.x_1, ..., x_m, 0, ...), v),$$

$$\widetilde{i_1}((x_{-m+1}, ..., x_0, x_1, ..., x_m)) = (..., 0, x_{-m+1}, ..., x_0.x_1, ..., x_m, 0, ...),$$

$$\widetilde{\pi}(x, v) = ((x_{-m+1}, ..., x_0, x_1, ..., x_m), v),$$

$$\widetilde{\pi_1}((x_{-m+1}, ..., x_0, x_1, ..., x_m), v) = (x_{-m+1}, ..., x_0, x_1, ..., x_m),$$

and $\widetilde{\pi_1'}(x) = (x_{-m+1}, ..., x_0, x_1, ..., x_m)$. Then, we associate with the input map I_∞ the *modified input map* $\widetilde{I_\infty} = \widetilde{\pi} \circ I_\infty : \mathcal{I}_M \longmapsto \mathcal{S}_m$, and with each

$g_\eta = (p_\eta \circ \pi_1, f_\eta)$ the *modified computation map* $\widetilde{g_\eta} = (\widetilde{p_\eta} \circ \widetilde{\pi_1}, \widetilde{f_\eta}) : \mathcal{S}_m \longmapsto \mathcal{S}_m$, where $\widetilde{p_\eta} = \widetilde{\pi_1'} \circ p_\eta \circ \widetilde{i_1} : R^{2m} \longmapsto R^{2m}$, $\widetilde{f_\eta} = f_\eta \circ \widetilde{i} : \mathcal{S}_m \longmapsto \mathbb{Z}^{k_M}$.

Now suppose γ is a computation path $\eta^0, \eta^1, ..., \eta^k, ...$ At each step k in the path γ, M evaluates a map $\widetilde{P}_{\gamma(k)} : \mathcal{I}_M \longmapsto R^{2m}$ defined by $\widetilde{P}_{\gamma(k)} = \widetilde{p_{\eta^k}} \circ \cdots \circ \widetilde{p_{\eta^0}} \circ \widetilde{\pi_1} \circ \widetilde{I_\infty}$, and a map $\widetilde{F}_{\gamma(k)} : \mathcal{I}_M \longmapsto \mathbb{Z}^{k_M}$ defined by $\widetilde{F}_{\gamma(k)} = \widetilde{f_{\eta^k}} \circ \widetilde{g_{\eta^{k-1}}} \cdots \circ \widetilde{g_{\eta^0}} \circ \widetilde{I_\infty}$. Restricted to R^n, where $m \geq \max(K_M, n) + k$, $\widetilde{P}_{\gamma(k)}$ is an ordinary polynomial function.

We also associate with the output map O_∞ the *modified output map* $\widetilde{O_\infty} = O_\infty \circ \widetilde{i} : \mathcal{S}_m \longmapsto R^\infty$. Let $\widetilde{G}_{\gamma(k)} = (\widetilde{P}_{\gamma(k)}, \widetilde{F}_{\gamma(k)})$, then, for $\gamma \in \Gamma_T$, $\Phi_M|_{\mathcal{V}_\gamma^n} = \widetilde{O_\infty} \circ \widetilde{G}_{\gamma(T-1)}$. It's easy to see that the input-output map Φ_M, restricted to the n-dimensional component of the path set $\mathcal{V}_\gamma$, is a polynomial map composed with the output map $\widetilde{O_\infty}$.

Likewise, suppose η_k is a standard branch node (see Remark 4), M evaluates the step-k branching function $\widetilde{F}^B_{\gamma(k)} : \mathcal{I}_M \longmapsto \mathbb{Z}$ defined by $\widetilde{F}^B_{\gamma(k)} = \Pi_{j_\eta} \circ \widetilde{F}_{\gamma(k)}$, where $\Pi_{j_\eta} : \mathbb{Z}^{k_M} \longmapsto \mathbb{Z}$ is the projection onto the j_ηth $(1 \leq j_\eta \leq k_M)$ coordinate.

We note that $\mathcal{V}_{\gamma(k)}$ is determined by the branching conditions along the path $\gamma(k)$. Suppose m is large enough, for example $m = \max(K_M, n) + k$, let

$$\mathtt{L}_{\gamma(k)} = \{\widetilde{F}^B_{\gamma(k')} | k' < k,\ k' \text{ a branch step in } \gamma, \text{ and } \eta^{k'+1} = \beta^-(\eta^{k'})\}$$
$$\mathtt{R}_{\gamma(k)} = \{\widetilde{F}^B_{\gamma(k')} | k' < k,\ k' \text{ a branch step in } \gamma, \text{ and } \eta^{k'+1} = \beta^+(\eta^{k'})\}$$

Theorem 1. *For any pseudo-division machine M, the following properties hold.*

(1) *For a computation path $\gamma(k)$,*

$$\mathcal{V}^n_{\gamma(k)} = \{x \in R^n | f(x) < 0, g(x) \geq 0, f \in \mathtt{L}_{\gamma(k)}, g \in \mathtt{R}_{\gamma(k)}\}.$$

(2) *For $\gamma \in \Gamma_T$, $\Phi_M|_{\mathcal{V}_\gamma^n} = \widetilde{O_\infty} \circ \widetilde{G}_{\gamma(T-1)} = \widetilde{O_\infty} \circ (\widetilde{P}_{\gamma(T-1)}, \widetilde{F}_{\gamma(T-1)})$. Because $\widetilde{P}_{\gamma(T-1)}$, restricted to the n-dimensional component of the path set $\mathcal{V}_\gamma$, is a polynomial map, the input-output map Φ_M, restricted to the n-dimensional component of the path set $\mathcal{V}_\gamma$, is a polynomial map composed with the output map $\widetilde{O_\infty}$.*

Remark 9. *For any map $f \in \mathtt{L}_{\gamma(k)} \bigcup \mathtt{R}_{\gamma(k)}$, if it can be expressed as $f = f_1 \circ f_2$, where $f_2 : \mathcal{I}_M \mapsto R$ is a polynomial function and $f_1 : R \mapsto \mathbb{Z}$ satisfies that $f_2(x) <_R 0$ if and only if $f(x) < 0$, then $\mathcal{V}^n_{\gamma(k)}$ is a basic semi-algebraic set. Here, $<_R$ means the ordering in R. So, from Examples 1 and 2, for a pseudo-division machine simulating a BSS machine, we have the same* path decomposition theorem *as in* [1].

5 An Example

In this section, we will give a formal, detailed construction of a pseudo-division machine that performs pseudo-division of two polynomials like the `prem` algorithm in Section 2. And then, we point out some essential differences between our machine and the BSS machine.

Let $\mathbb{D}$ be a UFD, A and B be two polynomials in $\mathbb{D}[u_1, ..., u_s]$ and u_k a fixed variable. We pseudo-divide A by B with respect to u_k. Let $R = \mathbb{D}[u_1, ..., u_{k-1}, u_{k+1}, ..., u_s]$, $k_M = 5$ and $A = (a_1, ..., a_n)$, $B = (b_1, ..., b_m)$, where $a_i, b_j \in R$.

Input $(A, B) = (a_1, ..., a_n, b_1, ..., b_m) \in R^\infty$, let $t = n - m$ and

$$I_\infty(A, B) = (x, v) = ((..., 0.a_1, ..., a_n, b_1, ..., b_m, 0, ...), (n, m, t, n, m - 1)).$$

Then, by a sequence of machine operations, we get a sequence of states:

$$\begin{gathered}
\vdots \\
((...0.a_1, 0, a_1, ..., a_n, b_1, ..., b_m, 0, ...), (n, m, t, n, m - 1)) \\
\vdots \\
((...0, a_1, ..., a_n.a_1, b_1, b_1, ..., b_m, 0, ...), (n, m, t, 0, m - 1)) \\
((...0, a_2, ..., a_n.a_1, b_1, b_1, ..., b_m, 0, ...), (n, m, t, n - 1, m - 1)) \\
\vdots \\
((...0.a_1, b_1, a_2b_1, ..., a_nb_1, b_1, ..., b_m, 0, ...), (n, m, t, 0, m - 1)) \\
((...0.a_1, b_1, a_2b_1, ..., a_nb_1, b_1, ..., b_m, 0, ...), (n, m, t, n, m - 1)) \\
\vdots \\
((...0, a_2b_1, ..., a_nb_1, b_1.a_1, b_2, b_2..., b_m, 0, ...), (n, m, t, 0, m - 2)) \\
((...0, a_2b_1, ..., a_nb_1, b_1.a_1, b_2, b_2..., b_m, 0, ...), (n, m, t, n - 1, m - 2)) \\
\vdots \\
((...0, a_2b_1 - a_1b_2.a_1, b_2, a_3b_1, ..., a_nb_1, b_1, b_2..., b_m, 0, ...), (n, m, t, 0, m - 2)) \\
\vdots \\
((..., a_2b_1 - a_1b_2, ..., a_mb_1 - a_1b_m.a_1, b_m, a_{m+1}b_1, ..., a_nb_1, b_1, ..., b_m, ...), (n, m, t, 0, 0)) \\
\vdots \\
((...0.a_1, b_m, a_2b_1 - a_1b_2, ..., a_mb_1 - a_1b_m, a_{m+1}b_1, ..., a_nb_1, b_1, ..., b_m, ...), (n, m, t, 0, 0)) \\
((...0.a_2^{(1)}, 0, a_2^{(1)}, ..., a_n^{(1)}, b_1, ..., b_m, ...), (n - 1, m, t - 1, n - 1, m - 1)),
\end{gathered}$$

where

$$a_i^{(1)} = \begin{cases} a_ib_1 - a_1b_i & \text{if } 2 \le i \le m \\ a_ib_1 & \text{if } m < i \le n. \end{cases}$$

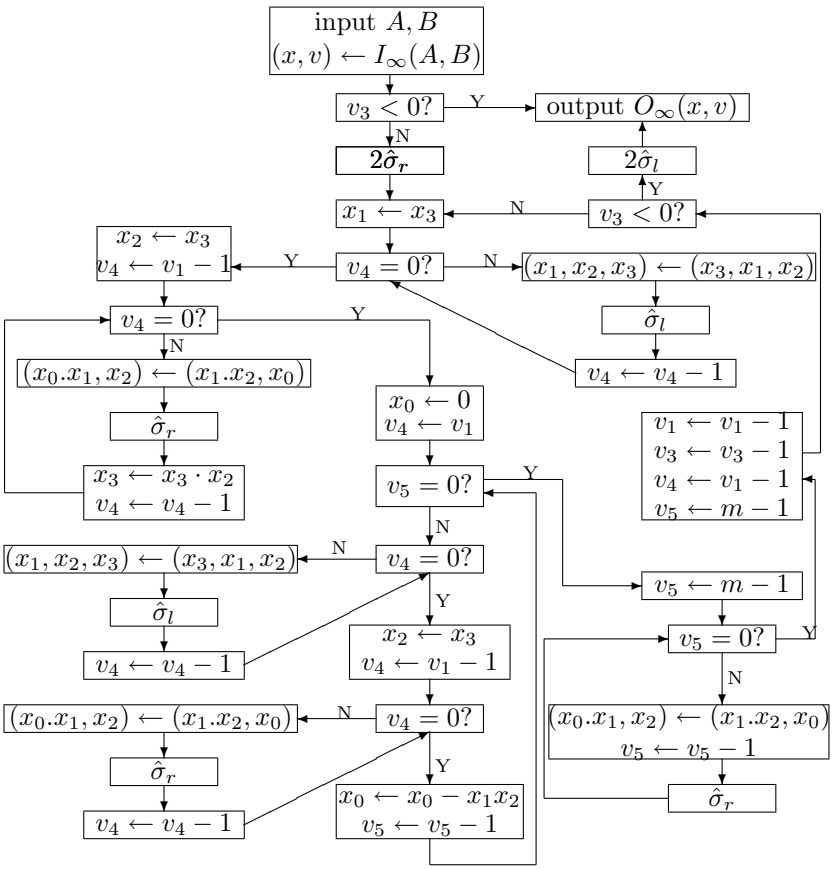

Figure 3. The machine doing pseudo-division

Now proceeding as before, the machine produces in turn the states:

$$((...0.a_3^{(2)}, 0, a_3^{(2)}, ..., a_n^{(2)}, b_1, b_2..., b_m, 0, ...), (n-2, m, t-2, n-2, m-1))$$
$$\vdots$$
$$((...0.a_{t+2}^{(t+1)}, 0, a_{t+2}^{(t+1)}, ..., a_n^{(t+1)}, b_1, ..., b_m, 0, ...), (m-1, m, -1, m-1, m-1)),$$

where

$$a_i^{(j)} = \begin{cases} a_i^{(j-1)} b_1 - a_1^{(j-1)} b_i & \text{if } i \leq m \\ a_i^{(j-1)} b_1 & \text{if } m < i \leq n. \end{cases}$$

And finally, after two shift left operations, output the first $m-1$ elements of the computation space, that is $(a_{n-m+2}^{(n-m+1)}, ..., a_n^{(n-m+1)})$. See Fig.3.

There are some essential differences between our machine and the BSS machine. First of all, we separate the computations on the ring R from those on $\mathbb{Z}$. When the characteristic of R is $p > 0$, it seems difficult and unnatural for a BSS machine to manage computations over R while the computations in our machine are natural and easy to understand. Secondly, the branching conditions are now tested in $\mathbb{Z}$ other than R. This is necessary in most computations over such rings as rings of polynomials in several variables. For pseudo-division, in any BSS machine one has to simulate the test of branching condition $n - m < 0$ since it cannot be checked under the ordering of $R = \mathbb{D}[u_1, ..., u_{k-1}, u_{k+1}, ..., u_s]$.

Acknowledgments

The authors are indebted to Professor H. Ganzinger for helpful discussion and to NKBRSF-(G1998030600) for support. Also, one of the authors, Xia, thanks the Max-Planck-Institut für Informatik for hospitality.

References

1. L. Blum, F. Cucker, M. Shub and S. Smale, *Complexity and Real Computation*, Springer-Verlag, New York, 1997.
2. L. Blum, M. Shub and S. Smale, *On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines*, Bulletin (New Series) of the American Mathematical Society, 1989, 21(1): 1-46.
3. D. S. Bridges, *Computability*, New York: Springer-Verlag, 1997.
4. B. Buchberger, Gröbner bases: An algorithmic method in polynomial ideal theory, In *Multidimensional Systems Theory*, (Edited by N.K. Bose), pp. 184-232, Reidel, Dordrecht, 1985.
5. S. C. Chou, *Mechanical geometry theorem proving*, Reidel, Dordrecht, 1988.
6. M. Garey and D. Johnson, *Computers and Intractabilitiy: A Guide to the Theory of NP-Completeness*, Freedman, 1979.
7. D.M. Wang, *Elimination Methods*, Springer-Verlag, New York, 2001.
8. W. T. Wu, On the decision problem and the mechanization of theorem-proving in elementary geometry, *Sci. Sinica* **21**: 159-172 1978.
9. W.T. Wu, *Mechanical theorem proving in geometries: Basic principles* (translated by X. Jin and D. Wang), Springer, New York, 1994.