Transactions of the Institute for Informatics and Automation Problems of NAS RA.
Collected reports of participants of JAF-29.

WEAK ARITHMETIC FOR FUNCTIONS OF EXPONENTIAL SIZE[1]

Henri-Alex Esbelin

It is known that substitution of polynomials to variables does not modify the complexity of relations of various sublinspace classes, the most important of which are the class of rudimentary relations, the class of rudimentary relations with counting, and LINSPACE. Substitution of functions of exponential size is not allowed (in general). However, it is possible to develop some arithmetic for such functions in this framework.

§1. INTRODUCTION

This paper is mainly devoted to the problem of comparing functions of exponential size using linear space, that is using operations (to be precised) over functions of polynomial size. It is a bottelneck in modular arithmetic: D. Knuth wrote „... methods are available to test overflow, ..., but they are so complicated that they nullify the advantage of modular arithmetic" (p291 in [Kn98]). The methods of this paper were initiated by A. Woods in [Wo86]. For other applications, see [Al01] and [BEPP99].

Definition 1.1 Let us denote by $\mathfrak{R}$ (resp. $\mathfrak{R}^\sharp$) the smallest class of relations over integers containing the graphs of addition and multiplication (seen as ternary relations) and closed under boolean operations ($\neg, \wedge, \vee, \rightarrow$), explicit transformations (i.e. adding, cancelling, renaming, permuting and confusing variables), variable bounded quantifications (i.e. $(\forall x)_{<y}...$ meaning $\exists x(x < y \rightarrow ...)$ and $(\exists x)_{<y}...$ meaning $\forall x(x < y \wedge ...)$), (resp. and closed under counting operation, i.e. if R lies in $\mathfrak{R}^\sharp$, then $z = \sharp\{i < x; R(i, x_1, ..., x_r)\}$ lies also in $\mathfrak{R}^\sharp$). Relations in $\mathfrak{R}$ are called rudimentary.

Let us consider here and subsequently two functions with non negative integer variables and values, $f_1(x_1, ..., x_r)$ and $f_2(x_1, ..., x_r)$ less than a^{x_1}, where a is a positive integer and such that the relations $z = f_i(x_1, ..., x_r) \mod (p)$ are rudimentary; is the relation $f_1(x_1, ..., x_r) < f_2(x_1, ..., x_r)$ rudimentary? The main result is the following:

[1]This work has been supported by Intas 2000-447

Theorem 1.2 Suppose that $f_i(\vec{x}) \leq a^{x_1}$ for $i = 1, 2$, where a is a positive integer. Suppose that the graphs of the functions $f_i(\vec{x})$ mod (p) are in RUD^1. Then the relation $f_1(\vec{x}) < f_2(\vec{x})$ lies in RUD^1.

Notice that the rudimentarity of the graphs of the f_i functions does not ensure the rudimentarity of the relations $z = f_i(x_1, ..., x_r)$ mod (p). Thanks to Hesse's theorem (see [He01]), simple exemples of functions satisfying the hypothesis of theorem 1.2 are $f(x) = b^x$, with positive constante b.

In the first section we give an algorithm for solving the problem. The second section is devoted to its complexity. The third section provides some applications. This paper is not selfcontained: as for prerequisites, the reader is expected to be familliar with the complexity in the framework of classes of relations defined with bounded quantifications (for an extended overview, see [EM98]).

§2. ALGORITHM

The first property we use reduces the problem to comparing parity of derived functions. Let us denote by a mod (b) the remainder in the division of a by b.

Property 2.1 Let X and Y be integers in $[0; \Pi)$, where Π is an odd positive integer. Then $X < Y$ iff $\left(X < \frac{\Pi}{2} \text{ and } \frac{\Pi}{2} < Y\right)$ or, in the other cases, $(X - Y)$ mod (Π) and $X - Y$ are of opposite parity. Moreover, $X < \frac{\Pi}{2}$ iff $2X$ mod (Π) is even.

The easy proof is left to the reader.

In order to make use of modular arithmetic, we choose Π as one of the $\prod_{i=1}^{i=n} p_i = \Pi_n$, where p_i is the $i+1$-th prime number (hence excluding $p_0 = 2$ to get an odd Π). We now explain how to get the parity of Z mod (Π_n) from the remainders of Z modulo the p_i. For any integer X and any positive integer n, let us denote $CRR_n(X) = (X \text{ mod } (p_j))_{1 \leq j \leq n}$.

In the other way, let us consider a given $n-$tuple $(\alpha_1, ..., \alpha_n)$, where α_j are non negative integers less than $p_j - 1$. From the Chinese Remainders Theorem (CRT for short), there is a unique integer X in $[0; \Pi_n)$ such that $CRR_n(X) = (\alpha_1, ..., \alpha_n)$; we denote it by $|(\alpha_1, ..., \alpha_n)|_{\Pi_n}$. As a consequence, we have $|CRR_n(X)|_{\Pi_n} = X$ mod (Π_n). From the CRT again, we know that $CRR_n(X) = (\alpha_1, ..., \alpha_n)$ iff Π_n divides $X - |(\alpha_1, ..., \alpha_n)|_{\Pi_n}$. The *à la Lagrange* solution of $CRR_n(X) = (\alpha_1, ..., \alpha_n)$ leads to the following:

Definition 2.2 Define $\bar{\alpha}_i$ the non negative integer less than p_i such that $\left(\bar{\alpha}_i \frac{\Pi_n}{p_i} \equiv \alpha_i\right)$ mod (p_i).

Denote $\sum_{i=1}^{i=n} \bar{\alpha}_i \frac{\Pi_n}{p_i}$ as $TRC_n(\alpha_1, ..., \alpha_n)$. Then $CRR_n(TRC_n(\alpha_1, ..., \alpha_n)) = (\alpha_1, ..., \alpha_n)$.

Hence $\frac{TRC_n(CRR_n(X)) - X \bmod (\Pi_n)}{\Pi_n}$ is a positive integer. Let us denote it by $\rho_n(X)$.

It is easy to verify that $\rho_n(X)$ is less than n. Hence, if $X < \Pi_l$, we can compute $\rho_l(X)$ performing all the operations modulo p_{l+1}, which is greater than l:

Property 2.3 if $X < \Pi_l$, then $\rho_l(X)$ is equal to

$$((TRC_l(CRR_l(X)) - X) \bmod (p_{l+1}) \times (\Pi_l)^{-1} \bmod (p_{l+1})) \bmod (p_{l+1})$$

where $(\Pi_l)^{-1} \bmod (p_{l+1})$ is the positive integer β less than p_{l+1} such that $(\beta \Pi_l \equiv 1) \bmod (p_{l+1})$.

We are now able to present an algorithm comparing f_1 and f_2 from their CRR representation, provided $f_i(\vec{x}) < \Pi_l$; it uses as a subalgorithm the computation of $(Z \bmod (\Pi_l)) \bmod (2)$ from $CRR_l(Z)$ which is postponed.

Algorithm 2.4 Compute $(2f_1(\vec{x}) \bmod (\Pi_l)) \bmod (2)$ and $(2f_2(\vec{x}) \bmod (\Pi_l)) \bmod (2)$: if they are different, conclude following 2.1.
if not, compute $((f_1(\vec{x}) - f_2(\vec{x})) \bmod (\Pi_l)) \bmod (2)$ and $(f_1(\vec{x}) - f_2(\vec{x})) \bmod (2)$, conclude following 2.1.

And now the subalgorithm:

Algorithm 2.5 For $1 \leq i \leq l$ get $\alpha_i = Z \bmod (p_i)$ and solve the equation $\bar{\alpha}_i \frac{\Pi_l}{p_i} = \alpha_i \bmod (p_i)$ with $\bar{\alpha}_i$ unknown in $[0; p_i)$;

compute $(TRC_l(CRR_l(Z))) \bmod (p_{l+1})$ as $\left(\sum_{i=1}^{i=l} \bar{\alpha}_i \frac{\Pi_l}{p_i}\right) \bmod (p_{l+1})$

compute $\rho_l(Z)$ as $((TRC_l(CRR_l(Z) - Z) \bmod (p_{l+1}) \times (\Pi_l)^{-1} \bmod (p_{l+1})) \bmod (p_{l+1})$

compute $(TRC_l(CRR_l(Z))) \bmod (2)$ as $\left(\sum_{i=1}^{i=l} \bar{\alpha}_i \frac{\Pi_l}{p_i}\right) \bmod (2)$

compute $(Z \bmod (\Pi_l)) \bmod (2)$ as

$$((TRC_l(CRR_l(Z)) \bmod (2) - \rho_l(Z) \bmod (2) \times \Pi_l \bmod (2)) \bmod (2)$$

§3. COMPLEXITY

Computations in the previous algorithm are mainly iterated sums and iterated products (namely the Π_l) modulo 2 and modulo p_{l+1}, the length of which are less than l. These are bounded recursions, motivating the introduction of the least class of the Grzegorczyck hierarchy.

Definition 3.1 Let us denote $\vec{x} = (x_1, ..., x_r)$. We denote $\mathfrak{E}^0$ is the smallest class of functions containing projections, constants and the successor function and closed under composition and bounded recursion $(f(\vec{x}, 0) = g(\vec{x}), f(\vec{x}, i+1) = h(\vec{x}, i, f(\vec{x}, i)), f(\vec{x}) < j(\vec{x}, i))$. We denote $\mathfrak{E}^0_*$ the class of relations which characteristic functions are in $\mathfrak{E}^0$.

It is easy to prove a weakened version of the theorem 1.2, where $\mathfrak{E}^0_*$ take place of $\mathfrak{N}^\sharp$. Indeed, from Chebyschev's theorem, a constant number A exists such that $\Pi_n > 2^{An\log(n)}$. Hence $f_i(\vec{x}) < \Pi_{x_1}$ for all but a finite number of values of x_1. With the choice $l = x_1$, the computations in the previous algorithms are in $\mathfrak{E}^0_*$. We give now the main ideas of the proof of the complete result.

Proof of the main theorem. It comes from the two following properties of $RUD^\sharp$:

Theorem 3.2 Let $f(x_0, \vec{x})$ define a function, the graph of which is in $RUD^\sharp$. Suppose that there exists a polynomial function ϕ with positive integers as coefficients, such that $f(x_0, \vec{x}) \leq \phi(x_0, \vec{x})$. Then $\sum_{i=0}^{i=x_0} f(i, \vec{x})$ defines a function, the graph of which is in $RUD^\sharp$.

Proof: From the CRT, the equality $z = \sum_{i=0}^{i=x_0} f(i, \vec{x})$ is equivalent to:

$$(z \leq (1+x_0)\phi(x_0, \vec{x})) \wedge (\forall c)_{\leq 1+\lfloor log_2((1+x_0)\phi(x_0,\vec{x}))\rfloor} \left(z \equiv \sum_{i=0}^{i=x_0} f(i, \vec{x}) \right) \mod (c)$$

Notice now the equality $\sum_{i=0}^{i=x_0} (f(i, \vec{x}) \mod (c)) = \sum_{h=0}^{h=c-1} h \times \sharp\{i \leq x_0; (f(i, \vec{x}) \equiv h) \mod (c)\}$. This last summation is in $RUD^\sharp$ (see lemma 4.5 in [EM98]), which completes the proof.

Theorem 3.3 Let p be a prime number. Let a_i define a sequence of non negative integers, such that the ternary relation $z \equiv a_i \mod (p)$ is in $RUD^\sharp$. Then the ternary relation $\left(z \equiv \prod_{i=0}^{i=n} a_i \right) \mod (p)$ is in $RUD^\sharp$.

Proof: Thanks to Hesse's theorem, a primitive root g for p is computed in $RUD^\sharp$ using the formula $(\forall n)_{<p-2} \left(g^{n+1} \not\equiv 1 \mod (p) \right)$ and the discrete logarithm $l(a)$ of any a is computed in $RUD^\sharp$ from its definition $\left(g^{l(a)} \equiv a \right) \mod (p)$. The relation $\left(z \equiv \prod_{i=0}^{i=n} a_i \right) \mod (p)$ is equivalent to $l(z \mod (p)) = \sum_{i=0}^{i=n} l(a_i)$, hence is in $RUD^\sharp$, which completes the proof.

§4. CONCLUSION

Using theorem 1.2, it is possible to solve arithmetical problems concerning functions of

exponential size from their CRR representations. Let us recall the *à la Newton* solution of $CRR_n(X) = (\alpha_1, ..., \alpha_n)$, which leads to the following *mixed radix system* (MRS for short):

Definition 4.1 For any non negative integer X, there exists a unique sequence of non negative integers $(c_i(X))_{i\in N}$ with finite support such that $X = \sum_{i=0}^{+\infty} c_i(X)\Pi_i$ and $c_i(X) \leq p_{i+1} - 1$ for all i. For non zero X, let $\lambda(X)$ be $Inf\{k; X < \Pi_k\} = Sup\{k; c_k \neq 0\} + 1$ the length of the MRS representation of X.

As an easy consequence, $l = \lambda(X)$ iff $\Pi_{l-1} \leq X < \Pi_l$. The usual algorithm for computing $(c_i(X))_{i\in N}$ from $CRR_l(X)$ when $X < \Pi_l$, is the so-called *mixed radix conversion* (see [Kn98]):

Algorithm 4.2 For $0 \leq i \leq l-1$, let $T[0,i] = X \bmod(p_{i+1})$
For $1 \leq k \leq l-1$, for $k \leq i \leq n+1$, let $T[k,i] = \dfrac{T[k-1,i] - T[k-1,k-1]}{p_k} \bmod(p_{i+1})$,
$T[k,k]$ gives $c_{k-1}(X)$.

This algorithm is of high space complexity because of the necessity of storing each row of the T table at time. Let f a function satisfying the hypothesis of 1.2. We introduce now another method, based on the modular arithmetic ideas. The first algorithm compute the length of the MRS representation of $f(\vec{x})$ from $CRR_{x_1}(f(\vec{x}))$:

Algorithm 4.3 For $0 \leq i \leq x_1$, compute $CRR_{x_1}(\Pi_i)$.

Use algorithm 2.4 to compare $f(\vec{x})$ with Π_i. The first i such that $f(\vec{x}) < \Pi_i$ is $\lambda(f(\vec{x}))$.

We abbreviate $\lambda(f(\vec{x}))$ to λ. Now it is possible to compute the digit of highest weight $c_{\lambda-1}(f(\vec{x}))$:

Algorithm 4.4 For $0 \leq k \leq p_\lambda - 1$, compute $CRR_{x_1}(k\Pi_{\lambda-1})$ and compare $k\Pi_{\lambda-1}$ with $f(\vec{x})$ using algorithm 2.4. The last k such that $k\Pi_{\lambda-1} \leq f(\vec{x})$ is $c_{\lambda-1}(f(\vec{x}))$.

As $c_n(X) = c_n(X \bmod (\Pi_{n+1}))$ and $CRR_n(X \bmod (\Pi_{n+1})) = CRR_n(X)$, it is easy to compute each $c_n(f(\vec{x}))$. It is straightforward to verify that all of these operations are possible in $RUD^\sharp$.

A theoretical consequence of 1.2 concerns the question of the collapsing of the first classes of the relational Grzegorczyk hierarchy defined in [Gr53]: replacing the successor function by the set of polynomial functions in the definition of $\mathfrak{E}^0$ leads to $\mathfrak{E}^2$; the equality between $\mathfrak{E}^0_*$ and $\mathfrak{E}^2_*$ is an open question. It is known that $RUD^\sharp = \mathfrak{E}^0_*$ implies $RUD^\sharp = \mathfrak{E}^2_*$. Theorem 1.2 emphasizes the similarity of the expessive power of $RUD^\sharp$ and $\mathfrak{E}^0_*$.

REFERENCES

[BEPP99] H. Brönninan, I. Emiris, V. Pan, and S. Pion, Sign determination in Residue Number System. *Theoret. Comput. Sci.*, Special Issue on Real Numbers and Computers 210(1), 1999, pp. 173-197.

[Al01] E. Allender, The Division Breakthroughs, in The Computational Complexity Column *EATACS Bull.*, 74, 2001, pp. 61-77.

[EM98] H.A. Esbelin, M. More, Rudimentary relations and primitive recursion: A toolbox, *Theoret. Comput. Sci.*, 193, 1975, pp. 129-148.

[Gr53] A. Grzegorczyk, Some Classes of Recursive Functions, *Rozprawy Matematyczne*, 4, 1953, pp. 1-46.

[He01] W. Hesse, Division is in uniform TC^0. In *ICALP 2001: Twenty-Eighth International Colloquium on Automata, Langages and Programming*, Lecture Notes in Computer Science 2076, Springer-Verlag, 2001, pp. 104-114.

[Kn98] D.E. Knuth, Seminumerical Algorithms, volume 2 of *The Art of Computer Programming*, Addison-Wesley, third edition, 1998.

[Wo86] A. Woods, Bounded Arithmetic Formulas and Turing Machines of Constant Alternation, In *Logic Colloquium'84, volume 120 of Studies in logic and the foundations of mathematics*, p 355-377 North Holland, 1986.

14 May 2005

LLAIC1, IUT Clermont 1, BP 86, 63172 Aubière Cedex, France

E-mail: esbelin@llaic.u-clermont1.fr