PARALLEL ARITHMETIC COMPUTATIONS

Joachim von zur Gathen
University of Toronto

Abstract

A survey of parallel algorithms for algebraic problems is presented.

1. Introduction

Motivation for research in parallel algorithms comes from two different directions. The main impetus is from the rapidly expanding technology of parallel computer architectures, such as VLSI, systolic, and vector processing, and large projects in fixed connection networks. On the other hand, the theory presents a plethora of interesting mathematical problems. As the appropriate framework and fundamental problems for sequential algorithms are becoming clear, we are challenged to obtain similar insights into parallel algorithms. Cook [1985] discusses the Boolean theory. We want to survey the algebraic aspects of this relatively young area of research. We first present a framework for these parallel algorithms, and then review the relevant literature on linear algebra and polynomial arithmetic.

There are two basically different approaches to what constitutes a "fast parallel algorithm". The first is to start from a good sequential algorithm and try to parallelize it with a near-optimal speed-up, i.e., try to achieve parallel time close to (sequential time)/(number of processors). This seems appropriate for the present technology, where in effect only a limited amount of parallelism is available (Jordan [1982]). The second approach is to attempt to make the parallel time as small as possible (here: polylogarithmic), allowing an almost arbitrary (here: polynomial) number of processors. We

Part of this work was done while the author was visiting Universität Zürich, and supported by Schweizerischer Nationalfonds, grant 2175-0.83, and by NSERC, grant 3-650-126-40.

Author's address: Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A4, Canada.

will only discuss this second "asymptotic" approach.

In the 1970's, two surprising results were that Gaussian elimination (Csanky [1976]) and the evaluation of multivariate polynomials (Hyafil [1979]) can be performed fast in parallel. However, a systematic study of the subject was only started around 1982. A basic goal is to understand how the solution of problems depends on the ground field; from this point of view, the most satisfactory methods do not depend on the field at all. In this survey, we exclude results from the active and important area of Boolean computations for arithmetic problems (say, over $\mathbf{Q}$ or finite fields), and similarly algorithms over $\mathbf{Q}$ that use integer division with remainder.

The paper is organized as follows. In Section 2, we introduce the two models of computation. *Arithmetic circuits*, using only field operations, are sufficient to compute functions like the determinant. For "combinatorial" functions like the rank of matrices, we also allow Boolean operations in *arithmetic networks*. These networks compute "piecewise rational functions" (Section 3) consisting of a rational function on each piece of a partition of the input space into algebraic subsets. The "degree" of these piecewise rational functions, in the sense of algebraic geometry, is studied in Section 4. The degree of rational functions gives lower bounds on the depth of arithmetic circuits: depth $\geq$ log(degree). Unexpectedly, the corresponding degree bound in Section 5 is exponentially weaker for arithmetic networks.

In Section 6, we discuss arithmetic networks as the input size and the ground field varies. For the circuit part, the Boolean notions of *uniformity* carry over nicely. We propose several notions of what it means to describe the arithmetic constants of a network family "uniformly". The observation that many algorithms (e.g. standard matrix multiplication) work over arbitrary fields is made precise in the new notion of *universality* (Section 7). Eberly's [1986] results on polynomial arithmetic (Section 11) show interesting trade-offs between depth, uniformity for the circuit part, uniformity for the constant part, and universality (characteristic zero vs. infinite vs. finite fields).

In Section 8, we define the arithmetic analogues of the Boolean complexity classes P (polynomial size), NC^k (depth $\log^k$(input size) and polynomial size), and $NC = \bigcup_{k \in \mathrm{N}} NC^k$, and of reductions. A test for the definitions is that the fundamental properties should carry over from the Boolean setting. A major surprise is that $P = NC^2$ when restricted to rational functions (Valiant, Skyum, Berkowitz & Rackoff [1983], Kaltofen [1986]).

In Sections 9 and 10 we review results on linear algebra. Most elementary problems fall into one of two complexity classes, $RANK \subseteq DET$, for which many natural problems are complete. A basic open question is whether one of the inclusions $NC^1 \subseteq RANK \subseteq DET \subseteq NC^2$ is proper.

Section 11 discusses arithmetic of polynomials. Many interesting problems can be solved in optimal depth. The apparently hopeless problem of computing large powers yields interesting insights: over large finite fields of small characteristic, the model of arithmetic circuits over the field is inappropriate; one should use arithmetic circuits over the prime field. In order to discuss Berlekamp's algorithms for factoring univariate polynomials over finite fields, we introduce a quantified version of "universality".

2. The model of computation

For arithmetic algorithms, the most basic model of computation are *arithmetic circuits* (or "straight-line programs"), using inputs, constants from the ground field F, and $+$, $-$, $*$, $/$. *Arithmetic networks* use these arithmetic operations and also Boolean inputs, constants, and operations. The interface is given by "sign" gates, which take an arithmetic input $a \in F$ and produce a Boolean value according to whether a is zero or not, and by "selection" gates, which produce the first or second of their two arithmetic inputs, according to the value of the one Boolean input.

As an example, the arithmetic network of Figure 2.1 decides whether a linear equation $x_1 t + x_2 = 0$ has a solution t, and computes one, if it exists. The two inputs x_1 and x_2 are supplied at the vertices v_1 and v_2. The Boolean value of the output vertex v_{10} is $\mathbf{T}$ (= true) if the equation has a solution, and $\mathbf{F}$ (= false) otherwise. If the value is $\mathbf{T}$, then the value of v_9 is $t = -x_2/x_1$ if $x_1 \neq 0$, and $t = x_2$ otherwise.

For precise definitions, we follow the general approach of Strassen [1972], with terminology borrowed from the language of Boolean circuits. (This general approach will allow a simple definition of reductions in Section 8.) We fix an arbitrary ground field F. (Most of what follows also works over more general rings, only division requires an extra treatment.) For later use, it is convenient to start with a set Ω (of operations). An operation $\omega \in \Omega$ will take arithmetic inputs from a field F and Boolean inputs from $\mathbf{B} = \{\mathbf{T}, \mathbf{F}\}$, and produce either an arithmetic value from F or a Boolean value from $\mathbf{B}$. Formally, also given is an arity function $\sigma = (\sigma_1, \sigma_2) : \Omega \to \mathbf{N} \times \mathbf{N}$ (ω has $\sigma_1(\omega)$ arithmetic and $\sigma_2(\omega)$ Boolean inputs), and a type function $\tau : \Omega \to \{F, \mathbf{B}\}$ (giving the type "arithmetic" or "Boolean" of the output of ω). An (Ω,σ,τ)-program is a tuple $(G, \lambda, \iota, \rho)$ where $G = (V, E)$ is a finite directed acyclic graph with vertex set V and edge set $E \subseteq V \times V$, $\lambda: V \to \Omega$ a labelling of the vertices, $\iota: V \to \{1, \ldots, card(V)\}$ an injective numbering of the vertices, and $\rho = (v_1, \ldots, v_k)$ a sequence of (output) vertices. Arities have to be respected, and ι has to be consistent with the topological order of G, so that the following has to hold.

> For any $v \in V$, there are exactly $\sigma_1(\lambda(v))$ in-edges (w, v) to v with $\tau(\lambda(w)) = F$, and exactly $\sigma_2(\lambda(v))$ in-edges (w, v) to v with $\tau(\lambda(w)) = \mathbf{B}$. For each such w, $\iota(w) < \iota(v)$.

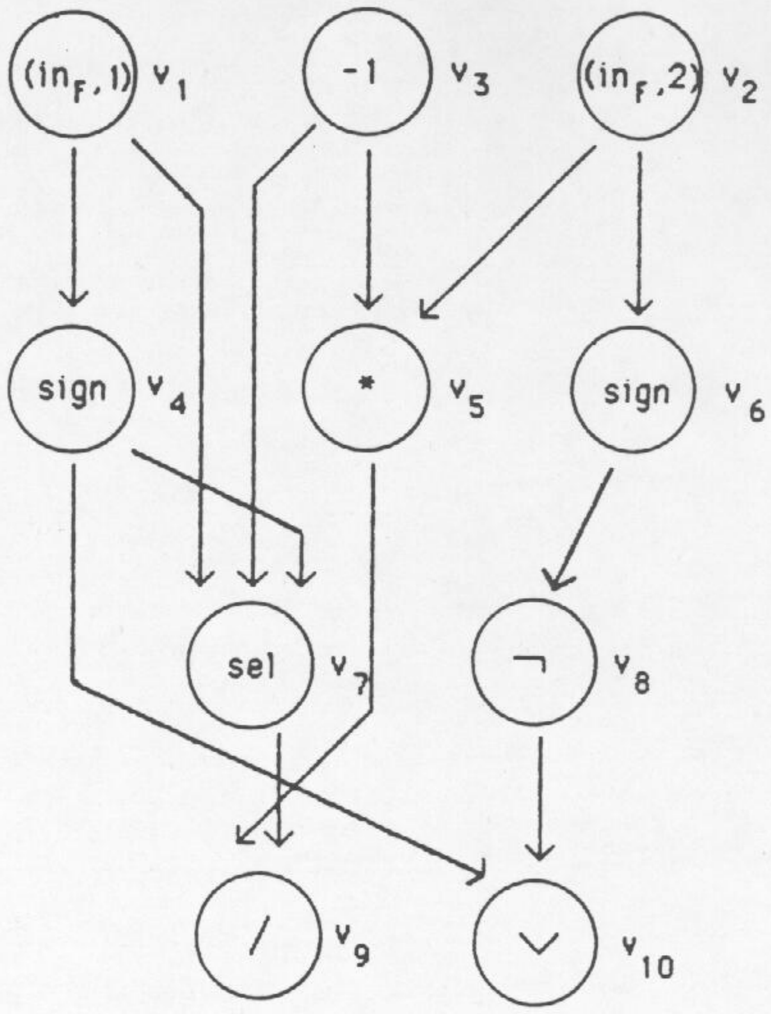

Figure 2.1

The simplest model has the following set Ω^0 of operations (and arities σ^0, types τ^0). Let F be a field, $n \in \mathbf{N}$, $N = \{1,...,n\}$, and

$$\Omega^0 = \Omega_F^0(n) = F \cup \{+,-,*,/\} \cup (\{in_F\} \times N),$$

$$\forall \omega \in \Omega^0 \;\; \sigma^0(\omega) = \sigma_F^0(n)(\omega) = \begin{cases} (0,0) & \text{if } \omega \in F \cup (\{in_F\} \times N), \\ (2,0) & \text{if } \omega \in \{+,-,*,/\}. \end{cases}$$

$$\tau^0(\omega) = \tau_F^0(n)(\omega) = F,$$

(The above description of Ω^0 is assumed to be a disjoint union; similar notational assumptions are made implicitly in the sequel. Whenever no confusion arises, we leave away the arguments F and n.)

Definition 2.1. An *arithmetic circuit over F* (with n inputs and ν outputs) is an $(\Omega_F^0(n), \sigma_F^0(n), \tau_F^0(n))$-program $(G, \lambda, \iota, \rho)$ with ρ of length $k = \nu$. $\square$

These arithmetic circuits are powerful enough to calculate rational functions such as the determinant of a matrix, but they cannot compute "Boolean functions" such as deciding whether a system of linear equations has a solution. We now describe the appropriate extension of the model.

Let F be a field, $n, m, \nu, \mu \in \mathbf{N}$, $N = \{1,\ldots,n\}$, $M = \{1,\ldots,m\}$, and

$$\Omega^1 = \Omega_F^1(n, m) = \Omega_F^0(n) \cup \mathbf{B} \cup \{\text{sign}, \text{sel}, \neg, \bigwedge, \bigvee\} \cup (\{in_{\mathbf{B}}\} \times M).$$

Furthermore, $\tau^1 = \tau_F^1(n, m)$ and $\sigma^1 = \sigma_F^1(n, m)$ are the extensions of τ^0 and σ^0, resp., with

$$\tau^1(\omega) = \begin{cases} F & \omega \in \Omega^0 \cup \{\text{sel}\}, \\ \mathbf{B} & \text{otherwise,} \end{cases}$$

$$\sigma^1(\omega) = \begin{cases} (1, 0) & \omega = \text{sign}, \\ (2, 1) & \omega = \text{sel}, \\ (0, 1) & \omega = \neg, \\ (0, 2) & \omega \in \{\bigvee, \bigwedge\}, \\ (0, 0) & \omega \in \mathbf{B} \cup (in_{\mathbf{B}} \times M). \end{cases}$$

Definition 2.2. An *arithmetic network over* F (with (n, m) inputs and (ν, μ) outputs) is an $(\Omega_F^1(n, m), \sigma_F^1(n, m), \tau_F^1(n, m))$-program $(G, \lambda, \iota, \rho)$, where ν output vertices in ρ have arithmetic type F, and μ vertices in ρ have type $\mathbf{B}$. $\square$

In particular, the length of ρ is $k = \nu + \mu$.

Remark 2.3. We can actually simulate the Boolean part of an arithmetic network by arithmetic operations, if we allow e.g. the operation ω^- of conditional inverse with

$$\omega^-(x) = \begin{cases} x^{-1} & \text{if } x \neq 0, \\ 0 & \text{if } x = 0, \end{cases}$$

for $x \in F$. Then the Boolean values **F**, **T** are simulated by the field elements 0, 1, sign(x) by $x * \omega^-(x)$, and sel(x_1, x_2, y) by $x_1 y + x_2(1-y)$; $\neg$, $\bigvee$, and $\bigwedge$ are clear. However, we prefer to keep the arithmetic and Boolean parts distinct. $\square$

Remark 2.4. We want to discuss the related random access model of algebraic decision trees (Strassen [1972, 1983]), where one has a binary tree with vertices having one child labelled by an operation from Ω, and vertices having two children labelled by a test from a set P of relations. Our notion of an arithmetic network over F with (n, m) inputs is equivalent to Strassen's notion of a computation tree of type $(\Omega_F^1(n, m), \emptyset)$. However, it is more in the spirit of the model to compare with computation trees of type (Ω, P), where $\Omega = \{+, -, *, /\} \cup F$, $P = \{=0\}$. These computation trees easily simulate the conditional inverse and thus any arithmetic network, as in Remark 2.3.

However, arithmetic networks cannot efficiently simulate these computation trees. Let $n \in \mathbf{N}$, $N = \{1, \ldots, n\}$, and $b_I \in N$ for $I \subseteq N$. Then, testing $a_1, \ldots, a_n$ successively, one obtains a computation tree of cost n which computes the function $f : F^n \longrightarrow F$ with

$$f(a_1, \ldots, a_n) = b_I, \text{ where } I = \{i : a_i = 0\}.$$

Suppose that the b_I are algebraically independent over the prime field K of F. and that the arithmetic network N over F computes f, and has size s. Then, on input $a \in K^n$, all values $\phi_v(a)$ are in the field generated by the constants in N over K. This field has transcendence degree 2^n, and thus $s \geq 2^n$. >PP For a similar argument over a finite field F (or $F = \mathbf{Q}$), let $g : \{0,1\}^n \longrightarrow \{0,1\}$ be a hard Boolean function", requring a Boolean circuit of size $2^n/n$ (Savage [197?]. Then one can use $f(a_1, \ldots, a_n) = g(\text{sign}(a_1), \ldots, \text{sign}(a_n)) \in \{0,1\}^n \subseteq F^n$ in the above example. Since arithmetic networks over such a field F can be efficiently simulated by Boolean circuits (von zur Gathen [1985]), any arithmetic circuit over F computing f has exponential size $\Omega(2^{\epsilon n})$, while there is a computation tree of cost n computing f.

The computation trees discussed here have an exponential number of leaves and an exponential number of constants.

3. Semantics

In this section, we define the semantics of arithmetic circuits and networks, and characterize the functions computed by them. An arithmetic circuit with one input and one output, say, computes a mapping $F \longrightarrow F$, and also a rational function in $F(x)$. This gives rise to two notions, the "value semantics" and the "function semantics". The value semantics are straightforward, both for circuits and for networks: one traces execution on the given input. The function semantics of an arithmetic circuit associate a rational function to each vertex of the computation graph. In the case of arithmetic networks, the presence of sign gates leads to partitions of the input space, say $F^n \times \{\mathrm{T},\mathrm{F}\}^m$, into zero- and nonzero-regions of sets of polynomials, together with rational functions on the pieces of the partition. I call these objects *piecewise rational functions.*

Let F be a field, $n, \nu \in \mathbf{N}$, $\alpha = (G, \lambda, \iota, \rho)$ an arithmetic circuit over F with n inputs and ν outputs, and $a \in F^n$. The *value semantics* of α on input a associates to each vertex of $G = (V, E)$ a value from $F \cup \{\infty\}$ by tracing the computation, using ι to order the in-edges to a vertex. ("∞" stands for "undefined".) Formally, we define ϕ: $V \times F^n \rightarrow F \cup \{\infty\}$ by induction along the depth of the vertex $v \in V$. (Input vertices have depth 0.) Let $a \in F^n$, $s = \sigma_1^0(\lambda(v)) \in \{0, 2\}$, and $e_1=(w_1,v), e_2=(w_2,v)$ be the in-edges to v if $s=2$ (in which case $\lambda(v) \in \{+,-,*,/\}$), and assume $\iota(e_1) < \iota(e_2)$. Then

$$\phi(v)(a) = \begin{cases} \lambda(v) & \text{if } \lambda(v) \in F, \\ a_k & \text{if } \lambda(v) = (in_F, k), \\ \phi(w_1)(a)\ \lambda(v)\ \phi(w_2)(a) & \text{if } s = 2, \end{cases}$$

where $b / 0 = \infty$ for any $b \in F$, and $b \,\omega c = \infty$ for any $\omega \in \{+,-,*,/\}$ and $\infty \in \{b, c\}$. Let $\rho = (v_1, \ldots, v_\nu)$ be the output vertices. The function ϕ_α: $F^n \rightarrow F^\nu \cup \{\infty\}$ computed by α is given by

$$\phi_\alpha(a) = \begin{cases} (\phi(v_1)(a),\ldots,\phi(v_\nu)(a)) & \text{if } \forall v \in V\ \phi(v)(a) \neq \infty, \\ \infty & \text{otherwise.} \end{cases}$$

Consider the two trivial circuits α_1 and α_2 of Figure 3.1, both with $\rho = (v_2)$. There

$$\phi_{\alpha_1}(a) = \begin{cases} 1 & \text{if } a \neq 0, \\ \infty & \text{if } a = 0, \end{cases}$$

and

$$\phi_{\alpha_2}(a) = 1$$

for all $a \in F$.

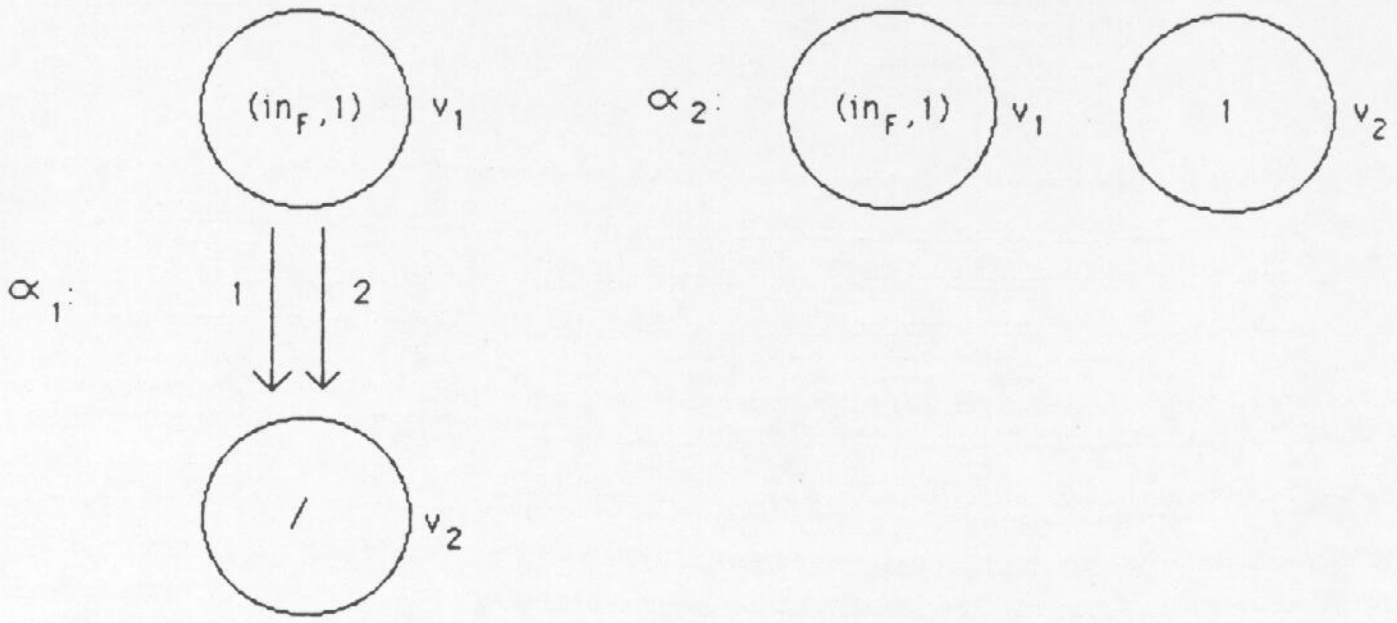

Figure 3.1

The *function semantics* associates to each vertex $v \in V$ a rational function $\psi(v) \in F(x_1,\ldots,x_n) \cup \{\infty\}$, again by induction on the depth of v. We use the above notation, and set

$$\psi(v) = \begin{cases} \lambda(v) & \text{if } \lambda(v) \in F, \\ x_k & \text{if } \lambda(v) = (in_F, k), \\ \psi(w_1)\, \lambda(v) \psi(w_2) & \text{otherwise,} \end{cases}$$

where $f / 0 = \infty$ for $f \in F(x_1,\ldots,x_n)$, and $f \omega g = \infty$ for $\omega \in \{+, -, *, /\}$ and $\infty \in \{f, g\}$. (For a ring F, one would use a slightly different notion, either disallowing divisions (e.g. when F is not an integral domain), or setting $f/g = \infty$ if $gh \neq f$ for all $h \in F[x_1, \ldots, x_n]$.) α computes the following sequence ψ_α of rational functions:

$$\psi_\alpha = \begin{cases} (\psi(v_1),\ldots,\psi(v_\nu)) & \text{if } \forall v \in V \quad \psi(v) \neq \infty, \\ \infty & \text{otherwise.} \end{cases}$$

Consider the two arithmetic circuits of Figure 3.2. We have $\psi_{\alpha_3} = x$ and $\psi_{\alpha_4} = x^2 \neq \psi_{\alpha_3}$. If $F = \mathbf{Z}_2$, then $\phi_{\alpha_3} = \phi_{\alpha_4}$. For any field F, $\psi_{\alpha_1} = \psi_{\alpha_2} = 1$, and $\phi_{\alpha_1} \neq \phi_{\alpha_2}$. Thus the two semantics just defined are mutually incomparable. However, if F is an infinite field, α, α' are arithmetic circuits with n inputs and $\phi_\alpha(a) = \phi_{\alpha'}(a)$ for all $a \in F^n$, then $\psi_\alpha = \psi_{\alpha'}$. Thus in this case, the value semantics is strictly finer than the functional semantics. For any $v \in V$ and $a \in F^n$, $\phi(v)(a)$ is either $\psi(v)(a)$ or ∞; $\phi(v)(a) = \infty$ if and only if $\psi(w) = f/g$ and $g(a) = 0$ for some vertex w on a path from an input to v.

It is clear that the rational functions computed by arithmetic circuits are precisely the elements of $F() = \bigcup_{n,\nu \in \mathbf{N}} F(x_1,\ldots,x_n)^\nu \cup \{\infty\}$. The remainder of this section is devoted to finding a similar description for arithmetic networks. It is clear how to extend the value semantics to an arithmetic network N over F with (n, m) inputs and (ν,μ) outputs. For each vertex $v \in V$ and input $(a, b) \in F^n \times \mathbf{B}^m$ we have $\phi(v)(a, b) \in F \cup \{\infty\}$ if $\tau(\lambda(v)) = F$, and $\phi(v)(a, b) \in \mathbf{B} \cup \{\infty\}$ if $\tau(\lambda(v)) = \mathbf{B}$. N *computes* the function

$$\phi_N : F^n \times \mathbf{B}^m \rightarrow (f^\nu \times \mathbf{B}^\mu) \cup \{\infty\},$$

given by the output vertices. E.g., in Figure 2.1 we have

$$\phi(v_4)(a_1,a_2) = \begin{cases} \mathbf{F} & \text{if } a_1 = 0, \\ \mathbf{T} & \text{otherwise,} \end{cases}$$

$$\phi(v_7)(a_1,a_2) = \begin{cases} -1 & \text{if } a_1 = 0, \\ a_1 & \text{otherwise.} \end{cases}$$

$$\phi_N(a_1,a_2) = \begin{cases} (-a_2/a_1, \mathbf{T}) & \text{if } a_1 \neq 0, \\ (0, \mathbf{T}) & \text{if} a_1 = a_2 = 0, \\ (a_2, \mathbf{F}) & \text{if} a_1 = 0 \neq a_2. \end{cases}$$

Recall from algebraic geometry that a set $U \subseteq F^n$ is called locally closed if and only if there exist $f_1,\ldots,f_r, g \in F[x_1,\ldots,x_n]$ such that

$$U = \{a \in F^n : f_1(a) = \cdots = f_r(a) = 0, \; g(a) \neq 0\}.$$

Definition 3.1.

(i) A set $U \subseteq F^n \times \mathbf{B}^m$ is *locally closed* if and only if there exist $V \subseteq F^n$ locally closed and $C \subseteq \mathbf{B}^m$ such that $U = V \times C$.

(ii) A *piecewise rational function* $f = (U_i, f_{i1}^F, \ldots, f_{i\nu}^F, f_{i1}^{\mathbf{B}}, \ldots, f_{i\mu}^{\mathbf{B}})_{i \in I}$ over F with (n, m) inputs and (ν, μ) outputs consists of a finite partition

$$F^n \times \mathbf{B}^m = \bigcup_{i \in I} U_i$$

into locally closed subsets U_i, rational functions

$$f_{ij}^F \in F(x_1, \ldots, x_n) \cup \{\infty\}$$

for $1 \leq j \leq \nu$, and Boolean functions

$$f_{ij}^{\mathbf{B}} \colon \mathbf{B}^m \to \mathbf{B}^\mu \cup \{\infty\}$$

for $1 \leq j \leq \mu$. It is required that for each $i \in I$, either $f_{ij}^* = \infty$ for all j, or $f_{ij}^* \neq \infty$ for all j and $* \in \{F, \mathbf{B}\}$, and

$$\text{projection}_{F^n}(U_i) \subseteq \text{domain}(f_{ij}^F)$$

for $1 \leq j \leq \nu$. □

(More properly, we should define each $f_{ij}^F \in F(U_i)$ to be a rational function on U_i; the above definition avoids this concept for simplicity.)

We denote by $\tau(f_{ij}^*) = *$ the type of f_{ij}^*, with $* \in \{F, \mathbf{B}\}$. We write f_{ij}^* when no confusion about the type is possible. Note as a peculiarity that our piecewise rational functions are "oblivious", i.e. the number $k = \nu + \mu$ of outputs is the same for each $i \in I$. This corresponds to the fact that an arithmetic network has a fixed sequence of output vertices.

We will define the function semantics along the computation graph. It is sufficient to say which piecewise rational function $\psi_\omega = (U_i, f_i^*)_{i \in I}$ is computed by an operation $\omega \in \Omega_F^1(n, m)$, and to define the composition of piecewise rational functions.

1. If $\omega = (in_F, k)$, then $I = \{1\}$, $U_1 = F^n \times \mathbf{B}^m$, and $f_1^F = x_k$; similarly for $\omega \in F$ or $\omega \in \mathbf{B}$.
2. If $\omega = (in_{\mathbf{B}}, k)$, then $I = \{1, 2\}$, $f_1^{\mathbf{B}} = \mathbf{T}$, $f_2^{\mathbf{B}} = \mathbf{F}$, $U_i = F^n \times \mathbf{B}^{k-1} \times \{f_i^{\mathbf{B}}\} \times \mathbf{B}^{m-k}$.
3. If $\omega \in \{+, -, *\}$, then $I = \{1\}$, $U_1 = F^2$, and $f_1^F = x_1 \omega x_2$.
4. If $\omega = /$, then $I = \{1, 2\}$, $U_1 = F \times (F \setminus \{0\})$, $U_2 = F \times \{0\}$, $f_1^F = x_1 / x_2$, and $f_2^F = \{\infty\}$.

5. If $\omega = \text{sign}$, then $I = \{1, 2\}$, $U_1 = F \setminus \{0\}$, $U_2 = \{0\}$, $f_1^B = \mathbf{T}$, $f_2^B = \mathbf{F}$.
6. If $\omega = \text{sel}$, then $I = \{1, 2\}$, $U_1 = F^2 \times \{\mathbf{T}\}$, $U_2 = F^2 \times \{\mathbf{F}\}$, $f_1^F = x_1$, $f_2^F = x_2$.
7. If $\omega = \neg$, then $I = \{1,2\}$, $U_1 = \{\mathbf{T}\}$; $f_1^{\mathrm{B}} = \mathbf{F}$, $U_2 = \{\mathbf{F}\}$, $f_2^{\mathrm{B}} = \mathbf{T}$; similarly for $\omega \in \{\wedge, \vee\}$.

In order to define the composition of piecewise rational functions, let $f_1, \ldots, f_s, g$ be piecewise rational functions, and $\sigma(g) = (s_1, s_2)$ the arity of g. In order to substitute the f 's into g, it is necessary that all the f 's have the same arity, say (n, m), and that exactly s_1 of the outputs of all the f 's have arithmetic type, and exactly s_2 have Boolean type. To simplify notation, we assume that $s = s_1 + s_2$ and each $f_l = (U_{li}, f_{li}^*)_{i \in I_l}$ has exactly one output, with $\tau(f_{li}^*) = F$ if $l \leq s_1$, and $\tau(f_{li}^*) = \mathbf{B}$ otherwise. We also assume that $g = (V_j, g_j^*)_{j \in J}$ has one output. This assumption is satisfied for any ψ_ω, and furthermore, the general case is easily reduced to this case.

Definition 3.2. The *composition*

$$h = g \circ (f_1, \ldots, f_s) = (W_k, h_k)_{k \in K}$$

is defined as follows. For

$$k = (i_1, \ldots, i_s, j) \in I_1 \times \cdots \times I_s \times J,$$

let

$$W_k = (f_{1i_1}^*, \ldots, f_{si_s}^*)^{-1}(V_j) \cap U_{1i_1} \cap \cdots \cap U_{si_s}.$$

Then

$$K = \{k \in I_1 \times \cdots \times I_s \times J : W_k \neq \emptyset\}.$$

For $k \in K$, let

$$\hat{h}_k = g_j(f_{1i_1}^*, \ldots, f_{si_s}^*)$$

and $h_k = \hat{h}_k$ if $W_k \subseteq \text{dom}(\hat{h}_k)$, and $h_k = \infty$ otherwise. □

One verifies that h is a piecewise rational function, with (n, m) inputs.

We can now define the *function semantics* of an arithmetic network $N = (G, \lambda, \iota, \rho)$ over F with (n, m) inputs and (ν, μ) outputs. To each vertex $v \in V$ of the computation graph $G = (V, E)$, we associate a piecewise rational function $\psi(v)$, by induction on the depth of v. First, if v has depth zero, then $\psi(v) = \psi_{\lambda(v)}$. If the depth of v is at least 1, then $\psi(v)$ is the composition of $\psi_{\lambda(v)}$ with the piecewise rational functions of the input vertices w to v, ordered by $\iota(w)$. Finally, the piecewise rational function ψ_N computed by N is the combination of $\psi(v_1), \ldots, \psi(v_{\nu+\mu})$, where $\rho = (v_1, \ldots, v_{\nu+\mu})$ are the output vertices of N. (Formally, it is the composition of $g = (F^\nu \times \mathbf{B}^\mu, f_1^*, \ldots, f_{\nu+\mu}^*)$, where each f_i^* is the identity, with $\psi(v_1), \ldots, \psi(v_{\nu+\mu})$.)

For example, in Figure 2.1 we have

$$\psi(v_7) = (U_i, f_i^F)_{i \in \{1,2\}} \text{ with (2, 0) inputs and (1, 0) outputs,}$$
$$U_1 = (F \setminus \{0\}) \times F, \; f_1^F = x_1,$$
$$U_2 = \{0\} \times F, \; f_2^F = -1;$$
$$\psi(v_9) = (U_i, f_i^F)_{i \in \{1,2\}} \text{ with (2, 0) inputs and (1,0) outputs,}$$
$$U_i \text{ as above, } f_1^F = -x_2 / x_1, \; f_2^F = -x_2.$$

The piecewise rational function ψ_N of the network N in Figure 2.1 is

$$f = (U_i, f_{i1}^F, f_{i1}^{\mathbf{B}})_{1 \le i \le 3},$$
$$U_1 = (F \setminus \{0\}) \times F, \; f_{11}^F = -x_2 / x_1, \; f_{11}^{\mathbf{B}} = \mathbf{T},$$
$$U_2 = \{0\} \times \{0\}, \; f_{21}^F = 0, \; f_{21}^{\mathbf{B}} = \mathbf{T},$$
$$U_3 = \{0\} \times (F \setminus \{0\}), \; f_{31}^F = 0, \; f_{31}^{\mathbf{B}} = \mathbf{F}.$$

Thus f describes the solvability (by $f_{i1}^{\mathbf{B}}$) and solution (by f_{i1}^F) of the linear equation $x_1 t + x_2 = 0$.

Remark 3.3. The functions ψ_N computed by arithmetic networks N over F are precisely the piecewise rational functions over F.

4. Degree of piecewise rational functions

If $f = \frac{g}{h} \in F(x_1,\ldots,x_n)$ and $g, h \in F[x_1,\ldots,x_n]$ with $\gcd(g, h) = 1$, then the degree of f is

$$\deg f = \max\{\deg g, 1 + \deg h\},$$

where $\deg g$ is the total degree of g.

In this section, we recall the notion of degree from algebraic geometry for a piecewise rational function, generalizing the degree of a rational function. The degree is a powerful tool for proving lower bounds, in some cases fulfilling the complexity theorists's dream: matching upper and lower bounds (see Strassen [1984] for sequential results, and Kung [1976], von zur Gathen [1984b] for a simple case in parallel computation). In Section 5, the degree will yield a lower bound on the depth of arithmetic networks. For a general background on algebraic geometry, see Shafarevich [1974], Chapter I, and for the degree, Griffiths and Harris [1979] and Heintz [1979]. Throughout the section, we assume an algebraically closed ground field F. $X \subseteq F^n$ is called a *closed subvariety* if and only if there exist $f_1,\ldots,f_r \in F[x_1,\ldots,x_n]$ such that

$$X = \{f_1 = \cdots = f_r = 0\} = \{a \in F^n : f_1(a) = \cdots = f_r(a) = 0\}$$

The degree of X then is

$$\deg X = \max\{\#L \cap X : L \subseteq F^n \text{ affine linear, } L \cap X \text{ finite}\}.$$

If $f \in F[x_1,\ldots,x_n] \setminus F$ is squarefree and $X = \{f = 0\}$, then $\deg X = \deg f$. For $X \subseteq F^n$ arbitrary, the Zariski closure of X is

$$\overline{X} = \bigcap\{Y : Y \subseteq F^n \text{ closed, } X \subseteq Y\}.$$

$\overline{X}$ is closed. If $X \subseteq F^n$ and $C \subseteq \mathbf{B}^m$, we define

$$\deg(X \times C) = \deg \overline{X}.$$

Now let $n, m, \nu, \mu \in \mathbf{N}$, and

$$f = (U_i, f_{i1}^F, \ldots, f_{i\nu}^F, f_{i1}^{\mathbf{B}}, \ldots, f_{i\mu}^{\mathbf{B}})_{i \in I}$$

be a piecewise rational function. In order to define $\deg f$, we consider for $i \in I$ the following permutation of the graph of $(f_{i1}^F, \cdots f_{i\nu}^F, f_{i1}^{\mathbf{B}}, \cdots f_{i\mu}^{\mathbf{B}}) \restriction U_i$:

$$X_i = \{(a, c, b, d) \in F^n \times F^\nu \times \mathbf{B}^m \times \mathbf{B}^\mu :$$

$$\forall j \quad (a, b) \in U_i \text{ and } c_j = f_{ij}^F(a, b) \text{ and } d_j = f_{ij}^{\mathbf{B}}(a, b)\}.$$

If some f_{ij}^* is ∞ (and therefore all f_{ij}^*'s), we use $X_i = U_i \subseteq F^n \times \mathbf{B}^m$. Since U_i is locally closed and each $f_{ij}^{\mathbf{B}}$ is constant on U_i, there exist $X_i' \subseteq F^{n+\nu}$, $X_i'' \subseteq \mathbf{B}^{m+\mu}$ such that $X_i = X_i' \times X_i''$, and thus $X_i \subseteq F^{n+\nu} \times \mathbf{B}^{m+\mu}$ is locally closed.

Definition 4.1. In the above notation, we define

$$\deg((U_i, f_i)) = \deg \overline{X_i}$$

$$\deg f = \max_{i \in I} \deg((U_i, f_i)). \quad \square$$

We recall the following facts.

Fact 4.2. (i) Let $\phi: F^n \to F^m$ be an affine linear mapping, and $X \subseteq F^n$ closed. Then

$$\deg \overline{\phi(X)} \leq \deg X.$$

(ii) Let proj: $F^n \to F^m$ with $m \leq n$ be a projection on some coordinates, and $X \subseteq F^m$ closed. Then

$$\deg \text{proj}^{-1}(X) = \deg X.$$

(iii) (Bezout's inequality) Let $X, Y \subseteq F^n$ be closed. Then $X \cap Y$ is closed, and

$$\deg(X \cap Y) \leq \deg X \cdot \deg Y. \quad \square$$

For proofs of (iii), see Schnorr [1981] and Heintz [1979]. Recall that in this section, we assume that F is algebraically closed.

Theorem 4.3. Let F be algebraically closed, $g, f_1, \ldots, f_s$ be piecewise rational functions, $\deg f_i \leq d$ for $1 \leq i \leq s$, and $h = g(f_1, \ldots, f_s)$ their composition. Then

$$\deg h \leq \deg g \cdot \prod_{1 \leq i \leq s} \deg f_i \leq \deg g \cdot d^s.$$

Proof. In the notation used for defining the composition h (and assuming $\tau(h) = F$), we fix some $k = (j, i_1, \ldots, i_s) \in K$, and consider

$$\begin{aligned} Z_k &= \text{graph}(h_k \restriction \text{proj}_1(W_k)) \\ &= \{(a, c) \in F^n \times F : \exists! \; b \in \mathbf{B}^m \; (a, b) \in W_k \text{ and } c = h_k(a, b)\} \\ &= \{(a, c) \in F^n \times F : \exists! \; b \in \mathbf{B}^m \; (a, b) \in U_{1i_1} \cap \cdots \cap U_{si_s}, \\ &\qquad (f_{1i_1}(a, b), \ldots, f_{si_s}(a, b)) \in V_j, \text{and } c = g_k(f_{1i_1}(a, b), \ldots, f_{si_s}(a, b))\}. \end{aligned}$$

Now consider

$$\begin{aligned} A_i &= \{(a, b, d, c) \in F^n \times \mathbf{B}^m \times (F^{s_1} \times \mathbf{B}^{s_2}) \times F: \\ &\quad (a, b) \in U_{1i_1} \cap \cdots \cap U_{si_s}, (d_1, \ldots, d_s) \in V_j, \\ &\quad c = g_j(d_1, \ldots, d_s), d_l = f_{l\, i_l}(a, b) \ \forall 1 \le l \le s\} \\ &= \mathrm{proj}_{34}^{-1}(\mathrm{graph}(g_j \restriction V_j)) \cap \\ &\quad \bigcap_{1 \le l \le s} \mathrm{proj}_{12l}^{-1}(\mathrm{graph}(f_{l\, i_l} \restriction U_{l\, i_l})), \end{aligned}$$

where "proj" is the appropriate projection. By Fact 4.2 (ii) and (iii), $\deg A_k \le \deg g \cdot (\deg f)^s$. Also, Z_k is a projection of A_k, and by Fact 4.2 (i),

$$\deg h = \max_{k \in K} \deg Z_k \le \deg g \prod_{1 \le i \le s} \deg f_i \le \deg g (\max_{1 \le i \le s} \deg f_i)^s. \qquad \square$$

Corollary 4.4. Let $\omega \in \Omega_F^1$ be an operation with $s = \tau(\omega)$ inputs, and $f_1, \ldots, f_s$ piecewise rational functions with $\deg f_i \le d$ for $1 \le i \le s$. Then

$$\deg(\omega \circ (f_1, \ldots, f_s)) \le 2d^2.$$

Proof. For all $\omega \in \Omega_F^1$ except $\omega = \mathrm{sel}$ we have $s \le 2$, and the claim follows from Theorem 4.3. Now let $\omega = \mathrm{sel}$, so that $s = 3$, and, changing notation slightly, let

$$e = \mathrm{sel} \circ (f, g, h),$$

$$f = (U_i, f_i)_{i \in I}, \quad g = (V_j, g_j)_{j \in J}, \quad h = (W_k, h_k)_{k \in K}.$$

Then

$$e = (T_l, e_l)_{l \in L}, \quad L = I \times J \times K,$$

and for $l = (i, j, k) \in L$ we have

$$T_l = U_i \cap W_k \text{ and } e_l = f_i \text{ if } h_k = T,$$

$$T_l = V_j \cap W_k \text{ and } e_l = g_j \text{ if } h_k = F.$$

In the first case,

$$\deg(T_l, e_l) \le \deg(U_i, f_i) \cdot \deg(W_k, h_k),$$

and similarly in the second case. $\square$

5. Size and Depth

In this section, we define the two cost measures of interest to us, namely size and depth, and establish a lower bound on the depth in terms of the degree. It turns out that the degree may be exponentially larger for arithmetic networks than for arithmetic circuits of the same depth.

We first return to the general Ω-programs of Section 2, and now assume that we have a (depth) cost function for the operations

$$\delta\colon \Omega \rightarrow \mathbf{R}_{\geq 0}.$$

Definition 5.1. If δ is a cost function on Ω, and $\alpha = (G, \lambda, \iota, \rho)$ with $G = (V, E)$ an Ω-program, then

$$S(\alpha) = \sum_{v \in V} \delta(\lambda(v))$$

is the size of α, and

$$D(\alpha) = \max_{\substack{P \text{ path} \\ \text{in } G}} \sum_{v \text{ on } P} \delta(\lambda(v))$$

is the depth of α. □

As usual, a path P in G is a sequence $(v_0, v_1, \ldots, v_t)$ with $(v_i, v_{i+1}) \in E$ for all i, and the above sum is over $v_0, \ldots, v_t$. Now let F be a field, $n, m, \nu, \mu \in \mathbf{N}$; then our standard cost function is

$$\delta\colon \Omega_F^1(n, m) \longrightarrow \mathbf{N},$$

$$\delta(\omega) = \begin{cases} 1 & \text{if } \omega \in \{+, -, *, /, \dashv, \bigvee, \bigwedge, \text{sel}, \text{sign}\}, \\ 0 & \text{otherwise.} \end{cases}$$

The size and depth of arithmetic circuits and networks over F, as now defined, are well-studied in several contexts. Strassen [1985] gives an overview concerning the (non-scalar) size for Ω^0 ("length of computations"), where his degree bound (Strassen [1972]) yields matching asymptotic upper and lower bound for interesting problems. Strassen [1984] and Ben-Or [1983] give very strong results about the sequential complexity of computation trees for non-scalar cost; this model is discussed in Remark 2.4. Cook [1985] provides an overview of depth results for Boolean circuits.

The depth of arithmetic circuits equals, within a constant multiple, the logarithm of the formula size for rational functions (Brent [1974], Muller & Preparata [1976]).

Fact 5.2. Suppose that F is infinite, and the arithmetic circuit α over F computes the rational function f. Then

$$D(\alpha) \geq \log(\deg f). \quad \square$$

This was shown by Kung [1976]; von zur Gathen [1984b] has a version over a finite field F. (All logarithms in this paper are in base 2.) Recall that for

$$f = \frac{g}{h} \in F(x_1,\ldots,x_n)$$

with $g, h \in F[x_1,\ldots,x_n]$ and $\gcd(g, h) = 1$ we have

$$\deg f = \max\{\deg g, 1 + \deg h\}.$$

(This agrees with the usual degree of a polynomial f, unless $f \in F$.) Consider the corresponding piecewise rational function $\tilde{f} = (U_i, f_i^F)_{i \in I}$ with $I = \{1, 2\}$, $(U_1, f_1^F) = (\{h \neq 0\}, f)$ and $(U_2, f_2^F) = (\{h = 0\}, \infty)$. Then

$$\deg f = \deg \tilde{f}.$$

We first show that for any arithmetic network of depth d computing this $\tilde{f}$ we have $d \geq \log(\deg \tilde{f})$ (Proposition 5.5). Then we see that for more general piecewise rational functions only a much weaker lower bound on depth holds (Theorem 5.7).

Let $N = (G, \lambda, \iota, \rho)$ be an arithmetic network over F with (n, m) inputs and (ν, μ) outputs, $G = (V, E)$, and

$$V_{\mathbf{B}} = \tau^{-1}(\mathbf{B}) = \{v \in V : \tau(v) = \mathbf{B}\}$$

the set of Boolean vertices. A function

$$\pi : V_{\mathbf{B}} \to \{\mathbf{T}, \mathbf{F}, \infty\}$$

is called a Boolean assignment for N, and

$$A_\pi = \{(a, b) \in F^n \times \mathbf{B}^m : \forall v \in V_{\mathbf{B}} \ \phi_v(a, b) = \pi(v)\}$$

is the set of inputs that yields π. Note that b is determined by π, and that A_π may be empty.

Let N_π be the arithmetic circuit over F with n inputs obtained from the network N by leaving away the Boolean part of N, and replacing the selection gates by a fixed connection according to π. Then for $(a, b) \in A_\pi$, the value semantics of N_π on input a is equal to the value semantics (at the arithmetic vertices) of N on input (a, b).

Definition 5.3. Let N be an arithmetic network over F, and $f = (U_i, f_i)_{i \in I}$ a piecewise rational function. Then N ***computes*** f ***strictly*** if and only if N computes f and for all $i \in I$ there exists a Boolean assignment π_i for N such that $U_i = A_{\pi_i}$. □

Note that in particular each $\mathrm{proj}_2(U_i) \subseteq \mathbf{B}^m$ consists of at most one element. The piecewise rational function computed strictly by N is essentially unique, except that in our definition (say, with $m = \mu = 0$) $f_{ij} \in F(x_1, \ldots, x_n)$, and different such f_{ij} may restrict to the same rational function on U_i. Without some condition such as "strictly" we cannot expect a relation between the depth $D(N)$ and the degree $\deg f$, since the trivial arithmetic network N computing 0 also computes $f = ((U_1, 0), (U_2, 0))$ for a partition $F^n = U_1 \cup U_2$ where U_2, say, might have arbitrarily large degree.

Definition 5.4. If $f = (U_i, f_i)_{i \in I}$ is a piecewise rational function, then $F^n \times \mathbf{B}^m = \bigcup_{i \in I} U_i$ is a partition, and $F^n = \bigcup_{i \in I} \mathrm{proj}_1(U_i)$. Any U_i such that the Zariski-closure of $\mathrm{proj}_1(U_i)$ equals F^n is called a thick subset. The thick degree of f is

$$\text{thick-deg}(f) = \max_{U_i \text{ thick}} \deg(U_i, f_i). \quad \square$$

Trivially, $\deg f \geq \text{thick-deg}(f)$.

Proposition 5.5. Suppose that F is algebraically closed, and the arithmetic network N computes f strictly. Then

$$\mathrm{D}(N) \geq \log(\text{thick-deg}(f)).$$

Proof. Let $f = (U_i, f_i)_{i \in I}$, and $i \in I$ with U_i thick. By definition, the Boolean components of U_i and f_i are constant, and there exists a Boolean assignment π for N such that $U' = U_i \cap A_\pi$ is such that $\mathrm{proj}_1(U')$ is dense in F^n. Let $f' = (U', f_i \restriction U')$. Then $\deg f' = \deg(U_i, f_i)$, and N_π is an arithmetic circuit computing the rational function f'' given by f'. By Fact 5.2,

$$D(N) \geq D(N_\pi) \geq \log(\deg f'') \geq \log(\deg f') = \deg(U_i, f_i). \quad \square$$

Example 5.6. It comes as a surprise that Proposition 5.5 does not hold with thick-deg(f) replaced by $\deg(f)$. Consider the arithmetic network N of Figure 5.1, with $(n, 0)$ inputs over F, where n is a power of two. (For simplicity, we assume $\mathrm{char} F = 0$.) The bold + stands for a binary tree of +-gates. Let v_{8j} be the j-th sel-gate, $1 \leq j \leq n/2$. The piecewise rational function computed at v_{8j} is

$$\psi_{v_{8j}} = ((U_{j1}, f_{j1}), (U_{j2}, f_{j2})),$$

where $U_{j1} = \{x_{2j-1}x_{2j} - 1 \neq 0\}$, $f_{j1} = 0$, $U_{j2} = F^n \setminus U_{j1}$, $f_{j2} = 1$. ψ_N consists of $2^{n/2}$ pieces, each a product of hyperbolas or their complements in F^2. The "smallest" piece is

$$V = \bigcap_{1 \leq j \leq n/2} U_{j2} = \{(a_1,\ldots,a_n) \in F^n : \forall j \leq n/2 \;\; a_{2j-1}a_{2j} - 1 = 0\}$$

with the (constant) rational function $g = n/2$. V is a product of $n/2$ hyperbolas, and has degree $\deg V = 2^{n/2}$. Thus $\deg(\psi_N) \geq 2^{n/2}$ (in fact, equality holds), and $D(N) = 4 + \log_2(n/2) = 4 + \operatorname{loglogdeg}\psi_N$. □

Given our general need of methods for proving lower bounds, this example is rather disappointing. Next, we show that things at least cannot get much worse.

Theorem 5.7. Let F be an algebraically closed field, N an arithmetic network over F, and f a piecewise rational function that N strictly computes. Then

$$D(N) \geq \log(1 + \log \deg(f)).$$

Proof. As usual, let $N = (G, \lambda, \iota, \rho)$ and $G = (V, E)$. The depth $D : V \to \mathbf{N}$ of vertices is defined in a natural way by

$$D(v) = \max_P \sum_{w \text{ on } P} \delta(w),$$

where the maximum is over all paths P in G ending in v. ψ_v denotes the function semantics of N. We show by induction on $D(v)$ that

$$\deg \psi_v \leq 2^{2^{D(v)}-1}$$

for any $v \in V$. Let $\omega = \lambda(v)$. If $D(v) = 0$, then

$$\omega \in F \cup (\text{in}_F \times N) \cup (\text{in}_{\mathbf{B}} \times M),$$

and $\psi_v = (U, f)$ with $U = F^n \times \mathbf{B}^m$ and

$$f \in F \cup \{x_1,\ldots,x_n\} \cup \{y_1,\ldots,y_m\}.$$

Thus $\deg \psi_v = 1 = 2^{2^0-1}$.

Now let $D(v) > 0$, $\omega = \lambda(v) \in \Omega^1$, and $e_1,\ldots,e_s$ the in-edges to v, with $e_i = (w_i, v)$ and $w_i \in V$. and let $d = \max_{1 \leq i \leq s} (\deg \psi_{w_i})$. Then $1 \leq s \leq 3$, $\delta(\omega) = 1$, $\deg(\omega) \leq 2$, and $D(v) = \max_{1 \leq i \leq s} 1 + D(w_i)$. By the induction assumption,

$$d \leq 2^{2^{D(v)-1} - 1}.$$

By Corollary 4.4,

$$\deg \psi_v = \deg(\omega(\psi_{w_1},\ldots,\psi_{w_s})) \leq 2d^2 \leq 2(2^{2^{D(v)-1}-1})^2 = 2^{2^{D(v)}-1}. \qquad \square$$

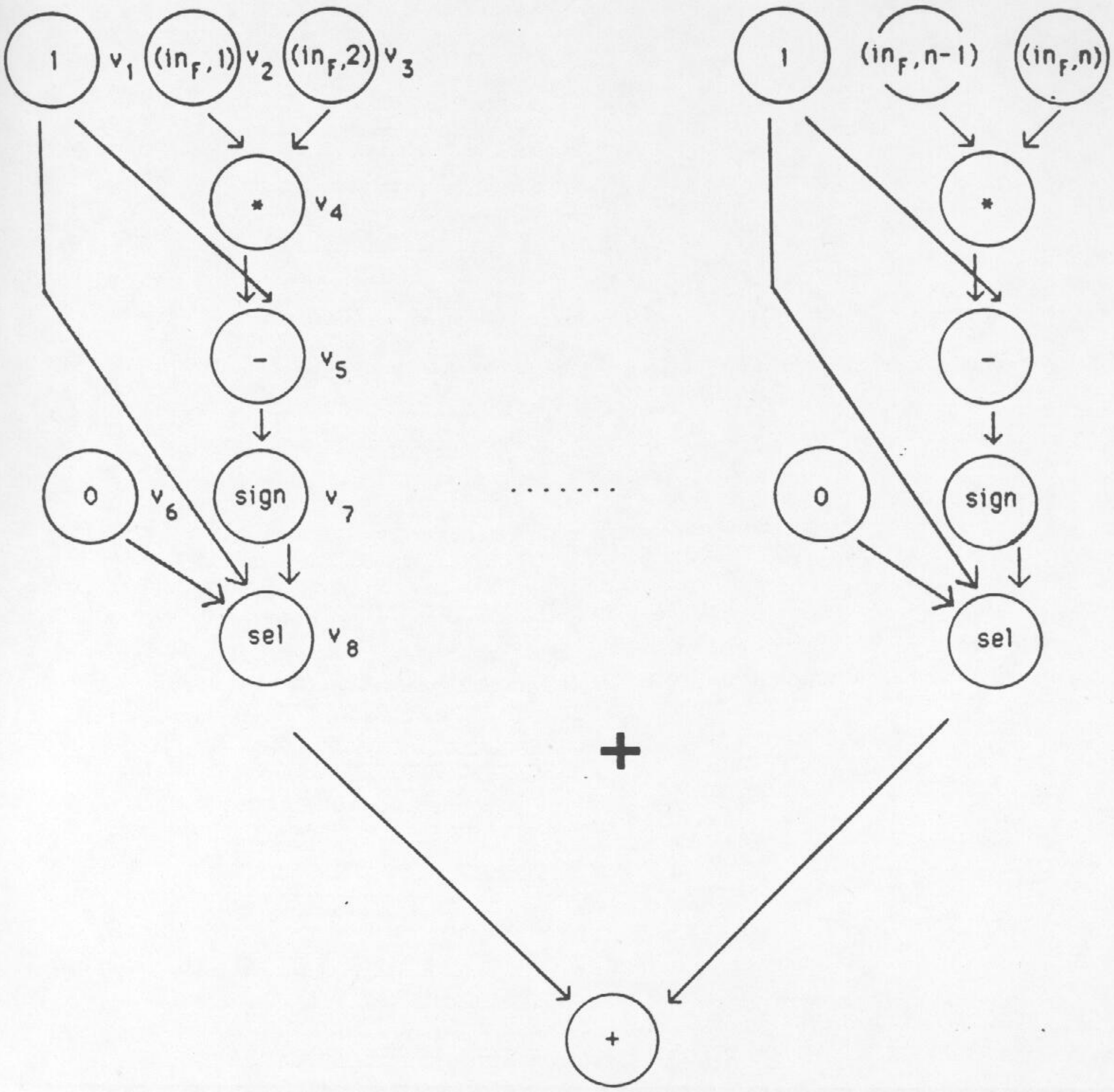
1
v_1
$(in_F, 1)$
v_2
$(in_F, 2)$
v_3
*
v_4
-
v_5
0
v_6
sign
v_7
sel
v_8
....
1
$(in_F, n-1)$
(in_F, n)
*
-
0
sign
sel
+
+

Figure 2.2

6. Network families and uniformity

In this Section, we give a framework for discussing families of arithmetic networks (which can deal with inputs of varying size) and *uniformity*. Since any given network has a fixed number of inputs and outputs, we only consider the computation of oblivious functions $f : F^* \times \mathbf{B}^* \rightarrow F^* \times \mathbf{B}^*$ where the length ν, μ of the outputs are functions only of the input lengths n, m. Let

$$f^{n,m} : F^n \times \mathbf{B}^m \rightarrow F^{\nu(n,m)} \times \mathbf{B}^{\mu(n,m)}$$

be f restricted to inputs of length n, m.

Definition 6.1. A *network family over* F is a sequence $N = (N_i)_{i \in \mathrm{N}}$ of networks over F, where N_i has $(n(i), m(i))$ inputs. It is assumed that $n(i) + m(i) \leq n(j) + m(j)$ for $i < j$, and, if $n(i)+m(i) = n(j)+m(j)$, then $n(i) < n(j)$. Then N *computes* f if and only if for all i, N_i computes $f^{n(i),m(i)}$, i.e. $\phi_{N_i} = f^{n(i),m(i)}$. □

Remark 6.2. The order condition on $n(i)$, $m(i)$ ensures that $n(i)+m(i) \geq \frac{1}{2}\sqrt{i}$ and $n(i)+m(i) \geq \sqrt{i}$ for $i \geq 11$; this will be used in Theorem 8.3. (The smallest possible values for $n(i)+m(i)$ are 0, 1, 1, 2, 2, 2, 3, ... for $i = 0, 1, 2, 3, 4, 5, 6, \cdots$.) □

The networks that we consider have varying degrees of "uniformity" (when we vary the input size) and "universality" (when we consider varying ground fields). For example:

1. The networks of Borodin, von zur Gathen & Hopcroft [1982], Berkowitz [1984] and Chistov [1985] for the characteristic polynomial of a $n \times n$ matrix look "the same" for all fields F and all natural n. Csanky's [1976] network for the same problem is only defined for fields of characteristic zero or $p > n$, since it involves a division by $n!$. However, where it is defined, it looks "the same" for any field and any n.
2. The deterministic network of Ibarra-Moran-Rosier [1980] for computing the rank of $n \times n$ matrices is defined "uniformly in n" and over any field, but for example works correctly only for real fields.
3. A network for the Discrete Fourier Transform would only be defined if the field has the required root of unity; furthermore this root of unity would have to be one of

the constants of the network.

4. Some of the algorithms mentioned in Section 11 use interpolation at n points. This is only possible if the field has at least n elements. For a finite field F this requires essentially different algorithms for the case when $n \leq \# F$ and when $n > \# F$.

5. Some of the algorithms in Section 11 use a Vandermonde matrix of n field elements, and also its inverse.

There are two problems in describing an arithmetic network N over F. One is how to specify the structure of the network, the other is how to specify the constants from F. To emphasize this, we will represent N by a "circuit" part $\alpha = \alpha_N$, where the constant gates are described by symbols, and the "constant" part $\beta = \beta_N$ consisting of the actual constants.

Definition 6.3. Let F be a field, $(G, \lambda, \iota, \rho)$ an arithmetic network over F, $k = \# \lambda^{-1}(F)$ the number of arithmetic constant gates in α, say $v_1, \ldots, v_k$, and $\lambda_1, \ldots, \lambda_k$ new symbols. Then the *circuit part* α_N of N is $\alpha_N = (G, \tilde{\lambda}, \iota, \rho)$, where $\tilde{\lambda}(v) = \lambda(v)$ if $\lambda(v) \notin F$, and $\tilde{\lambda}(v_i) = \lambda_i$. The *constant part* β_N of N is the vector $\beta_N = (\lambda(v_1), \ldots, \lambda(v_k)) \in F^k$. $\Box$

From α_N and β_N, N can be recovered, and we write $N = (\alpha_N, \beta_N)$, often dropping the subscript N. The circuit part of N can be described using the same kind of standard encoding as for Boolean circuits, since the $\lambda(v) \notin F$ can be encoded over **B**.

It is well-known that families of Boolean circuits can compute functions which are not Turing-computable. For lower bounds, this may not be relevant, but for upper bounds it is desirable to be able to compare them with models that take inputs of arbitrary length, such as Turing machines. Borodin [1977] proposed a notion of "uniformity", where a Turing machine, on input i, constructs the i-th circuit of the family. This circuit constructor is like a factory program that produces chips for inputs of arbitrary size for the given problem. We will be rather generous in allowing the circuit constructor to run in polynomial time, e.g., but want circuits that run very fast in parallel, e.g. in poly-logarithmic time. For certain applications, one may even allow random polynomial-time circuit constructors (see Fact 8.1).

A description of β_N similar to that of α_N - of finite length over a finite alphabet - is of course not possible for arbitrary constants from an uncountable field, e.g.. In our standard notion of uniformity, we only allow the constant 1, and for the circuit part follow Borodin [1977] (Definition 6.5). We then extend this to allow lists of pairwise different elements, and also more general constants (Definition 6.7). Eberly's [1986] results about polynomial arithmetic (Section 11) illustrate the subtle interplay between these notions of uniformity for constants, of L- and P-uniformity for the circuit part, and of the type of field (characteristic zero vs. infinite vs. finite). Abusing the usual notation, we denote by L (resp. P) the class of Boolean functions (rather than languages) computed by log-space (resp. polynomial-time) bounded deterministic Turing machines.

Definition 6.4.

(i) The *standard encoding* $\overline{\alpha}$ of the circuit part $\alpha = \alpha_N$ of a network $N = (\alpha, \beta)$ over F is a sequence of tuples $(\overline{v}, \overline{\lambda}, u_1, \ldots, u_s) \in \mathbf{B}^*$, one for each vertex v of N, where:

(a) $\overline{v}$ is the vertex number of v, encoded in unary,

(b) $\overline{\lambda} = \tilde{\lambda}(v)$ is the type of vertex v, encoded in some reasonable fashion,

(c) $s = \sigma(\lambda(v))$ is the arity of v, and the j-th input to v comes from the vertex with number u_j, for $1 \leq j \leq s$.

(ii) The *standard encoding* of a constant vector $\beta = (\beta_1, \ldots, \beta_k) = (1, \ldots, 1)$ is $\mathbf{T}^k$. □

Definition 6.5. A family $N = (N_i)_{i \in \mathbf{N}} = (\alpha_i, \beta_i)_{i \in \mathbf{N}}$ of arithmetic networks is *P-uniform* if $\beta_i \in \{1\}^*$ for all i, and the encodings $\overline{\alpha}_i, \overline{\beta}_i$ of α_i and β_i can be computed by a deterministic Turing machine using time $S(N_i)^{O(1)}$ on input i (in unary). If the Turing machine uses space $O(\log S(N_i))$, then N is *L-uniform.* □

L-uniformity will be our standard notion. The remainder of this section is only required for Section 11. Other notions of uniformity for Boolean circuits, as discussed e.g. in Ruzzo [1981] and Cook [1985], would also carry over to our arithmetic setting. In order to illustrate the power of more general constants, we introduce the following notions, which will only be used in Section 11.

Definition 6.6.

(i) A *description of constants* is a sequence $\gamma = (b_1, \ldots, b_k, d_1, \ldots, d_k)$, where each $b_j \in \{\infty, =, \neq\}$ and each d_j is a polynomial in $\mathbf{Z}[y_1, \ldots, y_{j-1}, y]$.

(ii) Let F be a field, and $F_0 \subseteq F$ its prime field. A *description of constants over F* is a description of constants $\gamma = (b_1, \ldots, b_k, d_1, \ldots, d_k)$ with the following property. There exist $\beta_1, \ldots, \beta_k \in F$ (the constants being described) such that for all j, $1 \leq j \leq k$, the following holds.

(a) If b_j is ∞, then β_j is transcendental over $F_0(\beta_1, \ldots, \beta_{j-1})$.

Otherwise, denote by $\overline{d}_j \in F_0(\beta_1, \ldots, \beta_{j-1})[y]$ the image of d_j under the mapping $y_l \longmapsto \beta_l$, for $1 \leq l < j$. Then

(b) If b_j is $=$, then $\overline{d}_j(\beta_j) = 0$ and $\overline{d}_j$ is an irreducible polynomial over $F_0(\beta_1, \ldots, \beta_{j-1})$.

(c) If b_j is $\neq$, then $\overline{d}_j(\beta_j) \neq 0$.

For any sequence $\beta = (\beta_1, \ldots, \beta_k)$ satisfying these conditions we say that γ *describes* β. If $\gamma = (\gamma_i)_{i \in \mathbf{N}}$ is a family such that γ_i describes $\beta_i = (\beta_{i1}, \ldots, \beta_{ik_i})$, then we say that γ *describes* $\beta = (\beta_i)_{i \in \mathbf{N}}$.

(iii) If α is a circuit part with constant vertices $v_1, \ldots, v_k$, and $\gamma = (b_1, \ldots, b_k, d_1, \ldots, d_k)$ a description of constants over F, then for any $\beta \in F^k$ described by γ we have an arithmetic network $N = (\alpha, \beta)$ over F. Now $\overline{N} = (\overline{\alpha}, \overline{\gamma})$ is called a *network description over F* if the value semantics of the outputs of (α, β) are independent of the particular choice of β described by γ. (The rational functions associated with the arithmetic vertices of (α, β) may depend on β, however.) We say that $(\overline{\alpha}, \overline{\gamma})$ *describes* any such $N = (\alpha, \beta)$. □

We now have a way of encoding circuit and constant parts of arithmetic networks. The one notion still lacking is the "size" of such description.

Clearly all such descriptions of constants can be encoded over some finite alphabet. We now assume a fixed such encoding, with the polynomial d_j given by a arithmetic circuit over $\mathbf{Q}$ (in effect a straight-line program over $\mathbf{Z}$) that uses inputs $y_1, \ldots, y_{j-1}, y$, constants 0,1, and only the operations $+, -, *$; and with all indices written in binary. We denote this encoding by $\overline{\gamma}$, and call it the *standard encoding of a description of constants.* Let s_j be the size of the arithmetic circuit computing d_j. The *size* of γ is $S(\gamma) = \sum_{1 \leq j \leq k} s_j$. Let $\gamma = (b_1, \ldots, b_k, d_1, \ldots, d_k)$ be a description of constants. Then the *degree* of γ_i is

$$\deg(\gamma) = \prod_{\substack{1 \leq j \leq k \\ b_j \text{ is } =}} \deg d_j .$$

Obviously, one can describe elements of arbitrary fields, finitely generated over the prime field, with $b_1 = \cdots = b_u$ being ∞, $b_{u+1} = \cdots = b_v$ being $=$, and $b_{v+1} = \cdots = b_k$ being $\neq$ for some u, v. Then $\deg(\gamma)$ is the degree of the finite algebraic field extension $[F_0(\beta_1, \ldots, \beta_v) : F_0(\beta_1, \ldots, \beta_u)]$, if γ describes $(\beta_1, \ldots, \beta_k)$.

Definition 6.7.

(i) A family $N = (N_i)_{i \in \mathbf{N}} = (\alpha_i, \beta_i)_{i \in \mathbf{N}}$ of arithmetic networks over F is *P-P-uniform* if there exists a family $(\gamma_i)_{i \in \mathbf{N}}$ of description of constants, with $(\overline{\alpha}_i, \overline{\gamma}_i)$ describing N_i, such that $\deg\gamma_i = i^{O(1)}$, and $\overline{\alpha}_i, \overline{\gamma}_i$ can be computed by a deterministic Turing machine using time $S(N_i)^{O(1)}$ on input i (in unary).

(ii) If the Turing machine as above works in space $O(\log S(N_i))$, then N is L*-*L*-uniform*.

(iii) If N is L*-*L*-uniform* and each b_j in each γ_i is either $\neq$ or else $d_j = 1$ and b_j is $=$, then N is called $\neq$*-uniform*. (I.e., only 1 and pairwise different constants are allowed.) □

Example 6.8. We want a description of constants γ_i describing i arbitrary pairwise distinct elements of F. I.e., γ_i describes any $\beta_i = (\beta_{i1}, \ldots, \beta_{ii})$, where for all $j \neq k$, $t_{ij} \neq \beta_{ik}$. For $i \in \mathbf{N}$, let

$$\gamma_i = (\neq, \ldots, \neq, d_1, \ldots, d_i),$$
$$d_1 = 1, \text{ and } d_j = (y - y_1) \cdots (y - y_{j-1}) \text{ for } j > 1.$$

Thus, γ_i describes β_i if and only if $\overline{d}_{i1}(\beta_{i1}) \neq 0$, where $\overline{d}_{i1} = 1 \in F[y]$, so that t_{i1} is an arbitrary element of F, and similarly $\overline{d}_{ij}(\beta_{ij}) \neq 0$ for $2 \leq j \leq i$, which is satisfied if and only if the entries of β_i are pairwise distinct. $\gamma = (\gamma_i)_{i \in \mathbf{N}}$ is a family of descriptions of constants over F if and only if F is infinite.

Example 6.9. Given i distinct elements $\beta_{i1}, \ldots, \beta_{ii}$ as in Example 6.8, we can also describe the entries of the corresponding Vandermonde matrix and its inverse. (Such constants are used in Eberly's [1984] algorithms; see Section 11.) To have a constant equal to $\beta_1{}^j$, e.g., we encode an arithmetic circuit that computes the j-th power; this can be done by a Turing machine with work space $O(\log j)$. The determinant then is a product of differences of entries. For the adjoint, we encode Berkowitz' [1984] determinant algorithm; this can be done L-uniformly. The running time of the determinant calculation - which does not enter our notion of uniformity - would be $i^{O(1)}$, with parallel time $O(\log^2 i)$.

Example 6.10. We now give a description of constants γ_i describing any $\beta_i = (t_1, \ldots, t_i) = (1, -1, \ldots, t_i^2, t_i)$, where t_i is a primitive 2^i-th root of unity. (These roots of unity appear in the usual treatment of the Fast Fourier Transform. We assume $\mathrm{char}F = 0$.) Let $\gamma_i = (=, \ldots, =, d_0, \ldots, d_i)$ where $d_0 = y - 1$, $d_1 = y + 1$, and for $j > 1$, $d_j = y^2 - y_{j-1}$. Then $\gamma = (\gamma_i)_{i \in \mathbf{N}}$ is a uniform family of descriptions of constants. It is "over F" if F has a primitive 2^j-th root of unity for all $j \geq 0$. Note that $\deg\gamma_i = 2^i$.

Example 6.11. We extend the previous example to obtain descriptions for primitive k-th roots of unity, where k is not necessarily a power of 2. For every composite number j, choose a prime divisor $p(j)$ of j, e.g. the smallest. Let $\gamma_i = (=, \ldots, =, d_1, \ldots, d_i)$ where $d_1 = y-1$, and for $j \geq 2$,

$$d_j = \begin{cases} y^{j-1} + y^{j-2} + \cdots + 1 & \text{if } j \text{ is prime,} \\ y^{p(j)} - y_{j/p(j)} & \text{otherwise.} \end{cases}$$

Then $\gamma = (\gamma_i)_{i \in \mathbf{N}}$ is a uniform description of constants over $\mathbf{C}$ for $(\beta_i)_{i \in \mathbf{N}}$, where $\beta_i = (t_1, \ldots, t_i)$ and t_j is a primitive j-th root of unity. One only has to check that any $p(j)$ can be computed in space $O(\log j)$, and that the sequence is "coherent with respect to p", i.e. $t_j^{p(i)} = t_{j/p(j)}$ for every composite number j.

Example 6.12. Let F be a field of characteristic zero. The description of constants $\gamma = (\gamma_i)_{i \in \mathbf{N}}$ with $\gamma_i = (=, \ldots, =, 1, y - y_0 + y_0, y - y_1^2, \ldots, y - y_{i-1}^2$ is a uniform description of constants over F, describing $\beta_i = (1, 2, 4, \ldots, 2^{2^i})$.

Remark 6.13. We have the following relations between our notions of uniformity. Let F be a field, and N a network family over F.

(i)

$$\begin{tikzcd}
 & N \text{ is } \neq\text{-uniform} \arrow[d, equal] \\
N \text{ is } L\text{-uniform} \arrow[r, Rightarrow] \arrow[d, equal] & N \text{ is } P\text{-uniform} \arrow[d, equal] \\
N \text{ is } L\text{-}L\text{-uniform} \arrow[r, Rightarrow] & N \text{ is } P\text{-}P\text{-uniform}
\end{tikzcd}$$

(ii) N is $\neq$-uniform, $\mathrm{char}(F) = 0 \Rightarrow N$ is L-uniform.

(iii) Let F be a field of characteristic zero, and $f = (f_i)_{i \in \mathbf{N}}$ with $f_i(x_1, \ldots, x_i) = 2^{2^i} \in F$. Then f can be computed by an L-L-uniform family of networks over F (Example 6.12), but not by a P-uniform family.

(iv) Let p be a prime number. Then P-P-uniform families of networks over $\mathbf{Z}_p$ can be simulated by P-uniform networks. $\square$

7. Universality

We now formalize the notion of an arithmetic network being "the same" as the ground field varies.

Definition 7.1. Let Σ be a set of fields.

(i) An *arithmetic network* $N=(\alpha,\beta)$ *over* Σ consists of a circuit part α and a constant part β. The circuit part α is as in Section 6 (with an arithmetic vertex v having type $\tau(v)=\Sigma$) and has constant nodes labelled by symbols $\lambda_1, \ldots, \lambda_k$. The constant part β assigns constant values to $\lambda_1, \ldots, \lambda_k$ given a particular ground field $F\in\Sigma$, i.e. for each field $F\in\Sigma$, $\beta(F)\in F^k$.

(ii) A description of constants γ and a network description $\overline{N}=(\overline{\alpha},\overline{\gamma})$ is *over* Σ if it is over F for all $F\in\Sigma$. $\square$

Thus for every $F\in\Sigma$, we get from a network description $\overline{N}$ an arithmetic network $N_F=(\alpha,\beta(F))$ over the field F. As inputs to N we consider sequences $(F, a_1, \ldots, a_n, b_1, \ldots, b_m)$, where $F\in\Sigma$, $a_1, \ldots, a_n\in F$, and $b_1, \ldots, b_m\in\mathbf{B}$; and similarly for the output. Thus N computes a function

$$f_N\colon \bigcup_{F\in\Sigma}(\{F\}\times F^n\times\mathbf{B}^m)\rightarrow\bigcup_{F\in\Sigma}(\{F\}\times F^\nu\times\mathbf{B}^\mu)$$

for certain $n, m, \nu, \mu\in\mathbf{N}$.

Some interesting sets of fields are:

1. *Fields* = { all fields },
2. *Real* = { real fields },
3. *Charzero* = { fields of characteristic zero },
4. *Char(p)* = { fields of characteristic p }, where p is a prime number,
5. *Fin* = { finite fields },
6. *Infinite* = { infinite fields }.

(Recall that a field is real if -1 is not a sum of squares. Every subfield of $\mathbf{R}$ is real. To avoid set-theoretic difficulties we assume a fixed universe.)

Definition 7.2. Let Σ be a set of fields. A family $(\overline{\alpha}_i, \overline{\beta}_i)_{i\in\mathbf{N}}$ of encodings of circuit parts α_i and constant part β_i, using only the constant 1 and computable by a Turing machine in time $S(\alpha_i)^{O(1)}$, is a *P-uniform family of arithmetic networks over* Σ. Similarly for *L-uniform*. $\square$

We now have a nice class of "computing devices". We will say "network over Σ" for "L-uniform family of arithmetic networks over Σ". In particular, a network over $\{F\}$ is nothing but a L-uniform family of arithmetic networks over the field F. The other notions of uniformity can be universalized similarly.

Again we have two kinds of semantics for these networks. The value semantics of a network $N=(N_i)_{i\in\mathbf{N}}$ over Σ is a function

$$\phi_N : \bigcup_{F\in\Sigma} (\{F\}\times F^*\times\mathbf{B}^*) \rightarrow \bigcup_{F\in\Sigma} (\{F\}\times F^*\times\mathbf{B}^*) \cup \{\infty\},$$

given by the value semantics of N, and N also computes piecewise rational functions on $F^{n(i)}\times\mathbf{B}^{m(i)}$ for each $F\in\Sigma$ and $i\in\mathbf{N}$. (We resist the temptation to define rational and piecewise rational functions in $\Sigma(x_1, \ldots, x_n)$.)

8. Complexity classes and reductions

We want to define the arithmetic analogues of the Boolean complexity classes $NC^k \subseteq NC \subseteq P$. Cook [1985] gives an overview of the Boolean theory. Our choice of definitions is guided by postulating that three properties carry over from the Boolean setting:

1. the above inclusions should hold,
2. it should be a nontrivial question whether the inclusions are proper,
3. the classes should be closed under a natural notion of "reduction".

Our classes will consist of (piecewise rational) functions rather than of decision problems; in Section 10, we will extend the classes somewhat more.

For a set Σ of fields, Rat_Σ is the set of families of functions $\Sigma^* \times \mathbf{B}^* \rightarrow \Sigma^* \times \mathbf{B}^*$ that can be computed by uniform network families over Σ; in a similar way, one obtains classes of piecewise rational functions. The set $Rat_\Sigma(NON-UNISIGMAORM)$ of all families of piecewise rational functions (satisfying the conditions on input sizes of Definition 6.1) consists precisely of the functions computed by families of arithmetic networks over Σ (Remark 3.3). Our preliminary definition is:

$P_\Sigma = \{f \in Rat_\Sigma$: there exists a network $N = (N_i)_{i \in \mathbf{N}}$ over Σ computing $f = (f_i)_{i \in \mathbf{N}}$, with $S(N_i) = i^{O(1)}$, and thick-deg(ψ_{N_i}) is $i^{O(1)}\}$,

$NC^k_\Sigma = \{f \in P_\Sigma$: there exists a network $N = (N_i)_{i \in \mathbf{N}}$ over Σ computing $f = (f_i)_{i \in \mathbf{N}}$, with $S(N_i) = i^{O(1)}$ and $D(N_i) = O(\log^k i)\}$, for any $k \in \mathbf{N}$,

$NC_\Sigma = \bigcup_{k \geq 0} NC^k_\Sigma$.

One can consider several variants of these classes, e.g., $C(NON-UNIFORM)$ when the uniformity condition is dropped (and C is any of our classes), $C(P-UNIFORM)$ when L-uniformity is replaced by P-uniformity, and $C(ARB-DEG)$ when the condition on the thick degree is dropped. Remark 6.12 (iv) shows that for any prime p, $\Sigma = Char(p)$ and any class C we have $C_\Sigma(P-P-UNIFORM) = C_\Sigma(P-UNIFORM)$.

The condition that the thick degree of the piecewise rational function computed by N_i be polynomial is motivated by the fact that the proper inclusions

$NC_{\Sigma}^{k}(ARB\text{-}DEG) \subsetneq NC_{\Sigma}^{k+1}(ARB\text{-}DEG) \subsetneq NC_{\Sigma}(ARB\text{-}DEG) \subsetneq P_{\Sigma}(ARB\text{-}DEG)$, for $k \in \mathbf{N}$, are trivial if Σ contains an infinite field; for the first one, e.g., consider the family

$$f = (f_i)_{i \in \mathbf{N}} \text{ with } f_i = (F, x_1^{2^{\lceil \log i \rceil^{k+1}}}) \text{ for } F \in \Sigma \text{ and } i \geq 1,$$

and use Proposition 5.5 to show that $f \in NC_{\Sigma}^{k+1} \setminus NC_{\Sigma}^{k}$. The reason to use the thick degree rather than the degree is postulate 3; see Theorem 8.3.

An arithmetic network with $(0,m)$ inputs and $(0,\mu)$ outputs is equivalent to a Boolean circuit with m inputs and μ outputs. If we denote by B the set of families of (oblivious) Boolean functions $f : \mathbf{B}^* \to \mathbf{B}^*$, then $B \subseteq Rat_{\Sigma}$, and

$$NC_{\Sigma}^{k} \cap B = NC^{k},$$

$$NC_{\Sigma} \cap B = NC,$$

$$P_{\Sigma} \cap B = P,$$

where the Boolean classes are defined using L-uniformity and consist of functions rather than the (more usual) languages. Thus our definition satisfies the first two postulates: if in any of the inclusions

$$NC_{\Sigma}^{k} \subseteq NC_{\Sigma}^{k+1} \subseteq NC_{\Sigma} \subseteq P_{\Sigma}$$

equality holds, then also for the corresponding Boolean inclusion.

Thus the "purely Boolean part" of our complexity classes suggests the conjecture of proper inclusions. It is quite a surprise that in the "purely arithmetic part" the inclusions are actually equalities. So we consider the polynomials and rational functions in Rat_{Σ}: For a field F, $F[\,] = \bigcup_{\nu \in \mathbf{N}} F[x_1, x_2, \ldots]^{\nu} \subseteq F(\,) = \bigcup_{\nu \in \mathbf{N}} F(x_1, x_2, \ldots)^{\nu} \subseteq Rat_F$ are the sets of finite sequences of polynomials respectively rational functions over F; $\Sigma[\,]$ and $\Sigma(\,)$ are the corresponding unions over all $F \in \Sigma$. Then $P_{\{F\}}(NON\text{-}UNIFORM) \cap F[\,]$ is essentially Valiant's [1979] class of p-computable polynomials over F. (The only difference is that Valiant's class consists of polynomials, and ours of the corresponding polynomial functions.) The following non-trivial fact shows that any family of rational functions which can be computed in polynomial size, can actually be computed in logarithm-squared depth.

Fact 8.1. For any set Σ of fields, $P_{\Sigma}(NON\text{-}UNIFORM) \cap \Sigma(\,) = NC_{\Sigma}^{2}(NON\text{-}UNIFORM) \cap \Sigma(\,)$. □

Valiant, Skyum, Berkowitz & Rackoff [1983] prove Fact 8.1 for $\Sigma[\,]$. Their proof makes use of Strassen's [1973b] method for avoiding divisions, which shows that the class $P_{\Sigma}(NON-UNIFORM)$ does not change when we disallow divisions in the arithmetic networks. Kaltofen [1986] has extended Strassen's result to rational functions, by showing how to calculate numerator and denominator separately, and then Fact 8.1 follows. (Fact 8.1 is also true for "$R-UNIFORM$" instead of "$NON-UNIFORM$", where "R" stands for random polynomial time; see Borodin, von zur Gathen & Hopcroft [1982] for the case of finite fields.) Miller, Kaltofen & Ramachandran [1986] give a slightly different proof of Fact 8.1.

For a more precise statement of Fact 8.1, let us define R-uniform families $\beta = (\beta_i)_{i \in \mathbf{N}}$ of constants, where a constant β_{ij} can be described by the binary representation of an integer $\gamma_{ij} \in \mathbf{N}$ with $\log(\gamma_{ij}) = i^{O(1)}$. Then $N_i = (\alpha_i, \beta_i)$ computes $f^{(i)}$ if for all sets $B_{ij} \subseteq F$ of size γ_{ij},

$$\phi_{N_i} = f^{(i)}(a)$$

with probability at least 3/4, say, for all inputs a, where β_{ij} is chosen uniformly at random from B_{ij}.

1. Strassen [1973b] proves that any family of polynomials in $P_{\Sigma} \cap \Sigma[\,]$ can be computed by a P-R-uniform family of networks without divisions. (See BGH for finite fields.)
2. Kaltofen [1986] extends this to $\Sigma(\,)$.
3. Valiant, Skyum, Berkowitz & Rackoff [1983] show that any family as in 1. can be computed by an arithmetic network of depth X.

One test for whether our model is reasonable is a simulation by the standard model of (uniform) Boolean circuits over the fields of practical importance in computer algebra, e.g. finite fields or $\mathbf{Q}$. (By Remark 2.4, algebraic computation trees with many leaves fail this test.) This question is discussed in von zur Gathen & Seroussi [1986]. For $F = \mathbf{Q}$, a probabilistic simulation of arithmetic circuits is possible with only polynomial increase in size (von zur Gathen [1985]); if no divisions are present, Jung [1985] has a very efficient simulation. For finite fields, the simulations are satisfactory for size, and if either the characteristic is small or no divisions occur, also for depth.

The notion of "reduction" plays a key role in the classification of Boolean problems according to their sequential complexity (see Garey & Johnson [1979]) and parallel complexity (see Cook [1985]). It comes in (at least) two flavours: "Cook-reduction" and "Karp-reduction". For us, the main consequence of the statement "f is reducible to g" is: if g has a fast algorithm, then so does f. Hence we now introduce the parallel arithmetic analogue of "Cook-reduction".

Definition 8.2. Let Σ be a set of fields, f and $g = (g_i)_{i \in \mathbf{N}}$ families in $Rat_\Sigma(NON-UNIFORM)$, and $C \subseteq Rat_\Sigma$.

(i) Consider the set $\Omega = \Omega_\Sigma^1 \cup \bigcup_{i \in \mathbf{N}} \{g_i\}$ of operations, consisting of our usual operations plus "oracles" for any g_i, and let σ,τ be the appropriate arities and types for Ω. A *reduction* from f to g is a family $(N_i)_{i \in \mathbf{N}}$ of (Ω,σ,τ)-computations computing f. (We are slightly extending our notion of "operation" here, by allowing g_i to have more than one output.)

(ii) The cost of an oracle node g_i, with (n, m) inputs and (ν,μ) outputs, is

$$\delta(g_i) = \left\lceil \log(n+m) + \log\deg(g_i) \right\rceil.$$

We get various resource-bounded reductions by imposing on N any of the above conditions for our complexity classes. In particular, $N = (N_i)_{i \in \mathbf{N}}$ is a NC_Σ^1 $(P-UNIFORM)$-reduction if N is P-uniform and $D(N_i) = O(\log i)$; then also $S(N_i) = i^{O(1)}$. If in addition N is L-uniform, then it is a NC_Σ^1-reduction; this will be our standard notion of reduction. (In a uniform description of a node v with $\lambda(v) = g_i$, we use a symbol for g and encode i in unary.) f is *reducible* to g if there exists a NC_Σ^1-reduction N from f to g; we write $f \leq g$. We say that f and g are *equivalent* (written $f \sim g$) if $f \leq g$ and $g \leq f$. (For example, all members of NC_Σ^1 are equivalent.)

(iii) A function $f \in Rat_\Sigma$ is *hard* for C if $g \leq f$ for all $g \in C$. A function $f \in Rat_\Sigma$ is *complete* for C if f is hard for C and $f \in C$.

(iv) The closure C^* of C is

$$C^* = \{g \in Rat_\Sigma : \exists\, f \in C \quad g \leq f\}.$$

C is closed under reductions if $C^* = C$. □

Theorem 8.3. Let Σ be a set of fields.

(i) Reducibility is a partial order on Rat_Σ.

(ii) Equivalence is an equivalence relation on Rat_Σ.

(iii) $NC_\Sigma^k \subseteq NC_\Sigma \subseteq P_\Sigma$ are all closed under reductions, for any $k \in \mathbf{N}$.

Proof. (i) and (ii): $\leq$ is reflexive: For any $f = (f_{F,i})_{F \in \Sigma, i \in \mathbf{N}} \in Rat_\Sigma$ and $i \in \mathbf{N}$, let N_i consist of one vertex v with $\lambda(v) = f_i$, plus the appropriate input nodes. Then $(N_i)_{i \in \mathbf{N}}$ is a NC_Σ^1-reduction from f to f.

Let f , g , $h \in Rat_\Sigma$ with $f \leq g$ and $g \leq h$. and let $(N_i)_{i \in \mathbf{N}}$ and $(N'_i)_{i \in \mathbf{N}}$ be NC^1_Σ-reductions from f to g and g to h, respectively. Let M_i be obtained from N_i by replacing each occurrence of some g_j by N'_j. We claim that $M = (M_i)_{i \in \mathbf{N}}$ is an NC^1_Σ-reduction from f to h. Let $F \in \Sigma$, n , $m : \mathbf{N} \to \mathbf{N}$ be the input-size functions and $d : \mathbf{N} \to \mathbf{N}$ be the degree function for the family $g = (g_i)_{i \in \mathbf{N}}$ and $n', m', d' : \mathbf{N} \to \mathbf{N}$ be the corresponding functions for h. To simplify notation involving logarithms, we assume in the following that all i and $n(i) + m(i)$ are at least 2. Since both circuits are NC^1_Σ-reductions, there exists a positive constant $c \in \mathbf{R}$ such that for all $i \in \mathbf{N}$, $D(N_i)$, $D(N'_i) \leq c \log i$. We may assume that $c \geq \frac{1}{2}$.

Fix some $i \in \mathbf{N}$, and let P be any path in N_i. The length $\delta(P)$ of P is the sum of the depth costs $\delta(\lambda(v))$ of each vertex v on P, and is bounded from above by $D(N_i)$. Let

$$\Delta = \sum_{\substack{\lambda(v) \in \Omega^1_\Sigma \\ v \text{ on } P}} \delta(\lambda(v))$$

be the depth contribution to P not involving g. After possibly modifying g slightly, we can assume that only g_j with $j \geq 16$ are used on P. Then, using Remark 6.2, we have

$$\Delta + \frac{1}{2} \sum_{\substack{\lambda(v) = g_j \\ j \in \mathbf{N}}} \log j \leq \Delta + \sum_{\substack{\lambda(v) = g_j \\ j \in \mathbf{N}}} \lceil \log(n'(j) + m'(j)) + \log d'(j) \rceil =$$

$$= \sum_{v \text{ on } P} \delta(\lambda(v)) = \delta(P) \leq D(N_i) \leq c \log i.$$

Let $\widetilde{P}$ be any path in M_i. $\widetilde{P}$ is obtained from some path P in N_i by substituting some path $P(v)$ from N'_j for each oracle node v with $\lambda(v) = g_j$ ($j \in \mathbf{N}$). Then

$$\delta(\widetilde{P}) = \Delta + \sum_{\lambda(v) = g_j} \delta(P(v)) \leq \Delta + \sum_{\lambda(v) = g_j} D(N'_j)$$

$$\leq \Delta + \sum_{\lambda(v) = g_j} c \log j \leq 2c \left(\Delta + \frac{1}{2} \sum_{\lambda(v) = g_j} \log j\right) \leq 2c^2 \log i,$$

where all sums are over the vertices v on P. Hence the depth of M_i is $O(\log i)$. The size of M_i is polynomial since at most polynomially many nodes (from N'_j) are being introduced for each of the polynomially many nodes of N_i. Also M is L-uniform, and therefore an NC^1_Σ-reduction from f to h, and so $f \leq h$.

The preceding shows that $\leq$ is reflexive and transitive on Rat_Σ. Hence so is $\sim$. Also, $\sim$ is symmetric by definition.

(iii) Let C be one of these complexity classes, $f = (f_i)_{i \in \mathbf{N}} \in Rat_\Sigma$, $g \in C$, and $N = (N_i)_{i \in \mathbf{N}}$ a NC_Σ^1-reduction from f to g. A calculation as above shows that f can be computed by an appropriate network. It remains to verify that $\deg(f_i) = i^{O(1)}$. (We identify g_i with the piecewise rational function of a network computing g_i, and f_i with $\psi_{N_i} \circ g$.)

There exists some $c \in \mathbf{R}$ such that for all $F \in \Sigma$, $i \in \mathbf{N}$, and Boolean assignments π for N_i with $A_\pi \subseteq F^{n(i)}$ dense (see Section 5), we have $D(\alpha) \leq i^c$, where $\alpha = (N_i)_\pi$ is an arithmetic circuit over F. Fix some c, F, i, π, α. Just as one proves Fact 5.2, we show by induction on the depth $D(v)$ of a node v of α that $\deg\psi_v \leq 2^{D(v)}$. This claim is clear for $D(v) = 0$, and for $\lambda(v) \in \Omega_F^1$. Suppose now that $\lambda(v) = g_j$, that g_j has $n = \sigma_1(g_j)$ (arithmetic) inputs, and that the in-edges to v come from nodes $w_1, \ldots, w_n$. Set $d = \max\limits_{1 \leq k \leq n} D(w_k)$. Then

$$d + \left\lceil \log n + \log\deg(g_j) \right\rceil \leq d + \delta(g_j) = D(v),$$

$$\deg\psi_v \leq \deg g_j \cdot \max_{1 \leq k \leq n} \deg\psi_{w_k} \leq \deg g_j \cdot 2^d \leq 2^{D(v)},$$

using Theorem 4.3. □

Note that with a cost $\delta(g_j) = 1$ – instead of our convention $\log(n+m) + \log(d)$ for an oracle node with (n, m) inputs and of degree d – we would only have that ("$f \leq g$" and $g \in NC_\Sigma^k$) implies $f \in NC_\Sigma^{k+1}$. However, $NC_\Sigma^k(ARB\text{-}DEG)$ is closed under "$NC_\Sigma^1(ARB\text{-}DEG)$-reductions", where the cost of g_i is only $\delta(g_j) = \lceil \log(n+m) \rceil$. Theorem 8.3 also holds for the P-uniform and non-uniform classes.

9. Linear algebra: determinant

In this section, we consider some of the most basic functions from linear algebra, including the determinant, characteristic polynomial, and the solution of nonsingular equations. It turns out that they are all equivalent, and can be solved in depth $O(\log^2 n)$.

Let Σ be a set of fields. (The case usually considered for algorithms is where $\Sigma = \{F\}$ consists of a single field; the point of the new notion of "universality" is to make observations like "(standard) matrix multiplication is the same algorithm for any field" into a precise statement.) Consider the function $\mathrm{DETERMINANT}_\Sigma \in Rat_\Sigma$, where

$$\mathrm{dom}(\mathrm{DETERMINANT}_\Sigma) = \bigcup_{\substack{F \in \Sigma \\ n \geq 1}} F^{n^2},$$

$$\mathrm{DETERMINANT}_\Sigma(a) = \det(a) \text{ for } F \in \Sigma \text{ and } a \in F^{n^2},$$

with appropriate conventions for interpreting $a \in F^{n^2}$ as an $n \times n$-matrix. Similarly we define the following functions from Rat_Σ, for any $n \in \mathbf{N}$ and $F \in \Sigma$.

$\mathrm{MATMULT}_\Sigma$:	computing the coefficients of the product of two $n \times n$-matrices over F;
$\mathrm{CHARPOLY}_\Sigma$:	computing the coefficients of the characteristic polynomial of a square matrix over F;
$\mathrm{MATPOWERS}_\Sigma$:	computing the first n powers $(A, A^2, \cdots, A^n)$ of a matrix $A \in F^{n \times n}$;
$\mathrm{ITMATPROD}_\Sigma$:	computing the "iterated" product $A_1 A_2 \cdots A_n$ of matrices $A_1, A_2, \cdots, A_n \in F^{n \times n}$;
$\mathrm{NONSINGEQ}_\Sigma$:	computing the unique solution $x \in F^n$ to $Ax = b$, given the nonsingular matrix $A \in F^{n \times n}$ and vector $b \in F^n$;
MATINV_Σ:	computing the inverse A^{-1} of a nonsingular matrix $A \in F^{n \times n}$.

Using an obvious algorithm for matrix multiplication, it is clear that $\mathrm{MATMULT}_\Sigma \in NC^1_\Sigma$; similarly $\mathrm{MATPOWERS}_\Sigma$, $\mathrm{ITMATPROD}_\Sigma \in NC^2_\Sigma$. On the other hand, it is not obvious how to perform Gaussian elimination very fast in parallel. Fortunately, very fast parallel algorithms have been found:

Fact 9.1. For any set Σ of fields we have:

(i) $\text{CHARPOLY}_\Sigma \le \text{ITMATPROD}_\Sigma$,

(ii) $\text{CHARPOLY}_\Sigma \in NC_\Sigma^2$. □

(i) was proven by Berkowitz [1984], and later by Chistov [1985]; (ii) then follows from the above remarks. (ii) had previously been shown by Csanky [1976] in characteristic zero, and by Borodin, von zur Gathen & Hopcroft [1982] for arbitrary fields.

We will use the following construction in several places. Let R be a ring, and x an indeterminate over R. For any $A \in R[x]^{n\times n}$, write

$$A = A^{(0)} + A^{(1)}x + A^{(2)}x^2 + \cdots$$

with $A^{(i)} \in R^{n\times n}$ for all i. For any $d \in \mathbf{N}$, the following mapping associates to A a block matrix consisting of $d \times d$ blocks, each of size $n \times n$.

$$\phi_d = \phi_{d,n,R,x}: R[x]^{n\times n} \longrightarrow (R^{n\times n})^{d\times d},$$

$$\phi_d(A) = \begin{bmatrix} A^{(0)} & A^{(1)} & \cdots & A^{(d-1)} \\ 0 & A^{(0)} & \cdots & A^{(d-2)} \\ \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & \cdots & A^{(0)} \end{bmatrix}.$$

Lemma 9.2. ϕ_d is a ring homomorphism with kernel $x^d \cdot R[x]^{n\times n}$. □

Theorem 9.3. For any set Σ of fields $\text{DETERMINANT}_\Sigma$, CHARPOLY_Σ, ITMATPROD_Σ, MATPOWERS_Σ, MATINV_Σ, and NONSINGEQ_Σ are equivalent.

Proof. Leaving away the subscript Σ, we show DETERMINANT $\le$ CHARPOLY $\le$ ITMATPROD $\le$ MATPOWERS $\le$ MATINV $\le$ NONSINGEQ $\le$ DETERMINANT. Let $F \in \Sigma$.

1. DETERMINANT $\le$ CHARPOLY is trivial: $\chi(A,0) = (-1)^n \det A$ and $(-1)^n$ can be computed in depth $\lceil \log n \rceil$.

2. CHARPOLY $\le$ ITMATPROD is Fact 9.1 (i).

3. ITMATPROD $\le$ MATPOWERS: Given $A_1, \cdots, A_n \in F^{n\times n}$, define $B \in (F^{n\times n})^{(n+1)\times(n+1)}$ by

$$B = \begin{bmatrix} I & A_1 & & 0 \\ & & \cdot & \cdot \\ & & & \cdot\ A_n \\ 0 & & & I \end{bmatrix}.$$

Then

$$B^n = \begin{bmatrix} I & * & * & A_1 \cdots A_n \\ & \cdot & & * \\ & & \cdot & * \\ 0 & & & I \end{bmatrix}.$$

4. MATPOWERS $\leq$ MATINV: Given $A \in F^{n \times n}$, let $B = \phi_{n+1}(1-Ax) \in (F^{n \times n})^{(n+1)\times(n+1)}$. Then $\phi_{n+1}((Ax)^{n+1}) = 0$, and

$$B^{-1} = (1-\phi_{n+1}(Ax))^{-1} = \sum_{i \geq 0} \phi_{n+1}(Ax)^i =$$

$$= \phi_{n+1}\Big(\sum_{0 \leq i \leq n} (Ax)^i\Big) = \begin{bmatrix} I & A & A^2 & \cdots & A^n \\ & \cdot & \cdot & \cdot & \cdot \\ & & \cdot & \cdot & A^2 \\ & & & \cdot & A \\ 0 & & & & I \end{bmatrix}.$$

5. MATINV $\leq$ NONSINGEQ: Given a nonsingular $A \in F^{n \times n}$, solve $Ax_i = e_i$ for $1 \leq i \leq n$, where e_i is the i-th column of I and $x \in F^n$. Then x_i is the i-th column of A^{-1}.

6. NONSINGEQ $\leq$ DETERMINANT follows from Cramer's rule. □

Some of these reductions are in Csanky [1976] (in characteristic zero), Borodin, von zur Gathen & Hopcroft [1982], and Cook [1985].

Let Σ be a set of fields. We define the complexity class DET_Σ as

$$DET_\Sigma = \{ g \in Rat_\Sigma : g \leq \text{DETERMINANT}_\Sigma \} = \{\text{DETERMINANT}_\Sigma\}^*.$$

By Fact 9.1 (ii), $DET_\Sigma \subseteq NC_\Sigma^2$. Trivially, DETERMINANT$_\Sigma$ is complete for DET_Σ, and from Theorem 9.3 we conclude:

Corollary 9.4. For any set Σ of fields, DETERMINANT$_\Sigma$, CHARPOLY$_\Sigma$, ITMATPROD$_\Sigma$, MATPOWERS$_\Sigma$, MATINV$_\Sigma$, and NONSINGEQ$_\Sigma$ are complete for DET_Σ. □

Open Question 9.5. For which of the inclusions $NC_\Sigma^1 \subseteq DET_\Sigma \subseteq NC_\Sigma^2$ does equality or inequality hold?

The most plausible conjecture seems $NC_\Sigma^1 \neq DET_\Sigma = NC_\Sigma^2$. Eberly [1984] has shown that the determinant (and characteristic polynomial, inverse of nonsingular matrices) of $n \times n$-matrices of bandwidth m can be computed in depth $O(\log n \log m)$. Let us define for any $m \in \mathbf{N}$

BANDDET$_{\Sigma,m}$: computing the determinant of matrices in $F^{n \times n}$ with bandwidth m and $F \in \Sigma$.

Fact 9.6. Let Σ be a set of fields.

(i) $\mathrm{BANDDET}_{\Sigma,m}$ can be computed by (L-uniform) arithmetic networks over Σ of depth $O(\log n \log\log n)$ and size $n^{O(1)}$.

(ii) If $\Sigma \subseteq \mathit{Infinite}$, then $\mathrm{BANDDET}_{\Sigma,m} \in NC_\Sigma^1$ ($\neq$-*UNIFORM*).

(iii) If $\Sigma \subseteq \mathit{Charzero}$, then $\mathrm{BANDDET}_{\Sigma,m} \in NC_\Sigma^1$.

We want to examine the computational power of computing just a single power of a matrix. To this end, suppose we are given a function $k : \mathbf{N} \to \mathbf{N}$ and define

$\mathrm{ONEPOW}_{\Sigma,k}$: computing $A^{k(n)}$, given $A \in F^{n\times n}$ and $F \in \Sigma$.

Clearly $k(n) = 1$ does not yield an interesting function. However, it turns out that $\mathrm{ONEPOW}_{\Sigma,k}$ is complete for DET_Σ when $k(n)$ is polynomial in n.

Theorem 9.7. Let $\Sigma \subseteq \mathit{Charzero}$. Suppose that $k : \mathbf{N} \to \mathbf{N}$ is computable by a Boolean circuit of logarithmic depth, and that positive $p, q \in \mathbf{Q}$ are given such that $n^p \leq k(n) \leq n^q$. Then $\mathrm{ONEPOW}_{\Sigma,k} \in DET_\Sigma$, and it is complete for DET_Σ under NC_Σ^1 (P-*UNIFORM*)-reductions.

Proof. Let $F \in \Sigma$. As usual, we leave away the subscripts. ONEPOW $\leq$ MATPOWERS: We assume $q \in \mathbf{N}$. Given $A \in F^{n\times n}$, let $j = k(n)$ and let $0 \leq d_0, \cdots, d_q < n$ be the digits of the representation of j in base n, i.e. $d_0 + d_1 n + \cdots + d_q n^q = j$. Clearly $d_0, \cdots, d_q$ can be computed in depth $O(\log n)$. Iterating MATPOWERS q times, compute $(A, A^2, \cdots, A^n)$, $(A^n, A^{2n}, \cdots, A^{n^2})$, $\cdots$, $(A^{n^{q-1}}, A^{2n^{q-1}}, \cdots, A^{n^q})$, i.e. A^{dn^i} for all d and i with $1 \leq d \leq n$, $0 \leq i < q$. Compute $A^j = A^{d_0 + d_1 n + \cdots + d_q n^q} = A^{d_0} A^{d_1 n} \cdots A^{d_q n^q}$. These steps can be performed in (log-space uniform) depth $O(\log n)$.

MATPOWERS $\leq$ ONEPOW: Given $A \in F^{n\times n}$, compute $r = \left\lceil n^{1/p - 1} \right\rceil$. Then $k(rn) \geq (rn)^p . \geq (\left\lceil n^{1/p-1} \right\rceil n)^p \geq n$. Now compute $B = \phi_{n+1}(1 + Ax) \in (F^{n\times n})^{n+1\times n+1}$, and $B^{k(rn)} = \phi_{n+1}((1+Ax)^{k(rn)})$. The first row of this block matrix contains $I, A, A^2, \cdots, A^n$ multiplied by the scalars $\binom{k(rn)}{0}, \binom{k(rn)}{1}, \binom{k(rn)}{2}, \cdots, \binom{k(rn)}{n}$ respectively. These scalars are all nonzero, since char $F = 0$, and can be computed in P-uniform logarithmic depth (Beame, Cook & Hoover [1984]). □

Open Question 9.8. Find sufficient and necessary conditions on Σ and k such that $\mathrm{ONEPOW}_{\Sigma,k}$ is complete for DET_Σ. In particular, is $MATPOW \leq \mathrm{ONEPOW}_k$, say for $k(n) = n$, under log-space uniform log-depth reductions?

10. Linear algebra: rank

The problems of Section 9 all concerned rational functions. In this section, we consider problems from linear algebra involving piecewise rational functions which are not rational functions, such as the rank of matrices.

As usual, Σ is a set of fields, $F \in \Sigma$, and $n \in \mathbf{N}$. We first consider the following auxiliary problem:

ITMATPROD$[X, Y]_\Sigma$: computing the "iterated" product $A_1 A_2 \cdots A_n$ of matrices A_1, $A_2, \cdots, A_n \in F[x, y]^{n \times n}$, with x and y indeterminates over F, and each entry of A_i a polynomial in x and y of total degree less than n.

The indeterminates x and y are only placeholders here; the input is given by elements of F.

Lemma 10.1. For any set Σ of fields, ITMATPROD$[X, Y]_\Sigma \leq$ ITMATPROD$_\Sigma$ $\in DET_\Sigma \subseteq NC_\Sigma^2$.

Proof. Let $F \in \Sigma$, x, y indeterminates over F, and $d, n \in \mathbf{N}$. For the reduction, we apply the mapping of Lemma 9.2 twice to get χ: $F[x, y]^{n \times n} \longrightarrow F^{n^5 \times n^5}$ as the composition

$$F[x, y] \overset{\phi_{n^2, n, F[x], y}}{\longrightarrow} (F[x]^{n \times n})^{n^2 \times n^2} \cong F[x]^{n^3 \times n^3} \longrightarrow$$

$$\overset{\phi_{n^2, n^3, F, x}}{\longrightarrow} (F^{n^3 \times n^3})^{n^2 \times n^2} \cong F^{n^5 \times n^5}.$$

The coefficients of $A_1 \cdots A_n$ (as above) are contained in the matrix $\chi(A_1) \cdots \chi(A_n) \in F^{n^5 \times n^5}$. This shows the reduction; the remainder is clear by Corollary 9.4 and Fact 9.1 (ii). $\square$

Consider the following "combinatorial" functions from linear algebra. As usual, Σ is a set of fields, $F \in \Sigma$, and $n \in \mathbf{N}$.

MATRANK$_\Sigma$:	computing the rank (encoded in unary) of a matrix $A \in F^{n \times n}$;
BASIS$_\Sigma$:	computing an n-bit vector marking some subset of the columns of $A \in F^{n \times n}$ that form a maximal linearly independent set;

SOLVABILITY$_\Sigma$:	computing a bit indicating whether the equation $Ax = b$ has a solution $x \in F^n$, given $A \in F^{n \times n}$ and $b \in F^n$;
MAXMINOR$_\Sigma$:	computing a marking of rows and columns forming a maximal non-singular minor of a matrix $A \in F^{n \times n}$;
IMAGE$_\Sigma$:	computing a basis for the image of F^n under the linear transformation on F^n specified (under the standard basis for F^n) by $A \in F^{n \times n}$;

The following crucial result was shown by Mulmuley [1986].

Fact 10.2. For any set Σ of fields,

(i) MATRANK$_\Sigma \leq$ ITMATPROD$[X, Y]_\Sigma$,

(ii) MATRANK$_\Sigma \in DET_\Sigma \subseteq NC^2_\Sigma$. □

Mulmuley [1986] reduces MATRANK to calculating the characteristic polynomial of a matrix in $F[x,y]^{n \times n}$, with each entry a polynomial of degree less than $3n$. Using Berkowitz' [1984] algorithm (Fact 9.1 (i)), this in turn reduces to ITMATPROD$[X, Y]$. (ii) follows from Lemma 10.1. Ibarra, Moran & Rosier [1980] had proven (ii) for $\Sigma \subseteq Real$, and Borodin, von zur Gathen & Hopcroft [1982] had shown that MATRANK$_F \in$ "random"NC_F^2, for any field F.

An inconvenience is caused by the fact that BASIS, MAXMINOR, and IMAGE are not functions. We want to say that any function satisfying the definition solves the problem. To make this more precise, we have to extend our complexity classes. Let $F \in \Sigma$, $n \in \mathbf{N}$. As an example, BASIS$_F(n)$ consists of all piecewise rational functions $f = (U_i, f_1^{\mathbf{B}}, \ldots, f_n^{\mathbf{B}})_{i \in \mathbf{B}^n}$ with $(n^2, 0)$ inputs and $(0, n)$ outputs solving the basis problem, so that

$$\forall i \in \mathbf{B}^n \ \forall A \in U_i \ \forall j \leq n \quad f_j^{\mathbf{B}} = i_j,$$

and the rows $A_{i^{-1}(\mathbf{T})}$ of A (i.e. the rows A_k with $f_k^{\mathbf{B}} = \mathbf{T}$) are linearly independent of rank equal to rank A. Here we assume some fixed correspondence $F^{n^2} \longrightarrow F^{n \times n}$, so that $\bigcup_{i \in \mathbf{B}^n} U_i = F^{n \times n}$, and A_k is the k-th row of $A \in F^{n \times n}$. Then BASIS$_\Sigma = \bigcup_{\substack{F \in \Sigma \\ n \in \mathbf{N}}}$ BASIS$_F(n)$, and an arithmetic network N over F solves BASIS$_F(n)$ if $\psi_N \in$ BASIS$_F(n)$.

More generally, if N is an arithmetic network over F with (n, m) inputs and (ν, μ) outputs, and Φ a set of piecewise rational functions over F, each with (n, m) inputs and (ν, μ) outputs, and $\psi_N \in \Phi$, then we say that N computes Φ. Now we have the final definition of our complexity classes.

Definition 10.3. Let Σ be a set of fields.

$$Rat_\Sigma = \{\Phi = (\Phi_{F,i})_{\substack{F\in\Sigma\\ i\in\mathbf{N}}}: \exists\, L-\text{uniform network family } N = (N_i)_{i\in\mathbf{N}}$$
$$\text{such thatfor all } F \in \Sigma \text{ and } i \in \mathbf{N}, N_i \text{ over } F \text{ computes } \Phi_{F,i}\},$$
$$P_\Sigma = \{\Phi \subseteq Rat_\Sigma: \exists\, L-\text{uniform network family } N = (N_i)_{i\in\mathbf{N}}, \text{ with}$$
$$N_i \text{ computing } \Phi_{F,i} \text{ over } F, S(N_i) = i^{O(1)}, \text{ and thick-deg}(\psi_{N_i}) = i^{O(1)}\}.$$

NC_Σ^k and NC_Σ are defined analogously. □

We define the complexity class $RANK_\Sigma$ as:

$$RANK_\Sigma = \{ f \in Rat_\Sigma : f \leq \text{MATRANK}_\Sigma \} = \{\text{MATRANK}\}^*.$$

Theorem 10.4. For any set Σ of fields,

(i) MATRANK_Σ, IMAGE_Σ, MAXMINOR_Σ, BASIS_Σ and $\text{SOLVABILITY}_\Sigma$ are complete for $RANK_\Sigma$.

(ii) $RANK_\Sigma \subseteq DET_\Sigma$.

Proof. (i) Leaving away the subscript Σ, we show MATRANK $\leq$ IMAGE $\leq$ MAXMINOR $\leq$ BASIS $\leq$ SOLVABILITY $\leq$ MATRANK. Let $F \in \Sigma$ and $n \in \mathbf{N}$.

1. MATRANK $\leq$ IMAGE is trivial, since rank A = dim im(A).

2. IMAGE $\leq$ MAXMINOR: Given $A \in F^{n\times n}$, a basis for im(A) is obtained by taking the columns of A corresponding to those in any maximal nonsingular minor of A.

3. MAXMINOR $\leq$ BASIS: Given $A \in F^{n\times n}$, obtain a basis $b_1, \ldots, b_k \in F^n$ for the column space of A. Pad $b_1, \ldots, b_k$ with $n-k$ zero columns to form a matrix $B \in F^{n\times n}$. Obtain a basis for the row space of B. The column and row markings so computed mark a maximal nonsingular minor of A.

4. BASIS $\leq$ SOLVABILITY: Given $A \in F^{n\times n}$ with columns $a_1, \ldots, a_n \in F^n$, determine for all i, $1 \leq i \leq n$, whether the system $\sum_{j=1}^{i-1} x_j a_j = a_i$ has a solution $x \in F^n$ (using the appropriately padded $n\times n$-matrices). If the i-th system has no solution, include a_i in the basis.

5. SOLVABILITY $\leq$ MATRANK: $Ax = b$ has a solution if and only if rank A = rank $[A \mid b]$. Adding a zero row to $[A \mid b]$ does not change its rank and makes it square.

(ii) follows from Fact 10.2 (i), Lemma 10.1, and Theorem 9.3. □

Consider the following functions:

INDEPENDENCE$_\Sigma$:	deciding whether $x_1, \ldots, x_i \in F^n$ are linearly independent;
SINGULAR$_\Sigma$:	deciding whether a matrix $A \in F^{n \times n}$ is singular;
EQ$_\Sigma$:	given $A \in F^{n \times n}$ and $b \in F^n$, compute $x \in F^n$ and $c \in \mathbf{B}$ satisfying $[(c = \mathbf{T}) \Longleftrightarrow (\exists \mid y\ Ay = b) \Rightarrow (Ax = b)]$;
NULLSPACE$_\Sigma$:	computing a basis for the null space of $A \in F^{n \times n}$.

Clearly SINGULAR$_\Sigma$ $\leq$ INDEPENDENCE$_\Sigma$ $\leq$ BASIS$_\Sigma$ $\in RANK_\Sigma$.

Open Question 10.5. Is either of SINGULAR$_\Sigma$ or INDEPENDENCE$_\Sigma$ complete for $RANK_\Sigma$?

Theorem 10.6. EQ$_\Sigma$ and NULLSPACE$_\Sigma$ are complete for DET_Σ.

Proof. Leaving away the subscripts, we show EQ $\leq$ NULLSPACE $\leq$ NONSINGEQ $\leq$ EQ. Let $F \in \Sigma$ and $n \in \mathbf{N}$.

1. EQ $\leq$ NULLSPACE: Let $A \in F^{n \times n}$ and $b \in F^n$. For all $x \in F^n$, $Ax = b$ if and only if $[A \mid b]\begin{pmatrix} x \\ y \end{pmatrix} = 0$ with $y = -1$. Determine a basis $z_1, \ldots, z_k \in F^{n+1}$ for the null space of the matrix $[A \mid b]$ made square with a zero row. If $z_{i,n+1} = 0$ for all i, $1 \leq i \leq k$, then no solution $x \in F^n$ to $Ax = b$ exists. Otherwise, let i, $1 \leq i \leq k$, be such that $z_{i,n+1} = w \neq 0$, and define $x = \frac{-1}{w}(z_{i1}, \ldots, z_{in}) \in F^n$. Then $Ax = b$.

2. NULLSPACE $\leq$ NONSINGEQ: Given $A \in F^{n \times n}$, determine a maximal nonsingular $r \times r$-minor M of A, using that MAXMINOR $\leq$ NONSINGEQ by Theorem 10.4 (ii). To simplify notation, we assume that M is the principal $r \times r$-minor of A. Solve the (nonsingular) systems $Mx_i = y_i$ for all i, $r < i \leq n$, where $y_i \in F^r$ consists of the first r rows of the i-th column of A. Then a basis for the null space of A is $((x_i, 0, \ldots, 0, -1, 0, \ldots, 0)_{r < i \leq n})$, where the -1 associated with x_i appears in the i-th row.

3. NONSINGEQ $\leq$ EQ is trivial. □

We have the following hierarchy of complexity classes in linear algebra:

$$NC^1_\Sigma \subseteq \{\text{SINGULAR}_\Sigma\}^* \subseteq \{\text{INDEPENDENCE}_\Sigma\}^* \subseteq RANK_\Sigma \subseteq DET_\Sigma \subseteq NC^2_\Sigma.$$

Kaltofen, Krishnamoorthy & Saunders [1986a, 1986b] give parallel algorithms to compute the Smith normal form, Hermite normal form, elementary divisors, and Jordan normal form of matrices over a field F. Their algorithms use random choices, made uniformly from a (sufficiently large) finite subset of F. (If F is a small finite field, one may have to extend the field, in order to make the algorithms work; see Eberly [1986] for very efficient parallel methods for such extensions.) The methods are "Las Vegas": the output is either the correct answer, or "failure", the latter with controllably small probability - and the algorithms run in depth $O(\log^2 n)$ and size $n^{O(1)}$, for input size n. (An appropriate name for the corresponding complexity class would be $ZP\text{-}NC_F^2$, analoguous to the Boolean probabilistic class ZPP = "zero probability (of incorrectness) polynomial time".) The Jordan normal form requires the calculation of the eigenvalues. If these are given, they compute the usual Jordan normal form. In general, we do not know how to factor polynomials fast in parallel (see Section 11); without the eigenvalues, they compute a variant of the Jordan normal form consisting of blocks corresponding to eigenvalues belonging to the same factors in a "relatively prime basis", a partial factorization of the characteristic polynomial.

Open Question 10.7. For any field F, is any of these problems in NC_F?

11. Polynomials

We first consider arithmetic of polynomials. Most problems are trivially in NC^2, but, in fact, can be proven to be in NC^1. The problem of factoring polynomials is not solvable in our model over $\mathbf{Q}$, for example (and seems hard in parallel in a Boolean model), but efficient algorithms are known over finite fields. To describe their running time appropriately, we introduce a new notion of universality which quantifies the dependence on the characteristic.

Let Σ be a set of fields. We start with the following problems in arithmetic of univariate polynomials, for a given input size $n \in \mathbf{N}$:

$\mathrm{POLYMULT}_\Sigma$:	Computing the product of two polynomials of degree less than n in $F[x]$, where $F \in \Sigma$, x is an indeterminate over F, and a polynomial $f = f_0 + \cdots + f_n x^n \in F[x]$ is represented by its coefficient sequence $(f_0, \ldots, f_n)$.
$\mathrm{ITPOLYPROD}_\Sigma$:	Computing the product of n polynomials, each of degree less than n.
$\mathrm{POLYINTERPOL}_\Sigma$:	Given a list of n pairwise distinct points in F and of n values, compute the coefficients of the unique polynomial of degree less than n that interpolates the given values at the given points.
$\mathrm{POLYDIV}_\Sigma$:	Compute the quotient and remainder of the division of two polynomials of degree less than n.
$\mathrm{POLYINV}_\Sigma$:	Compute the inverse of a polynomial $f \in F[x]$ modulo x^n, assuming $f(0) \neq 0$.
$\mathrm{POLYPOWERS}_\Sigma$:	Compute the first n powers of a polynomial of degree less than n.
ELSYMM_Σ:	Compute the $n+1$ elementary symmetric functions in n elements of F.

Clearly, $\mathrm{POLYMULT}_\Sigma \in NC_\Sigma^1$. All problems are easily reduced to linear algebra, and, by the results of Sections 9 and 10, in $DET_\Sigma \subseteq NC_\Sigma^2$. (Note that $\mathrm{POLYINTERPOL}_F \in NC_F^0$ for any finite field F, since it is a finite function.) Reif [1983] proved that some of these problems are in NC_F^1 if F supports a Fast Fourier Transform. Eberly [1984, 1986] generalized this approach considerably, and proved the following.

Fact 11.1. Let Σ be a set of fields.

(i) $\mathrm{POLYINTERPOL}_{\Sigma} \leq \mathrm{ELSYMM}_{\Sigma} \leq \mathrm{ITPOLYPROD}_{\Sigma}$, and $\mathrm{POLYPOWERS}_{\Sigma} \sim \mathrm{POLYINV}_{\Sigma} \sim \mathrm{POLYDIV}_{\Sigma} \leq \mathrm{ITPOLYPROD}_{\Sigma}$. For $\Sigma = Infinite$, the first three problems are also equivalent under $NC_{\Sigma}^{1}(\neq\text{-}UNIFORM)$ reductions. For $\Sigma = Charzero$, all problems are equivalent. (See Figure 11.1.)

(ii) $\mathrm{ITPOLYPROD}_{Infinite} \in NC^{1}_{Infinite}(L\text{-}L\text{-UNIFORM})$,
$\mathrm{ITPOLYPROD}_{Charzero} \in NC^{1}_{Charzero}(L\text{-}L\text{-UNIFORM}) \cap NC^{1}_{Charzero}(P\text{-UNIFORM})$,
For any $F \in Finite$, $\mathrm{ITPOLYPROD}_{F} \in NC_{F}{}^{1}(L\text{-}L\text{-UNIFORM})$.

(iii) $\mathrm{ITPOLYPROD}_{\Sigma}$ can be computed by an L-uniform family of arithmetic networks over Σ of depth $O(\log n \log\log n)$ and size $n^{O(1)}$. □

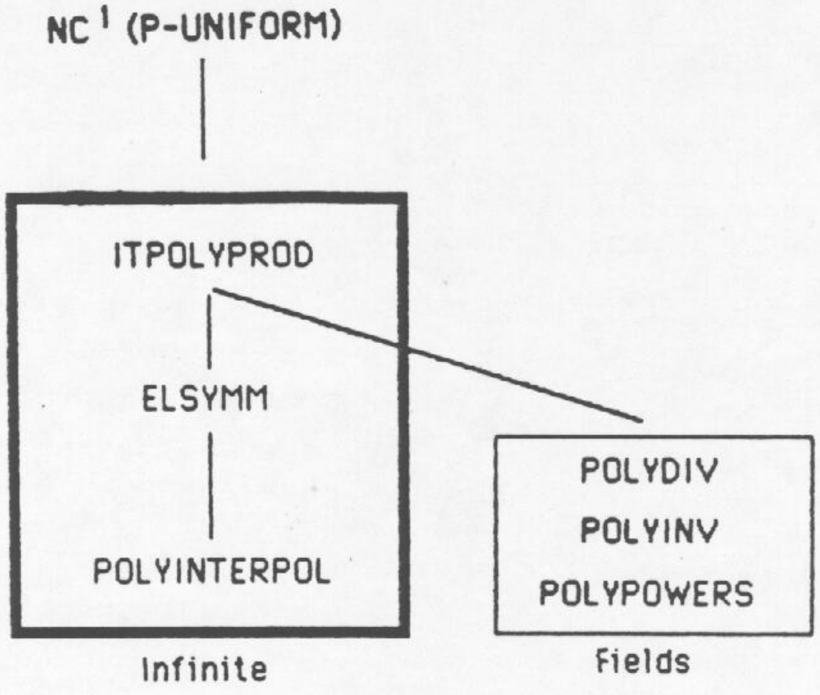

Figure 11.1

The reductions in (i) imply that (ii) and (iii) also hold for the other problems of Figure 11.1. Some of these results were first proven by Reif [1983], under $P\text{-}P$-uniformity and assuming that the ground fields supporsst a Fast Fourier Transform. Bini [1984] showed inversion of triangular Tŏplitz matrices in logarithmic depth, also assuming existence of roots of unity and certain other roots. Later, Bini & Pan [1984, 1986] took up Eberly's approach, showed equivalence of polynomial division with triangular Tŏplitz inversion, and obtained depth $O(\log n \log\log n)$ (and $O(\log n)$ if the field is large enough), ignoring the issue of uniformity for constants; however, they improved previous size bounds. The most ambitious goal for these problems would be to put them into (L-uniform) NC^1_{Fields}; the results show interesting trade-offs between uniformity for the circuit part, uniformity for constants, depth, and universality.

Problems concerning the factorization of univariate polynomials fall into two categories:

1. "rational factorizations", such as gcd and squarefree factorization,
2. factorization of squarefree polynomials.

Fact 11.2. For any set Σ of fields, the following problems are in $DET_\Sigma \subseteq NC^2_\Sigma$:

(i) (von zur Gathen [1984a]) Computing the gcd and lcm of many polynomials, and computing all entries of the Extended Euclidean Scheme of two polynomials.

(ii) (von zur Gathen [1986]) Chinese remainder algorithm, partial fraction decomposition, Padé approximation, Cauchy interpolation, and rational Hermite interpolation.

(iii) (Glover [1984]) Simultaneous Padé approximation.

(iv) (von zur Gathen [1984a]) Computing the squarefree decomposition of univariate polynomials, if $\Sigma \subseteq$ *Charzero*. □

All the above problems can be formulated in terms of linear equations. The interest in the proof of (ii) is in showing that all these problems can be viewed as instances of a general "conversion" problem; that proof does not make use of the results of Section 10 which were not known at the time.

Open Question 11.3. Is any of these problems in NC^1_F, or complete for DET_F?

Our model is not appropriate to address the question of factoring polynomials in characteristic zero, say in $\mathbf{Q}[x]$. No arithmetic network N over $\mathbf{Q}$ can decide on input $a \in \mathbf{Q}$ whether $x^2-a \in \mathbf{Q}[x]$ is irreducible or not: since infinitely many such polynomials are irreducible, there would exist a Boolean assignment π for N such that $A_\pi \subseteq \mathbf{Q}$ is dense (i.e. cofinite), and the output of N is "irreducible" for all $a \in A_\pi$. But $\mathbf{Q} \setminus A_\pi$ is finite, and so N does not work correctly.

Thus one can only hope for a Boolean fast parallel factorization algorithm in $\mathbf{Q}[x]$, with the binary representations of the coefficients as inputs. In this model, the polynomial-time algorithm of Lenstra, Lenstra & Lovász [1982] gives an NC-reduction to the problem of computing short vectors in $\mathbf{Z}$-modules. The most notorious of parallel Boolean problems with a number-theoretic flavour is that of computing the gcd of two integers; this problem is also reducible to the short vector problem (von zur Gathen [1984a]). Kaltofen [1986] proves that absolute irreducibility of bivariate (and multivariate) integer polynomials can be tested by Boolean circuits of poly-logarithmic depth and polynomial size; in our model, his method shows that the problem of absolute irreducibility of bivariate polynomials is in $NC_{Charzero}$.

Let FACTOR_F be the problem of factoring polynomials in $F[x]$, i.e. on input $(a_0, \ldots, a_n) \in F_{n+1}$ output the vector of coefficients of the (monic) irreducible factors of $f = a_0 + \cdots + a_n x^n \in F[x]$, together with Boolean vectors giving their degrees and multiplicities. Then

$$\text{FACTOR}_{\mathbf{Q}} \notin Rat_{\mathbf{Q}},$$

$$\text{FACTOR}_F \in DET_F,$$

for F finite; the latter by the deterministic parallel version of Berlekamp's algorithm (see von zur Gathen [1984a, 1986b]). Suppose F is finite with q elements and characteristic p. In FACTOR_F, p and q are constant. However, it is quite important to consider how the size and depth of networks over *Finite* depends on p and q. For the computation of large powers, say a^{2^n}, "there is no speedup; there is a minimum of n multiplies, and each must be done in sequence" (Jordan [1982]). Although this intuition is correct over infinite fields, the study of the problem showed that it fails over finite fields, and led to the observation that over large finite fields of small characteristic, the model of arithmetic networks over F is not appropriate for parallel solution of some problems:

Fact 11.5. Let p be a prime, $K = GF(p)$, $e \geq 1$, $F = GF(p^e)$, $0 < m < p^e$, and $\pi_F^m\colon F \to F$ with $\pi_F^m(a) = a^m$ for $a \in F$. Then

(i) For any arithmetic circuit over F computing π_F^m with depth d we have $d \geq \min\{\log m, \log(p^e - m)\}$.

(ii) π_F^m can be computed on arithmetic circuits over K of depth $O(\log e \ \log(ep))$ and size $(e \log p)^{O(1)}$. □

(i) is in von zur Gathen [1984b], and (ii) in Fich & Tompa [1985], also in von zur Gathen & Seroussi [1986]. As a special case, we see that if $e/2 \leq p \leq e$ and $p^{e-2} \leq m \leq p^{e-1}$, then π_F^m can be computed by arithmetic circuits over K of size $e^{O(1)}$ and depth $O(\log^2 e)$, but any arithmetic circuit over F computing π_F^m has depth $\Omega(e)$. Large powers are used as subroutines in many algorithms, e.g. in the factoring procedure, and for determining whether a field element is a square (if p is odd).

Arithmetic networks over F are not appropriate to discuss parallel algorithms over F, since the lower bound (i) over F is beaten by the upper bound (ii) over K. Thus, even to solve problems over F, we only consider arithmetic networks over K which have as input and output vectors of elements of K representing an element of F. Furthermore, we want to express the fact that Berlekamp's algorithm works over any finite field, with size proportional to the characteristic p (deterministically) or $\log p$ (probabilistically). In fact, it is even unlikely that $\mathrm{FACTOR}_{Finite} \in Rat_{Finite}$, since over a finite field of characteristic p, Berlekamp's algorithm computes p-th powers; this does not fit into our model.

So now we consider a finite field $F = GF(p^e)$ as given by a prime $p \in \mathbf{N}$ and a monic polynomial $g \in \mathbf{Z}[t]$ of degree e, with all coefficients of g between 0 and $p-1$, such that $g \bmod p \in GF(p)[t]$ is irreducible, and $F = GF(p)[t]/(g \bmod p)$. An element a of F is given by $(a_0, \ldots, a_{e-1}) \in GF(p)^e$, with $a = a_0 + \cdots + a_{e-1}t^{e-1} \bmod (g \bmod p)$. (Any arithmetic circuit over F computing $a \longmapsto a_0$ has depth $\Omega(e)$ (von zur Gathen & Seroussi [1986].)

Definition 11.6. Let $\Sigma \subseteq Finite$ consist of finite fields. A *c-universal network* (for "characteristic-universal") is a deterministic Turing machine M which on input p and g of degree e as above, with integers encoded in binary, and also i encoded in unary, produces a description of an arithmetic network $N_{p,g,i}$ over any $K = GF(p) \in \Sigma$ (intended to solve problems over $GF(p)[t]/(g \bmod p)$). We consider three resource bounds (all "$O(1)$" may depend on i):

(i) If $S(N_{p,g,i}) = (ep)^{O(1)}$ and $D(N_{p,g,i}) = (\log(ep))^{O(1)}$, then M is P-L-universal (for "size polynomial and depth poly-logarithmic in p").

(ii) If M is P-L-universal and $S(N_{p,g,i}) = (e \log p)^{O(1)}$, then M is L-L-universal.

(iii) If M is L-L-universal and $D(N_{p,g,i}) = (\log(e \log p))^{O(1)}$, then M is L-LL-universal. $\square$

Note that M is not an arithmetic network in our sense, but each $(N_{p,g,i})_{i \in \mathrm{N}}$ is a family of arithmetic networks over $GF(p)$. In each case, the size is polynomial and the depth poly-logarithmic in e. We then obtain complexity classes like NC_{Finite} (L-L-universal), consisting of problems f for which there exists an L-L-universal M such that for all $F \in Finite$, $N_{p,g,i}$ is L-L-uniform, has size and degree $i^{O(1)}$ and depth $(\log i)^{O(1)}$, and solves f for $GF(p^e) = GF(p)[t]/(g \bmod p)$. For a finite field F we have that e.g. NC_F (P-L-universal) $\subseteq NC_F$.

Fact 11.7. Let $\Sigma = Finite$.

(i) $\mathrm{POLYMULT}_\Sigma \in NC^1_\Sigma$ (L-LL-universal).

(ii) The powering function π with $\pi^m_F(a) = a^m$ for $a \in F \in \Sigma$ is in NC^2_Σ (L-L-universal).

(iii) The problem of computing the squarefree factorization of univariate polynomials is in NC_Σ^2 (L –L –universal).

(iv) $\mathrm{FACTOR}_\Sigma \in NC_\Sigma^3$ (P –L –universal).

(v) $\mathrm{FACTOR}_\Sigma \in ZP\text{–}NC_\Sigma^2$ (L –L –universal).

(vi) Multivariate polynomials over Σ can probabilistically be tested for irreducibility in depth $(\log np)^{O(1)}$ and size $(n \log p)^{O(1)}$. □

(See Section 10 for "ZP –NC".) For input size i, π has as inputs the binary representations of $m \leq 2^i$ and $a \in F$. (iii), (iv), and (v) follow from von zur Gathen [1984a], (v) from von zur Gathen [1985a]. Here, n is the input size, where the polynomial may be given either by all its coefficients up to its degree ("dense representation"), or only its nonzero coefficients ("sparse representation"), or even by a straight-line program. Instead of just testing for irreducibility, one can even determine the degrees and multiplicities of the irreducible factors (the "factorization pattern").

Open Question 11.8. Can one factor multivariate polynomials over finite fields, given by straight-line programs, in depth and size as in (vi)?

A more difficult problem than factoring polynomials is to determine the solution set of a system of polynomial equations. The more general problem of ideal membership is complete for exponential space (Mayr & Meyer [1982]). According to Chistov & Grigoryev [1984], the solution set can be determined in quasi-polynomial sequential time $2^{(\log n)^{O(1)}}$. For parallel algorithms, Ben-Or, Kozen & Reif [1984] prove that formulas of the theory of real closed fields can be decided in double-exponential time and exponential space, and also in exponential parallel time. For a fixed number of variables, the problem is in Boolean NC. Ben-Or, Feig, Kozen & Tiwari [1986] show that the roots of a univariate polynomial with only real roots can be approximated by Boolean circuits of depth $(\log n)^{O(1)}$ and size $n^{O(1)}$, where n is the input length plus the required precision.

Permutation groups offer a variety of algebraic questions, which can only be addressed by Boolean computations; see McKenzie & Cook [1983] for parallel aspects.

12. Conclusion

The area of parallel algorithms for algebraic problems is young and lively. We have described a framework in which most algorithms can be discussed. Many interesting problems have been classified as to their relative and absolute complexity. The subject is teeming with open questions, some of which may be relatively easy.

Beyond the mostly elementary problems mentioned here, a more general question is what parts of algebra (or mathematics) can be put into NC (or P).

Acknowledgements

This material is based on lectures given at University of Toronto in the spring of 1984. Many thanks go to all participants for lively discussions, in particular to Steve Cook and Charlie Rackoff, to Rod Glover and Jim Hoover for preparing formal course notes, and to Wayne Eberly.

References

P.W. Beame, S.A. Cook and H.J. Hoover, Log depth circuits for division and related problems. Proc. 25th Ann. IEEE Symp. Foundations of Computer Science, Singer Island FL, 1984, 1-6. To appear in SIAM J. Comput.

M. Ben-Or, Lower bounds for algebraic computation trees. Proc. 15th Ann. ACM Symp. Theory of Computing, Boston MA, 1983, 80-86.

M. Ben-Or, E. Feig, D. Kozen, and P. Tiwari, A fast parallel algorithm for determining all roots of a polynomial with real roots. To appear in Proc. 18th Ann. ACM Symp. Theory of Computing, Berkeley CA, 1986.

M. Ben-Or, D. Kozen, and J. Reif, The complexity of elementary algebra and geometry. Proc. 16th Ann. ACM Symp. Theory of Computing, Washington DC, 1984, 457-464.

S.J. Berkowitz, On computing the determinant in small parallel time using a small number of processors. Information Processing Letters **18** (1984), 147-150.

D. Bini, Parallel solution of certain Toeplitz linear systems. SIAM J. Computing **13** (1984), 268-276.

D. Bini and V. Pan, Fast parallel algorithms for polynomial division over arbitrary field of constants. Nota interna, Dipartimento di Informatica, Università di Pisa, 1984.

D. Bini and V. Pan, Polynomial division and its computational complexity. Tech. Rep. TR 86-2, Computer Science Department, SUNY Albany NY, 1986.

A. Borodin, On relating time and space to size and depth. SIAM J. Comput. **6** (1977), 733-744.

A. Borodin, J. von zur Gathen, and J. Hopcroft, Fast parallel matrix and GCD computations. Information and Control **52** (1982), 241-256.

R.P. Brent, The parallel evaluation of general arithmetic expressions. J. ACM **21** (1974), 201-206.

A.L. Chistov, Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic. Proc. Int. Conf. Foundations of Computation Theory, Springer Lecture Notes in Computer Science **199**, 1985, 63-69.

A.L. Chistov and D.Yu. Grigoryev, Fast decomposition of polynomials into irreducible ones and the solution of systems of algebraic equations. Soviet Math. Dokl **29** (1984), 380-383.

S.A. Cook, Towards a complexity theory of synchronous parallel computation. In: *Logic and Algorithmic*, Symposium in honour of Ernst Specker, Enseignement Mathématique **30** (1982), 99-124.

S.A. Cook, A taxonomy of problems with fast parallel algorithms. Information and Control **64** (1985), 2-22.

L. Csanky, Fast parallel matrix inversion algorithms. SIAM J. Comput. **5** (1976), 618-623.

W. Eberly, Very fast parallel matrix and polynomial arithmetic. Proc. 25th Ann. IEEE Symp. Foundations of Computer Science, Singer Island FL, 1984, 21-30. Tech. Rep. #178/85, Department of Computer Science, University of Toronto.

W. Eberly, Very fast parallel polynomial arithmetic. Preprint, University of Toronto, Canada, 1986.

F. Fich and M. Tompa, The parallel complexity of exponentiating polynomials over finite fields, Proc. 17th Ann. ACM Sympos. Theory of Computing, Providence RI, 1985, 38-47.

M.R. Garey and D.S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness.* W.H. Freeman, San Francisco, 1979.

J. von zur Gathen [1984a], Parallel algorithms for algebraic problems. SIAM J. Comput. **13** (1984), 802-824.

J. von zur Gathen [1984b], Computing powers in parallel. Proc. 25th Ann. IEEE Symp. Foundations of Computer Science, Singer Island FL, 1984, 31-36. To appear in SIAM J. Computing.

J. von zur Gathen, Irreducibility of multivariate polynomials. J. Computer System Sciences **31** (1985), 225-264.

J. von zur Gathen [1986a], Representations and parallel computations for rational functions. SIAM J. Comput **15** (1986), 432-452.

J. von zur Gathen [1986b], Factoring polynomials and primitive elements for special primes. To appear in Theor. Computer Science.

J. von zur Gathen and G. Seroussi, Boolean circuits versus arithmetic circuits. Proc. 6th Int. Conf. Computer Science, Santiago, Chile, 1986, 171-184.

R. Glover, Simultaneous Padé approximation. M. Sc. Thesis, University of Toronto, September 1984.

P. Griffiths and J. Harris, *Principles of algebraic geometry.* John Wiley & Sons, New York, 1978.

J. Heintz, Definability bounds of first order theories of algebraically closed fields. Extended Abstract in Fundamentals of Computation Theory, L. Budach, ed., Akademie-Verlag, Berlin, 1979, 160-166.

L. Hyafil, On the parallel evaluation of multivariate polynomials. SIAM J. Computing **8** (1979), 120-123.

O.H. Ibarra, S. Moran and L.E. Rosier, A note on the parallel complexity of computing the rank of order n matrices. Inf. Proc. Letters **11** (1980), 162.

T.L. Jordan, A guide to parallel computation and some Cray-1 experiences. In: *Parallel computations*, ed. by G. Rodrigues, Academic Press, New York, 1982, 1-50.

E. Kaltofen, Fast parallel absolute irreducibility testing. J. Symb. Computation **1** (1985), 57-67.

E. Kaltofen, Uniform closure properties of p-computable functions. Proc. 18th Ann. ACM Symp. Theory of Computing, Berkeley CA, 1986, 330-337.

E. Kaltofen, M. Krishnamoorthy, and B.D. Saunders [1986a], Fast parallel computation of Hermite and Smith forms of polynomial matrices. To appear in SIAM J. Algebraic and Discrete Methods, 1986.

E. Kaltofen, M. Krishnamoorthy, and B.D. Saunders [1986b], Fast parallel algorithms for similarity of matrices. Proc. ACM Symp. Symbolic and Algebraic Computation, Waterloo, Ontario, 1986, 65-70.

H.T. Kung, New algorithms and lower bounds for the parallel evaluation of certain rational expressions and recurrences. J. ACM **23** (1976), 252-261.

A.K. Lenstra, H.W. Lenstra, and L. Lovász, Factoring polynomials with rational coefficients. Math. Ann. **261** (1982), 515-534.

E.W. Mayr and A.R. Meyer, The complexity of the word problems for commutative semigroups and polynomial ideals. Adv. Math. **46** (1982), 305-329.

P. McKenzie and S.A. Cook, The parallel complexity of the Abelian permutation group membership problem. Proc. 24th Ann. IEEE Symp. Foundations of Computer Science, Tucson AZ, 1983, 154-161.

G.L. Miller, E. Kaltofen, and V. Ramachandran, Efficient parallel evaluation of straight-line code. Proc. Aegean Workshop on Computing, VLSI Algorithms and Architectures, Attica, Greece, 1986.

D.E. Muller and F.P. Preparata, Restructuring of arithmetic expressions for parallel evaluation. J. ACM **23** (1976), 534-543.

K. Mulmuley, A fast parallel algorithm to compute the rank of a matrix over an arbitrary field. Proc. 18th Ann. ACM Symp. Theory of Computing, Berkeley CA, 1986, 338-339.

J. Reif, Logarithmic depth circuits for algebraic functions. Proc. 24th Ann. IEEE Symp. Foundations of Computer Science, Tucson AZ, 1983, 138-145.

W.L. Ruzzo, On uniform circuit complexity. J. Computer System Sciences **22** (1981), 365-383.

C.P. Schnorr, An extension of Strassen's degree bound. SIAM J. Comput. **10** (1981), 371-382.

V. Strassen, Berechnung und Programm. I.

V. Strassen [1973a], Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten. Numer. Math. **20** (1973), 238-251. Acta Inf. **1** (1972), 320-335.

V. Strassen [1973b], Vermeidung von Divisionen. J. reine u. angew. Math. **264** (1973), 182-202.

V. Strassen, The Computational Complexity of Continued Fractions. SIAM J. Comput. **12** (1983), 1-27.

V. Strassen, Algebraische Berechnungskomplexität. In: *Perspectives in Mathematics*, Birkhäuser Verlag, Basel, 1984.

L.G. Valiant, Completeness classes in algebra. Proc. 11th Ann. ACM Symp. Theory of Computing, Atlanta GA, 1979, 249-261.

L.G. Valiant, Computing multivariate polynomials in parallel. Information Processing Letters **11** (1980), 44-45, and **12**, 54.

L. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff, Fast parallel computation of polynomials using few processors. SIAM J. Comput. **12** (1983), 641-644.