J. Symbolic Computation (1987) **4,** 137–172

Feasible Arithmetic Computations: Valiant's Hypothesis

JOACHIM VON ZUR GATHEN

Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A4, Canada

(*Received* 12 *November* 1985)

An account of Valiant's theory of *p*-computable versus *p*-definable polynomials, an arithmetic analogue of the Boolean theory of *P* versus *NP*, is presented, with detailed proofs of Valiant's central results.

1. Introduction

The most important development in complexity theory during the past decade is the theory of *P* versus *NP* (Cook, 1971; Karp, 1972; see Garey & Johnson, 1979, for an account). The class *P* of "feasible" problems—solvable in polynomial time—is compared with the (probably) much larger class *NP*, which seems to include most computational problems that come up in practice. Some insight on Cook's fundamental hypothesis "$P \neq NP$" is provided by "polynomial-time reductions". This approach singles out the "hardest" problems in *NP*—one of them is the satisfiability problem for propositional formulas—and shows that Cook's hypothesis is actually equivalent to proving that satisfiability is not in *P*.

This tutorial presents Valiant's (1979*a*, 1982) arithmetic analogue of the Boolean theory. The objects now are not Boolean functions or decision problems, but (families of) multivariate polynomials over an arbitrary ground field. The notion of *p*-projection (somewhat more stringent than reduction) now singles out the hardest polynomials among the *p*-definable ones (corresponding to problems in *NP*); the permanent is such a "*p*-complete" family of polynomials (in characteristic different from two). (The terms "Cook's hypothesis" and "Valiant's hypothesis" were coined by Strassen, 1986.) The Boolean problem of computing integer permanents is $\#P$-complete; at least as hard as *NP*-complete (Valiant, 1979*b*).

Now Valiant's central hypothesis is that some *p*-definable polynomials are not *p*-computable (corresponding to problems not in *P*). Valiant's hypothesis implies that the permanent is not *p*-computable (in characteristic different from two). As early as 1913, Pólya and Szegö considered the problem whether the permanent can be expressed as the determinant of a matrix. Valiant's hypothesis could now be proven by an appropriate answer to this classical question: If the permanent is not a *qp*-projection of the

This tutorial was presented at the DMV Seminar "Die Berechnungskomplexität algebraischer und numerischer Probleme", Düsseldorf, Germany, September 1984, jointly with Volker Strassen. 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.

0747–7171/87/050137+36 $03.00/0

determinant, then the permanent is not *qp*-computable (and hence not *p*-computable). Here "*qp*" stands for "quasi-polynomial time" $2^{\log n^{O(1)}}$, rather than polynomial time $n^{O(1)}$ for input size n.

In the wake of *NP*-completeness, a number of techniques have been devised to deal with *NP*-complete problems in some sense different from the immediate one: approximate solutions, probabilistic algorithms and estimates, average case analysis, and, on a different level, relativisation techniques.

Valiant's arithmetic theory might have impact on *computer algebra* in several ways. First, once the inherent difficulty of some problems is accepted, one may try to devise techniques similar to the ones mentioned above that circumvent the problems that are unavoidable in the straightforward approach.

Second, it may shed light on difficulties that have been observed in practice. An example is the computation of iterated partial derivatives: expressions have been noted to become unmanageably large after some partial derivatives are taken, and Caviness & Epstein (1978) give examples involving exponentials where the memory required increases exponentially. Valiant (see section 4) shows that iterated partial derivatives of harmless (i.e. *p*-computable) polynomials may become *p*-complete.

Third, the whole development makes it clear that "straight-line programs" may be an advantageous data structure for representing multivariate polynomials. In theory, this approach is strictly more powerful than the popular "sparse representation", because the latter is a special case to which the former can be efficiently converted (Kaltofen, 1986), but the straight-line representation can handle special polynomials with a very large number of nonzero coefficients, e.g. determinants of polynomial matrices. Von zur Gathen (1985) (for testing irreducibility) and Kaltofen (1986) (for gcd's and factoring) have shown theoretical feasibility of this approach, by solving standard problems of symbolic manipulation in this data structure in random polynomial time. In the sparse representation, irreducible factors may have a length which is more than polynomial in the input size (von zur Gathen & Kaltofen, 1985). Kaltofen's powerful results show that this unpleasant phenomenon does not happen in the straight-line representation. Freeman *et al.* (1986) report implementation of a system based on this approach, in LISP with an interface to MACSYMA. Many symbolic manipulation packages, such as MACSYMA, only allow expressions as representations of polynomials. Some newer languages, such as Maple, have a "remember" option which is closer to using programs as representations.

Fourth, it is a well-known experience in "structured vs general computation" (Borodin, 1982) that the additional structure in algebraic computation (over Boolean computation) may give us the power to prove lower bounds for which we lack the tools in the Boolean context. In our case, it is a tantalising problem whether Valiant's arithmetic analogue of "$P \neq NP$" is easier to prove than Cook's hypothesis concerning Boolean problems. As noted above, Valiant's hypothesis would follow from the appropriate answer to a classical mathematical question, namely whether the permanent can only be expressed as the determinant of a matrix with huge increase in size. This might be proved by purely algebraic means, or, over finite fields, by combinatorial methods. Von zur Gathen (1987) goes a tiny step in this direction.

The tutorial is organised as follows. In section 2, we consider several measures of computational complexity for polynomials: straight-line program complexity (with non-scalar and division-free variants), expression size, and depth (=parallel time for straight-line programs). The *p*-computable polynomials are defined as those that have "feasible" straight-line programs (i.e. of polynomial length) and reasonable (polynomially-bounded)

degree. Under the more generous constraint of quasi-polynomial length, programs and arithmetic expressions (=formulas) become equally powerful.

In section 3, we prove "universality" of the determinant: every polynomial of *qp*-bounded expression size can be expressed as the determinant of a small matrix. An emphasis of this tutorial are detailed proofs in this section, and section 5.

Section 4 introduces the notion of *p*-definable polynomials, and gives several characterisations. If, on input the binary encoding of an exponent vector, a deterministic polynomial-time Turing machine can decide whether the coefficient of the corresponding monomial is zero or one (and no other coefficients occur), then the polynomial is *p*-definable. The set of *p*-definable polynomials is closed under operations such as taking partial derivatives, integrals, and coefficients, while the *p*-computable ones are not, under Valiant's hypothesis.

Section 5, the *pièce de résistance*, shows that the permanent (in characteristic different from two) and the Hamiltonian cycle polynomial are *p*-complete.

All results of this tutorial are due to Valiant, unless otherwise attributed.

2. The Model of Computation

Let F be a field, and $x_1, \ldots, x_n$ indeterminates over F. Following Strassen (1972), we consider arithmetic (straight-line) *programs over F* (or "arithmetic circuits") with inputs $x_1, \ldots, x_n$. Formally, such a program is a sequence $P = (P_1, \ldots, P_l)$ of either arithmetic operations $P_i = (\omega_i, j_i, k_i)$ or inputs $P_i \in F \cup \{x_1, \ldots, x_n\}$. For each $i \leqslant l$ of the first case, $\omega_i \in \{+, -, *, /\}$ is a binary operation, and $j_i, k_i < i$ are numbers of previous instructions. We will usually only consider the *result sequence* $(u_1, \ldots, u_l)$ of such a program P, defined by $u_i = P_i$ in the second case, and

$$u_i = u_{j_i} \omega_i u_{k_i}$$

otherwise. We stipulate that no division by the rational function zero occurs. The *length* of the program is the number of arithmetic operations used. The program *computes* the rational function $u_l \in F(x_1, \ldots, x_n)$. For a given rational function $f \in F(x_1, \ldots, x_n)$, the *complexity* $L_F^*(f)$ of f is the smallest length of programs that compute f.

EXAMPLE 2.1. $P_1 = x_1$, $P_2 = x_2$, $P_3 = (*, 1, 1)$, $P_4 = (*, 2, 2)$, $P_5 = (+, 3, 4)$ describes a program of length 3, with result sequence $(x_1, x_2, x_1^2, x_2^2, x_1^2 + x_2^2)$. The program computes $f = x_1^2 + x_2^2$, so that $L_F^*(f) \leqslant 3$, for any field F. □

The notion of "polynomial time" is a stable and mathematically satisfying property of algorithms which, in theoretical computer science, seems a good approximation to the distinction between the "feasible" algorithms—those that can be executed in practice on reasonably large inputs, and for which increased computer speed increases the range of solvable problems correspondingly—and the infeasible ones. The goal of the material presented here is to investigate this notion in an algebraic context.

Thus we consider families $f = (f_n)_{n \in \mathbb{N}}$ of polynomials

$$f_n \in F[x_1, \ldots, x_{v(n)}],$$

and want to know: Is $L_F^*(f_n)$ a polynomial function of n? Trivially, $L_F^*(f_n)$ is not polynomial for examples like $f_n = x^{2^{2^n}}$ or $f_n = x_1 + \ldots + x_{2^n}$. We will see that the

permanent family—which has small degree and few variables—is a candidate for also not having polynomial complexity. Some results take on a nicer form if we are somewhat more generous and allow "quasi-polynomial time", i.e. $2^{\log n^{O(1)}}$ operations for input size n.

In order to develop the theory of p-computable polynomials, we introduce three other measures of complexity: division-free complexity, depth, and expression size. Although of some interest by themselves, in the context of this exposition they only serve as technical means to derive results about the natural measure L^*.

As an aside, we first want to mention the *non-scalar complexity* $L_F^{ns}(f)$, which will not be used in the sequel (Ostrowski, 1954; see Strassen, 1984, for an overview). Here in a program as above, u_{j_i} and u_{k_i} are allowed to be arbitrary linear combinations over F of $1, x_1, \ldots, x_n, u_1, \ldots, u_{i-1}$. Thus only non-scalar multiplications and divisions contribute to the cost of a program. As an example, suppose that there exists $i \in F$ such that $i^2 = -1$. Then the program with

$$u_1 = (x_1 + ix_2) * (x_1 - ix_2)$$

computes $f = x_1^2 + x_2^2$, and thus $L_F^{ns}(x_1^2 + x_2^2) = 1$. However, if -1 has no square root in F, then no such one-liner exists, and $L_F^{ns}(x_1^2 + x_2^2) = 2$.

If $F \subseteq K$ are fields and $f \in F[x_1, \ldots, x_n]$, then $L_K^*(f) \leqslant L_F^*(f)$; similarly for L^{ns}. The above example shows that inequality may hold. The non-scalar complexity is a very satisfying measure, where powerful tools like Strassen's (1973*a*) degree bound sometimes fulfil the complexity theorist's dream: asymptotically matching upper and lower bounds.

It is a pleasant surprise that divisions actually do not help much, at least for rational functions of small degree:

PROPOSITION 2.2. *Let $f = g/h \in F(x_1, \ldots, x_n)$ be a rational function, and $g, h \in F[x_1, \ldots, x_n]$ relatively prime polynomials of degree at most d. If f can be computed by a program of length l, then g and h can be computed by a program without divisions of length polynomial in d and l.* □

This was proven by Strassen (1973*b*) for the case $h = 1$, and by Kaltofen (1986) in general; Borodin *et al.* (1982) deal with finite F. As an example, Gaussian elimination can be converted to a division-free program of size $O(n^5)$ computing the determinant of an $n \times n$-matrix.

In the sequel, we will only consider the computation of polynomials. As a consequence of Proposition 2.2, we may consider only division-free programs (where $\omega_i \in \{+, -, *\}$, in the above notation), and define the division-free complexity $L_F(f)$ as the smallest size of division-free programs that compute f. We usually leave away the qualifier "division-free".

We first note that for "general" multivariate polynomials, L_F is exponential in the degree d and the number n of variables (Strassen, 1974).

THEOREM 2.3. *Let F be an infinite field, $d, n \in \mathbb{N}$, $P \subseteq F[x_1, \ldots, x_n]$ the vector space of all polynomials of degree at most d, and*

$$p = \dim_F P = \binom{n+d}{d}.$$

Then

(*i*) $\forall f \in P \quad L_F(f) \leqslant 3p$.
(*ii*) $\exists f \in P \quad L_F(f) \geqslant p$.

PROOF.

(i) One can compute each of the p monomials of degree at most d in length p: first those of degree 0, then degree 1, etc. Then f can be computed as the sum of its monomials in the stated length.
(ii) The set of all programs with a fixed format of length l, but allowing arbitrary choices of constants in the programs, forms a vector space over F of dimension at most l. A dimension argument proves (ii). □

(ii) actually holds for "almost all" polynomials, in the strong sense of algebraic geometry. In the following lower bounds, we only consider $d = n$ for simplicity.

THEOREM 2.4. *Let F be an arbitrary field, and $n \geqslant 2$. There exists a polynomial $f \in F[x_1, \ldots, x_n]$ of degree at most n such that $L_F(f) \geqslant 2^{2n}/10n^2$.*

PROOF. For infinite F, the claim follows from Theorem 2.3(ii). For finite F, one uses a counting argument similar to the Shannon–Lupanov lower bound for Boolean circuits (see Savage, 1976). □

Using the more refined methods of number theory and algebraic geometry in Strassen (1974) and Heintz & Sieveking (1980) one can show that there exist polynomials f of degree n in n variables with all coefficients either 0 or 1 such that $L_F(f) \geqslant 2^n/n$. One also obtains lower bounds for specific polynomials with integer or algebraic coefficients. As an example, let $d \geqslant 2$, $n \geqslant 3$, and for $i = (i_1, \ldots, i_n) \in \mathbb{N}^n$, let

$$\beta(i) = \sum_{1 \leqslant j \leqslant n} i_j d^{j-1} \in \mathbb{N}$$

be the integer with "d-ary representation" i. Let

$$f = \sum_{i \in \{0, \ldots, d-1\}^n} \sqrt{\beta(i)}\, x_1^{i_1} \cdots x_n^{i_n} \in \mathbb{C}[x_1, \ldots, x_n].$$

Then

$$L_{\mathbb{C}}(f) > \left(\frac{d^n}{n \log d}\right)^{1/2}.$$

Continuing the general development, we define the *depth* of a program $(P_1, \ldots, P_l)$ as the length d of a longest chain $1 \leqslant i_1 < i_2 < \cdots < i_d$ of "consecutive steps", i.e. where $\omega_{i_l} \in \{+, -, *, /\}$ and either j_{i_l} or k_{i_l} is from $\{i_1, \ldots, i_{l-1}\}$ for every l, $2 \leqslant l \leqslant d$. The depth is the parallel execution time of the program. For a polynomial f, the depth $D_F(f)$ is the smallest depth of (division-free) programs that compute f.

EXAMPLE 2.5.

1. Let $f = x_1 \cdot x_2 \cdot \cdots \cdot x_n$. Then $L_F(f) = n - 1$, and $D_F(f) = \lceil \log n \rceil$. Here, as in the remainder of this tutorial, log means $\log_2$.
2. Let F be infinite, and $f = x_1^{2^n}$. Then $L_F(f) = D_F(f) = n$. □

The set of arithmetic *expressions* (or "formulas") over $F \cup \{x_1, \ldots, x_n\}$ is defined inductively as follows. Every element from $F \cup \{x_1, \ldots, x_n\}$ is an expression, and if φ_1 and φ_2 are expressions, then so are $(\varphi_1 + \varphi_2)$ and $(\varphi_1 * \varphi_2)$. The size of an expression φ is the number of $+$ and $*$ used to build it. φ obviously represents a polynomial $\mathrm{val}(\varphi) \in F[x_1, \ldots, x_n]$. For a polynomial f, the expression size $E_F(f)$ is the smallest size of expressions φ with $\mathrm{val}(\varphi) = f$.

EXAMPLE 2.6. The two expressions

$$((x_1 * x_1)+(x_2 * x_2)) \quad \text{and} \quad ((x_1+(i * x_2)) * (x_1+(-i) * x_2))$$

both represent $f = x_1^2 + x_2^2$ (assuming that $i^2 = -1$). The sizes are 3 and 5, respectively. □

An analogue of Proposition 2.2 holds, stating that one would not gain a lot by allowing divisions (Brent, 1974; see Borodin & Munro, 1975, section 6.3).

PROPOSITION 2.7. *For any $f \in F[x_1, \ldots, x_n]$ of degree d, we have*

(i) $D_F(f) \leqslant L_F(f) \leqslant E_F(f)$,
(ii) $D_F(f) = O(\log(d \cdot L_F(f)) \cdot \log d)$,
(iii) $\log E_F(f) \leqslant D_F(f) = O(\log E_F(f))$.

PROOF. (i) is trivial, (ii) is in Hyafil (1979) and (iii) in Brent (1974). (The big O in (iii) is meant to imply the existence of a universal constant c such that $D_F(f) \leqslant c \log E_F(f)$ provided $E_F(f) \geqslant 2$, and similarly for (ii).) □

In fact, a much stronger statement than (ii) is true: f can be computed by a program of simultaneous length $O(d^6(L_F(f))^3)$ and depth $O(\log(d \cdot L_F(f)) \cdot \log d)$ (Valiant *et al.*, 1983; Miller *et al.*, 1986, prove a variant of this result). This is an instance where the additional structure of arithmetic computations (here: degree) yields results which we do not have for Boolean computations; the Boolean analogue "$P = NC^2$" is unlikely to be true. Von zur Gathen (1986) discusses general parallel arithmetic computations.

In order to study asymptotic complexity, we consider *families $f = (f_n)_{n \in \mathbb{N}}$ of polynomials* with $f_n \in F[x_1, \ldots, x_{v(n)}]$. For the determinant, we have $v(n) = n^2$, and variables x_{ij} with $1 \leqslant i, j \leqslant n$.

EXAMPLE 2.8.

$$\begin{aligned}
\text{sum:} \quad & \mathrm{SUM}_n = x_1 + \cdots + x_n, \\
\text{product:} \quad & \mathrm{PROD}_n = x_1 \cdots x_n, \\
\text{power sum:} \quad & \mathrm{POWERSUM}_n = x_1^n + \cdots + x_n^n, \\
\text{determinant:} \quad & \mathrm{DET}_n = \det((x_{ij})_{1 \leqslant i, j \leqslant n}), \\
\text{permanent:} \quad & \mathrm{PER}_n = \mathrm{per}((x_{ij})_{1 \leqslant i, j \leqslant n}).
\end{aligned}$$

Recall that

$$\mathrm{per}((x_{ij})) = \sum_{\sigma \in \mathrm{Sym}_n} x_{1,\sigma(1)} \cdots x_{n,\sigma(n)},$$

where the sum is over the symmetric group Sym_n of all $n!$ permutations of $\{1, \ldots, n\}$.

To include the elementary symmetric functions $\sigma_{i,n} \in F[x_1, \ldots, x_n]$ (having degree i, for $0 \leqslant i \leqslant n$), we have to change their natural enumeration by pairs (i, n) into a linear enumeration. One possibility is to use

$$\mathrm{ELSYMM}_m = \sigma_{i,n} \in F[x_1, \ldots, x_n] \subseteq F[x_1, \ldots, x_m]$$

for the unique i, n such that

$$m = \binom{n+1}{2} + i.$$

Thus the first polynomials are

$$1, 1, x_1, 1, x_1 + x_2, x_1 x_2, \ldots$$

Permanents were introduced by Binet (1813) and Cauchy (1813). They occur naturally in several areas: in combinatorics, they describe the number of perfect matchings in a graph, and the number of solutions to the related *problème des ménages* and *problème des rencontres*. An account of the exciting story of van der Waerden's conjecture on permanents of doubly stochastic matrices is given in van Lint (1982). In geometry, the metric on symmetric powers of matrices is described by permanents (see Blokhuis & Seidel, 1984). The encyclopedic volume by Minc (1978) gives an overview of the history and classical results about permanents, together with a complete bibliography up to 1977.

Both the determinant and permanent are special cases of the following construction. Let $G \subseteq \mathrm{Sym}_n$ be a subgroup of the symmetric group, $\chi : G \to \mathbb{C}$ a character, and $A \in \mathbb{C}^{n \times n}$ an $n \times n$-matrix over $\mathbb{C}$. Then

$$d_\chi^G(A) = \sum_{\sigma \in G} \chi(\sigma) A_{1,\sigma(1)} \cdots A_{n,\sigma(n)}$$

defines the *Schur function* belonging to χ. When $G = \mathrm{Sym}_n$ and χ is irreducible, then d_χ^G is the *immanent* defined by χ. For the permanent, we take $\chi = 1$, and for the determinant, $\chi = \mathrm{sign}$. (The complexity of immanents is hard to determine; see Hartmann (1983) for some results.)

A function t: $\mathbb{N} \to \mathbb{N}$ is *p-bounded* (respectively *qp-bounded*) if there exists a constant c such that $t(n) \leqslant n^c$ ($t(n) \leqslant 2^{(\log n)^c}$, respectively) for all $n \geqslant 3$. Thus p-bounded stands for "polynomially-bounded", and a qp-bounded (for "quasi-polynomial") function is allowed to grow faster than any polynomial, but much slower than any exponential function 2^{n^ε} for $\varepsilon > 0$. (We ignore the values of $t(n)$ for $n \leqslant 2$.)

A family $f = (f_n)_{n \in \mathbb{N}}$ of polynomials with $f_n \in F[x_1, \ldots, x_{v(n)}]$ is

p-computable,
p-expressible,
qp-computable,
qp-expressible,

respectively, if and only if $v(n)$ and $\deg(f_n)$ are p-bounded functions of n, and

$L_F(f_n)$ is p-bounded,
$E_F(f_n)$ is p-bounded,
$L_F(f_n)$ is qp-bounded,
$E_F(f_n)$ is qp-bounded,

respectively. We consider a family "feasible" if it is p-computable; this is the analogue for polynomials of the Boolean class P. The notions with "qp" have nicer stability properties than "p" (Proposition 2.10, Corollaries 3.2 and 5.5).

Restricting the degree is quite reasonable over infinite fields, e.g. over $\mathbb{Q}$, where the binary representation of the value of a polynomial like x^{2^n} has exponential length even for small inputs. In a different setting—over varying finite fields—natural problems like the trace, testing for quadratic residuosity, or factoring polynomials, lead to polynomials of large degree, which can nevertheless be computed efficiently (von zur Gathen & Seroussi, 1986).

EXAMPLE 2.9. The families SUM, PROD, and POWERSUM are p-expressible. DET is p-computable, using Gaussian elimination and Proposition 2.2. That ELSYMM is

p-expressible follows from Reif (1986) if F has a primitive nth root of unity for infinitely many n, and from Eberly (1984) for arbitrary infinite F. □

The fastest known algorithm to compute the permanent is due to Ryser (1963) and based on a principle of inclusion and exclusion: Let $n \in \mathbb{N}$, $N = \{1, \ldots, n\}$, and $x = ((x_{ij})_{i,j \in N})$ be an $n \times n$-matrix of indeterminates over F. For $I \subseteq N$, let y_I be the matrix obtained from x by replacing the columns i with $i \in I$ by zero columns, and q_I the product of the row sums of y_I. Then

$$\operatorname{per} x = \sum_{0 \leqslant i < n} (-1)^i \sum_{\substack{I \subseteq N \\ \# I = i}} q_I.$$

This formula shows that $L_F(\mathrm{PER}) = O(n2^n)$ and $E_F(\mathrm{PER}) = O(n^2 2^n)$. In particular, it is not known whether PER is qp-computable.

We remark that for depth or formula size, the non-scalar model L_F^{ns} may not be appropriate. One example is an algorithm of Kung (1976) which computes x^n over an algebraically closed field with non-scalar depth two. As another example, the sum formula for the $n \times n$-determinant gives a program of non-scalar depth $\lceil \log_2 n \rceil$ (and exponential length). In our model, it is known but not trivial that the determinant can be computed in depth $O((\log n)^2)$ and polynomial length (Csanky, 1976; Borodin *et al.*, 1982; Berkowitz, 1984; Chistov, 1985). Proposition 2.7 (iii) and the above results imply the best known upper bound $2^{O((\log n)^2)}$ on the expression size of the determinant; polynomial expression size would be a major (unexpected) result. Therefore, it seems that the nice relation $D \approx \log E$ may not hold for the non-scalar model. (Intuitively, it seems unfair to neglect the huge fan-in in the non-scalar program of depth $\log n$ for the determinant.) Kalorkoti (1985) shows that $E(\mathrm{DET}_n) = \Omega(n^3)$.

PROPOSITION 2.10. *Let f be a family of polynomials. Then*

$$f \text{ } p\text{-expressible} \Rightarrow f \text{ } p\text{-computable} \Rightarrow$$
$$f \text{ } qp\text{-computable} \Leftrightarrow f \text{ } qp\text{-expressible}.$$

PROOF. The only non-trivial claim, namely "⇒" in the double implication, follows from Proposition 2.7 (ii) and (iii). □

In section 4 we will introduce the class of "p-definable" families, to which all "naturally occurring" families of polynomials seem to belong. However, Valiant conjectures that many p-definable families are not feasible:

VALIANT'S HYPOTHESIS. *Over any field, there exist p-definable families of polynomials which are not p-computable* (Fig. 1).

As for other similar conjectures in theoretical computer science, our civilisation does not seem ready for proofs of Valiant's hypothesis.

A fruitful approach to understanding such conjectures has been to introduce a notion of "reduction" (here: projection) and then exhibit specific "complete" candidates, which are "hardest" within their class. We will see that PER is p-complete, and thus Valiant's hypothesis is equivalent to the conjecture that PER is not p-computable (over a field of characteristic different from two; in characteristic two, PER = DET is p-computable).

Going back to Proposition 2.10, we note that the second implication is not reversible.

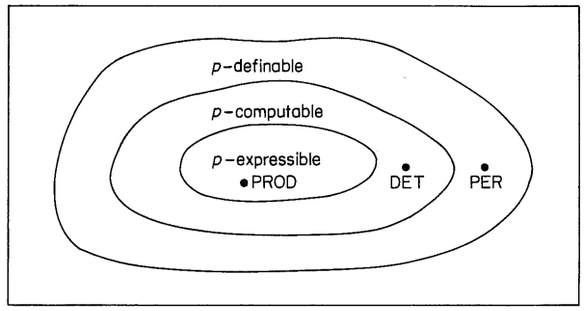


Fig. 1.

For $n \in \mathbb{N}$, let $m_n = \lceil \log n \rceil^2$, $f_n = \mathrm{PER}_{m_n}$ = permanent of a $m_n \times m_n$-matrix of $(m_n)^2$ indeterminates. Then $L_F(f_n) = O(\log^2 n\, 2^{\log^2 n})$, and $f = (f_n)_{n \in \mathbb{N}}$ is qp-computable. If $L_F(\mathrm{PER}_n) \geqslant 2^n$, then $L_F(f_n) \geqslant 2^{\log^2 n}$, and f is not p-computable. This example hinges on Valiant's hypothesis. Scaling the polynomials of Theorem 2.4 as above gives polynomials which are provably qp-computable and not p-computable.

It would be interesting to know whether the first implication can be reversed. This seems unlikely, since an affirmative answer would imply

$$f \text{ } p\text{-computable} \Rightarrow D_F(f_n) = O(\log n),$$

a result which I consider unlikely to be true.

One sees how much easier life gets by using "quasi-polynomial" rather than "polynomial": we do not have to distinguish between "expressible" and "computable" any more.

OPEN QUESTION 2.11.

(*i*) *Prove that there exists a p-computable family which is not p-expressible.*

(*ii*) *Prove that* DET *is not p-expressible. (Or prove that it is p-expressible. But not both, please! Or prove that this question is independent of Zermelo–Fraenkel.)* □

DEFINITION 2.12. Let F be a field.

(i) Let $f \in F[x_1, \ldots, x_n]$ and $g \in F[x_1, \ldots, x_m]$. f is a *projection* of g if and only if there exist $a_1, \ldots, a_m \in F \cup \{x_1, \ldots, x_n\}$ such that

$$f = g(a_1, \ldots, a_m).$$

(ii) Let $f = (f_n)_{n \in \mathbb{N}}$ and $g = (g_m)_{m \in \mathbb{N}}$ be families of polynomials over F, with $f_n \in F[x_1, \ldots, x_{v(n)}]$ and $g_m \in F[x_1, \ldots, x_{w(m)}]$, and $t : \mathbb{N} \to \mathbb{N}$. f is a *t-projection* of g if and only if for every n there exists $m \leqslant t(n)$ such that $v(n), w(m) \leqslant t(n)$ and f_n is a projection of g_m.

(iii) Let f, g be as in (ii). f is a *p-projection* of g if and only if f is a t-projection of g for some p-bounded t. Similarly, we define qp-projection. □

One can consider more generous notions of reduction, e.g. where in a computation for f one is allowed to use values of g at various inputs. The motivation to use projections here is that the results of sections 3 through 5 hold even for this stringent notion, and thus are stronger than when formulated with more liberal variants. In the analogy of "Turing-" or

"Cook-"reductions, one would be allowed to evaluate a subroutine for g at several inputs in order to compute f. In a projection, one has a package for g, and only plugs in constants or inputs to compute f.

EXAMPLE 2.13.

1. PROD and SUM are p-projections of ELSYMM.
2. PROD and SUM are p-projections of DET. This is obvious for PROD, and will follow from Theorem 3.1 for SUM.
3. PROD and SUM are not t-projections of each other for any t.
4. $x_1+x_2x_3$ is not a projection of any symmetric polynomial (Fich *et al.*, 1986). □

PROPOSITION 2.14.

(*i*) *The composition of two p-projections is a p-projection. Similar with "qp" for "p".*
(*ii*) *The classes of p-computable and p-expressible families are closed under p-projections. Similar with "qp" for "p".*

PROOF (easy, but uninspiring). (i) Suppose that f, g, h are families of polynomials, f is a p-projection of g, and g a p-projection of h. For simplicity, assume that

$$f_n, g_n, h_n \in F[x_1, \ldots, x_n];$$

in general, one can renumber the families and introduce dummy members to achieve this assumption, with only polynomial growth. Thus there exist functions $t, s, m, i: \mathbb{N} \to \mathbb{N}$, constants c, d, and

$$a_{n1}, \ldots, a_{n,m(n)} \in F \cup \{x_1, \ldots, x_n\},$$
$$b_{j1}, \ldots, b_{j,i(j)} \in F \cup \{x_1, \ldots, x_j\}$$

for all $n, j \in \mathbb{N}$, such that

$$\forall\, n \geqslant 3\ m(n) \leqslant t(n) \leqslant n^c,$$
$$\forall\, j \geqslant 3\ i(j) \leqslant s(j) \leqslant j^d,$$
$$f_n = g_{m(n)}(a_{n1}, \ldots, a_{n,m(n)}),$$
$$g_j = h_{i(j)}(b_{j1}, \ldots, b_{j,i(j)}).$$

Then

$$f_n = h_{i(m(n))}(b(a)_{n1}, \ldots, b(a)_{n,i(m(n))}),$$

where $b(a)$ is obtained by "substituting a in b", i.e.

$$b(a)_{np} = \begin{cases} b_{m(n),p} & \text{if } b_{m(n),p} \in F, \\ a_{nq} & \text{if } b_{m(n),p} = x_q \text{ for some } q,\ 1 \leqslant q \leqslant m(n). \end{cases}$$

If $n, m(n) \geqslant 3$, then

$$i(m(n)) \leqslant (n^c)^d = n^{cd}.$$

Set

$$e = \max\{cd, \log_3(i(0)), \log_3(i(1)), \log_3(i(2))\}.$$

Then

$$i(m(n)) \leqslant n^e$$

for all $n \geqslant 3$, and therefore f is a p-projection of h. The argument for "qp" is essentially the same, using that

$$2^{(\log(2^{(\log n)^c}))d} = 2^{(\log n)^{cd}}.$$

(ii) Suppose that g is p-computable, and f a p-projection of g. Use notation t, m, c, a as

in (i), and let $u_j = (u_{j1}, \ldots, u_{j,l(j)})$ be the result sequence of a computation for g_j, with $l: \mathbb{N} \to \mathbb{N}$ p-bounded. Using a_{ni} for every occurrence of x_i in $u_{m(n)}$ $(1 \leqslant i \leqslant m(n))$, we get a computation for f_n. The length of that computation is again p-bounded, and f therefore p-computable.

The argument applies *mutatis mutandis* to the other claims of (ii). □

3. Universality of the Determinant

In this section we show that the methods of linear algebra are, at least in principle, sufficient to devise feasible algorithms whenever they exist. More precisely, any p-expressible family of polynomials is a p-projection of the determinant; similarly for "qp".

Valiant complements his result by showing similar properties for the Boolean problems of transitive closure (for parallel algorithms) and of linear programming (for sequential algorithms), again under a restrictive notion of "projection". Jung (1985) shows that the discrete DET is complete for the probabilistic space class *PrSPACE* $(\log n)$, and that PER is complete for *PrTIME* $(n^{O(1)})$.

THEOREM 3.1. *Let $f \in F[x_1, \ldots, x_n]$ have expression size e. Then f is a projection of* DET_{2e+2}.

COROLLARY 3.2. *Let $f = (f_n)$ be a family of polynomials with $v(n)$ and* $\deg f_n$ *p-bounded. Then the following are equivalent:*

(*i*) *f is qp-computable,*
(*ii*) *f is qp-expressible,*
(*iii*) *f is a qp-projection of* DET. □

PROOF OF THEOREM 3.1. Let R be the set of expressions over $X = F \cup \{x_1, \ldots, x_n\}$, and denote by $X^{s \times t}$ the set of all $s \times t$-matrices with entries from X. We will define a mapping

$$\mu: R \to \{\text{square matrices over } X\}$$

with the following properties, for all $\varphi \in R$:

(1.1) $\mathrm{val}(\varphi) = \det \mu(\varphi)$.
(1.2) If φ has size e, then $\mu(\varphi)$ has size $s \times s$ with $s = 2e+2$.
(1.3) There exist $A \in X^{(s-1)\times(s-1)}$, $\alpha \in X^{1\times(s-1)}$, $\beta \in X^{(s-1)\times 1}$, with $s = 2e+2$ as in (1.2), such that A is upper triangular with 1's on the diagonal and $\mu(\varphi)$ is shown in Fig. 2.

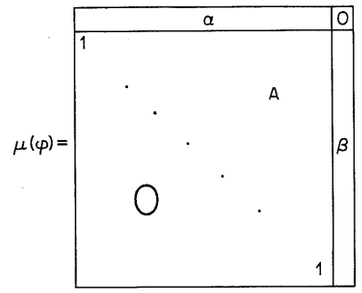


Fig. 2.

(1.1) and (1.2) are sufficient to prove the theorem; (1.3) is a technical requirement for the construction. The definition of μ proceeds by induction along the construction of φ. Without (1.3), one can trivially use diagonal blocks that simulate *; the non-trivial part is to find a construction that also works for $+$.

Case 1: $\varphi \in X$. Then

$$\mu(\varphi) = \begin{array}{|cc|}\hline \varphi & 0 \\ 1 & 1 \\ \hline \end{array} \in X^{2\times 2}.$$

Case 2: $\varphi = (\varphi_1 \times \varphi_2)$. Define $\mu(\varphi)$ as in Fig. 3.

Property (1.3) is clear. Using e and s as in (1.2), we have

$$e = e_1 + e_2 + 1,$$

and the size s of $\mu(\varphi)$ is

$$s = s_1 + s_2 = (2e_1+2)+(2e_2+2) = 2e+2;$$

(1.2) follows. Since $\mu(\varphi)$ is block lower triangular, (1.1) is clear, too.

Case 3: $\varphi = (\varphi_1 + \varphi_2)$. Assume that $\mu(\varphi_k)$ is as shown in Fig. 4(a) for $k = 1, 2$. First consider M as in Fig. 4(b) where

$$s = (s_1-1)+(s_2-1)+1 = 2e_1+1+2e_2+1+1$$

is odd. Let $I = \{1, \ldots, s\}$. Which permutations $\sigma \in \mathrm{Sym}\, I$ contribute to $\det M$? Suppose first that $\sigma(1) < s_1$, and that the diagonal product

$$p_\sigma(M) = \prod_{1 \leqslant i \leqslant s} M_{i,\sigma(i)}$$

is non-zero. Then

$$\sigma(\{1, \ldots, s_1\}) \subseteq \{1, \ldots, s_1 - 1, s\},$$

and, having equal cardinality, these two sets are equal. It follows that also

$$\sigma(\{s_1+1, \ldots, s\}) = \{s_1, \ldots, s-1\},$$

and σ lives on the diagonal of A_2. σ induces a permutation τ on $\{1, \ldots, s_1\}$ (by taking the sth column of M as the s_1 th), and $p_\sigma(M)$ equals the corresponding diagonal product $p_\tau(\mu(\varphi_1))$. For the signs we have

$$\operatorname{sign} \sigma = (-1)^{s-1-s_1+1} \cdot \operatorname{sign} \tau = (-1)^{2e_2+1} \operatorname{sign} \tau = -\operatorname{sign} \tau.$$

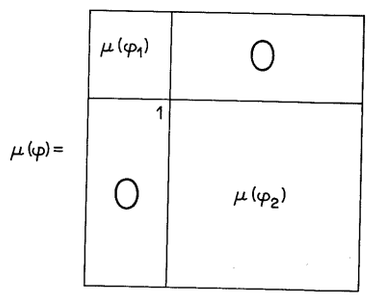


Fig. 3.

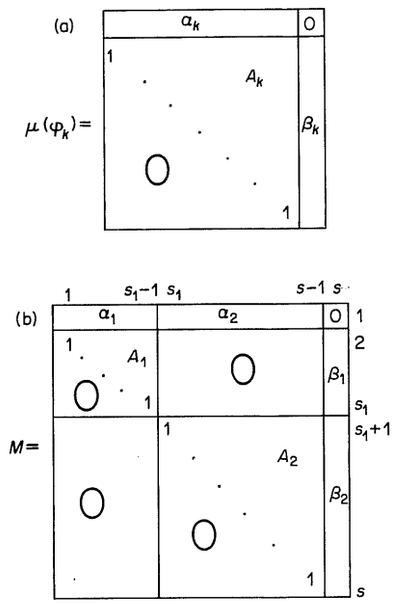


Fig. 4.

It follows that all $\sigma \in \mathrm{Sym}\, I$ with $\sigma(1) < s_1$ contribute a total of $-\det \mu(\varphi_1)$ to $\det M$. Similarly, the $\sigma \in \mathrm{Sym}\, I$ with $s_1 \leqslant \sigma(1) < s$ contribute a total of $-\det \mu(\varphi_2)$ to $\det M$. Hence,

$$\det M = -(\det \mu(\varphi_1) + \det \mu(\varphi_2)) = -\mathrm{val}(\varphi).$$

We now get $\mu(\varphi)$ by adding to M a last row and a last but one column, consisting of all zeros except a one at the intersection (Fig. 5). Then $\det \mu(\varphi) = -\det M$, and (1.1) holds. Property (1.3) is clear, and the size of $\mu(\varphi)$ is

$$s+1 = 2e_1+2e_2+4 = 2e+2. \quad \square$$

REMARK 3.3. The number of non-zero entries of $\mu(\varphi)$ is at most $4e+3$. Neither this number nor the size $2e+2$ are minimal; Valiant (1979*a*) constructs $\mu(\varphi)$ with property (1.1) and size $e+2$. $\square$

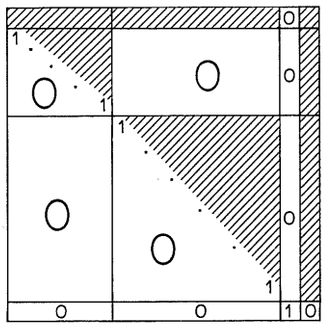


Fig. 5.

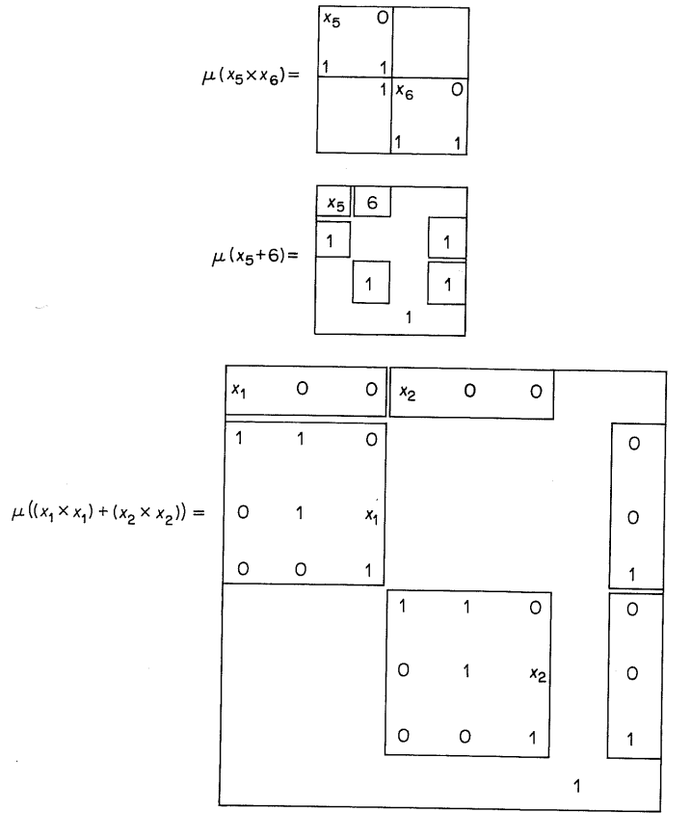


Fig. 6.

We give three examples of this construction in Fig. 6; all blank entries are zero.

One consequence of Theorem 3.1 is the "universality of the determinant", i.e. the fact that *every* polynomial over F is a projection of DET_n for some n. We now show that also PER is universal; this result will be used in the next section.

PROPOSITION 3.4. *Let $f \in F[x_1, \ldots, x_n]$ have expression size e. Then f is a projection of PER_{2e+2}.*

PROOF. Using the notation of the previous proof, we show that $\mathrm{val}(\varphi) = \mathrm{per}(\mu(\varphi))$. It is sufficient to prove that if a diagonal product $p_\sigma(\mu(\varphi))$ is non-zero, then σ is even. With what we know about μ, the proof is an easy induction. □

Although the determinant and permanent share this absolute property of "universality", the computational implications differ wildly. As pointed out in Corollary 3.2, universality of the determinant (in the specific form stated here) allows us to identify the algorithmic notion of "qp-computable" with the algebraic notion of "qp-projection of DET". The universality of the permanent is pointless under this aspect, due to the notorious lack of feasible computations for the permanent.

It is natural to ask for families having the property stated in Corollary 3.2 for the

determinant, but with "p-expressible" or "p-computable" instead of "qp-computable". Fich *et al.* (1986) give the following results.

Define $f_n \in F[x_1, \ldots, x_n]$ to be zero when n is not a power of 4, and otherwise inductively by $f_1 = x_1$ and

$$f_n = f_{n/4}(x_1, \ldots, x_{n/4}) * f_{n/4}(x_{n/4+1}, \ldots, x_{n/2}) + (f_{n/4}(x_{n/2+1}, \ldots, x_{3n/4}) * f_{n/4}(x_{3n/4+1}, \ldots, x_n).$$

Thus f_n is the polynomial computed by the complete binary tree with n leaves and alternating layers of * and +. Then the family $f = (f_n)_{n \in \mathbb{N}}$ is p-expressible, and any p-expressible family of polynomials is a p-projection of f.

No family consisting of symmetric polynomials has this property: $x_1 + x_2 x_3$, e.g. is not a projection of a symmetric polynomial over any field F. For $F = \mathbb{Z}_2$, this contrasts with the result of Skyum & Valiant (1985) that for every polynomial there exists a projection of a symmetric polynomial that assumes the same values everywhere. If $\sigma_{2,4} = x_1 x_2 + x_1 x_3 + \ldots$ is the elementary symmetric function of degree 2 in 4 variables, then

$$\forall\, a_1, a_2, a_3 \in \mathbb{Z}_2 \quad a_1 + a_2 a_3 = \sigma_{2,4}(a_1, a_1, a_2, a_3).$$

If F has characteristic zero, then also $x_1 - x_2$ is not a projection of a symmetric polynomial.

Let $n \in \mathbb{N}$, and consider the following arithmetic circuit with inputs $x_1, \ldots, x_n$ and $t(d, e, i, j, k)$ for $1 \leqslant d, e, i, j, k \leqslant n$, and layers $1, \ldots, n$. In each layer d, n polynomials $s_{d,i}$ are computed, all homogeneous in all variables $x_1, \ldots, x_n$ and $t(d, e, i, j, k)$, and homogeneous of degree d in $x_1, \ldots, x_n$ alone. In layer 1, these are

$$s_{1,i} = \sum_{1 \leqslant j \leqslant n} t(1, 1, i, j, 1) \cdot x_j$$

for all i, and in layer d

$$s_{d,i} = \sum_{\substack{1 \leqslant j,k \leqslant n \\ 1 \leqslant e < d}} t(d, e, i, j, k) \cdot s_{e,j} \cdot s_{d-e,k}$$

for $1 \leqslant i \leqslant n$. Setting

$$f_n = t_0 + \sum_{1 \leqslant d \leqslant n} s_{d,1},$$

we get a p-computable family $f = (f_n)_{n \in \mathbb{N}}$ such that any p-computable family of polynomials is a p-projection of f.

OPEN QUESTION 3.5. *Find more "natural" families that are "universal" as the above examples. Is the determinant universal for p-computable polynomials (under p-projections)?*

4. *P*-definable Families

In this section, we give a number of equivalent definitions of the very general class of "p-definable" polynomials. The p-complete polynomials are the computationally hardest among these. Valiant's hypothesis is the conjecture that some p-definable polynomials are not p-computable. It holds if and only if each p-complete polynomial is not p-computable. Finally, we find that the p-definable polynomials are closed under more natural operations than the p-computable ones.

DEFINITION 4.1.
(i) If g and h are two families of polynomials over F, then g *defines* h if and only if for all $n \in \mathbb{N}$

$$h_n = \sum_{e \in \{0,1\}^n} g_n(e)\mathbf{x}^e.$$

Here $\mathbf{x}^e$ denotes the monomial $x_1^{e_1} \cdots x_n^{e_n}$.
(ii) A family f of polynomials over F is *p-definable* if and only if there exists a p-expressible family g over F such that f is a p-projection of the family h defined by g. □

In the above definition, h consists of polynomials whose coefficients can be computed fast. We can think of the arithmetic expression for g as an efficient algorithm which, on input an exponent vector e, produces the coefficient of $\mathbf{x}^e$ in h_n. However, h itself may or may not be efficiently computable. Since such h are restricted to have degree at most 1 in each variable, we allow p-projections for the notion of "p-definable".
To motivate the definition, we note the following equivalent formulations.

THEOREM 4.2. *Let f be a family of polynomials over F. The following are equivalent.*

(i) f is p-definable.
(ii) f is p-definable as above, but with g allowed to be p-computable.
(iii) There exists a p-computable family g and a p-bounded $m\colon \mathbb{N} \to \mathbb{N}$ such that for all n

$$f_n = \sum_{e_{n+1},\ldots,e_{m(n)} \in \{0,1\}} g_{m(n)}(x_1, \ldots, x_n, e_{n+1}, \ldots, e_{m(n)}).$$

(iv) As (iii), but with g required to be p-expressible.
(v) There exists a p-computable g and p-bounded $m\colon \mathbb{N} \to \mathbb{N}$ such that f is a p-projection of $h = (h_n)_{n \in \mathbb{N}}$ with

$$h_n = \sum_{e_{n+1},\ldots,e_{m(n)} \in \{0,1\}} g_{m(n)}(x_1, \ldots, x_n, e_{n+1}, \ldots, e_{m(n)}) \cdot x_{n+1}^{e_{n+1}} \cdots x_{m(n)}^{e_{m(n)}}. \quad \square$$

The proof of this theorem is non-trivial (Valiant, 1979*a*, 1982).

COROLLARY 4.3. *Every p-computable family is p-definable.*

PROOF. Use Theorem 4.2 (iii), with $f = g$. □

The property (iii) is similar in form to the following characterisation of NP. A language L is in NP if and only if there exists a p-bounded $t\colon \mathbb{N} \to \mathbb{N}$ and a language M in P such that

$$\forall\, n \in \mathbb{N}\ \forall\, x \in \Sigma^n\ (x \in L \Leftrightarrow \exists\, e \in \Sigma^{t(n)}\ x@\, e \in M).$$

Here, Σ is the alphabet, and @ a new symbol. If we rewrite the condition with characteristic functions as:

$$\chi_L(x) = \bigvee_{e \in \Sigma^{t(n)}} \chi_M(x@\, e),$$

then property (iii) looks very similar, with the disjunction over e replaced by the sum over e. This similarity is supported by the following formal connection.

PROPOSITION 4.4. *Let $f = (f_n)_{n \in \mathbb{N}}$ with $f_n \in F[x_1, \ldots, x_n]$ be a family of polynomials, of degree at most 1 in each variable, and with all monomials having coefficient either zero or one. Suppose that there exists a deterministic polynomial-time bounded Turing machine which on input $e \in \{0,1\}^n$ decides whether the coefficient of $\mathbf{x}^e$ in f_n is zero or one. Then f is p-definable.* □

PROPOSITION 4.5. *The permanent family is p-definable.*

PROOF. Consider indeterminates x_{ij}, $1 \leqslant i, j \leqslant n$, over F, and

$$g = \Big(\prod_{\substack{1 \leqslant i,j,k,l \leqslant n \\ (i,j) \neq (k,l) \\ i=k \text{ or } j=l}} (1 - x_{ij}x_{kl})\Big) \cdot \Big(\prod_{1 \leqslant i \leqslant n} \sum_{1 \leqslant j \leqslant n} x_{ij}\Big).$$

We want to check that the polynomial h defined by g equals $\operatorname{per}((x_{ij}))$. On substituting $e \in \{0,1\}^{n \times n}$, the first factor of g vanishes if and only if in some row or some column of e, more than a single 1 occurs. If the first factor does not vanish, then the second is zero if and only if some row has all entries equal to 0. Together we get

$$g(e) \neq 0 \Leftrightarrow \text{every row and every column of } e \text{ has exactly one 1.}$$

(In characteristic zero, we only need the factors $1 - x_{ij}x_{kj}$ with $i \neq k$ in the first factor.) □

For the main results of this tutorial, we will use the following combinatorial interpretation of linear algebra. We consider complete weighted directed graphs G on a set N of nodes, with a edge-weight function

$$G\colon N \times N \to X = F \cup \{x_1, \ldots, x_m\}.$$

Obviously, the set of all such graphs on $N = \{1, \ldots, n\}$ is in a natural bijective correspondence with the set $X^{n \times n}$ of $n \times n$-matrices, via

$$G(i,j) = A_{ij}.$$

Under this correspondence, a permutation from $\operatorname{Sym}(N)$ corresponds to a cycle cover of G, and

$$\det A = \sum_{\substack{c \text{ a cycle} \\ \text{cover of } G}} \operatorname{sign}(c) \cdot (\text{product of weights on } c),$$

$$\operatorname{per} A = \sum_{\substack{c \text{ a cycle} \\ \text{cover of } G}} (\text{product of weights on } c).$$

In pictures of G, we do not draw those edges that have weight 0. Figure 7 shows three of the graphs (=matrices $\mu(\varphi)$) constructed in the proof of Theorem 3.1, corresponding to the examples in section 3. All edges shown have weight 1, unless otherwise specified.
We will use the following conventions. If G is a weighted directed graph with node set N, then $\operatorname{Sym} G$ is the group of permutations of N, and for $\sigma \in \operatorname{Sym} G$, we have the diagonal product

$$p_\sigma(G) = \prod_{i \in N} G(i, \sigma(i)).$$

If under the association discussed above G corresponds to A, then we also use $\operatorname{Sym} A$ and $p_\sigma(A)$ as above. Thus

$$\det A = \sum_{\sigma \in \operatorname{Sym} G} \operatorname{sign}(\sigma) p_\sigma(G),$$

$$\operatorname{per} A = \sum_{\sigma \in \operatorname{Sym} G} p_\sigma(G).$$

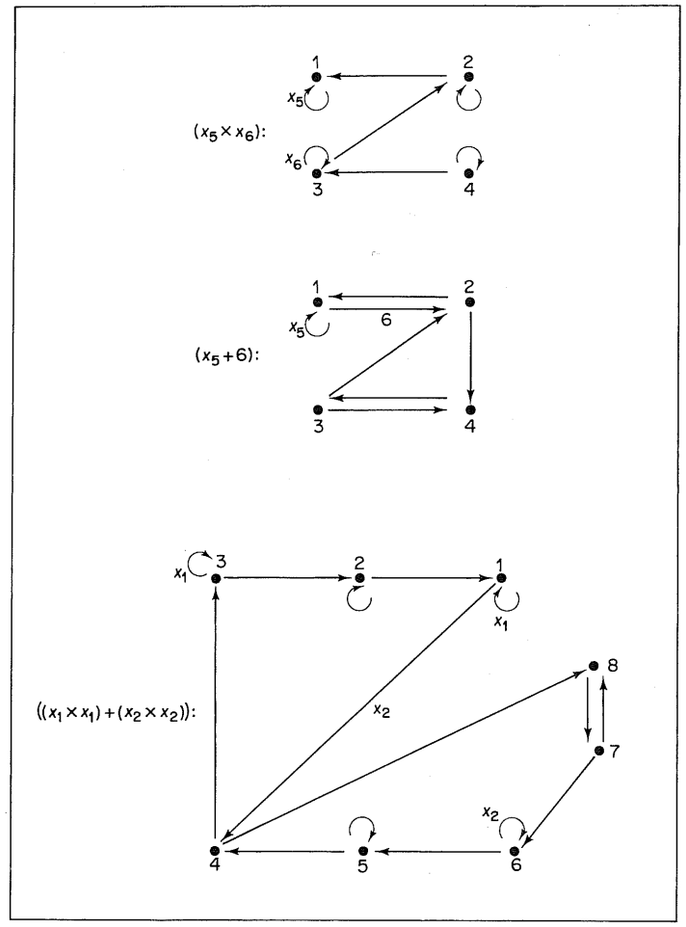


Fig. 7.

The combinatorial interpretation of matrices leads to considering the following polynomials. If S is a set of sets of edges of the complete graph, where the weight of edge (i, j) is an indeterminate x_{ij}, then

$$\sum_{E \in S} \prod_{(i,j) \in E} x_{ij}$$

is the polynomial for S. Examples:

1. Permanent: $S = \{\text{cycle covers}\}$.
2. Hamiltonian circuits: $S = \{\text{self-avoiding cycles covering each node}\}$.
3. Hamiltonian paths: $S = \{\text{self-avoiding paths from node 1 to node 2, covering each node}\}$.
4. Spanning trees: $S = \{\text{spanning trees in which each edge is directed away from node 1}\}$.
5. Reliability: $S = \{\text{edge sets in which there is a path from node 1 to node 2}\}$.

Varying the number n of nodes of the complete graph in these examples, we obtain families of polynomials over F.

PROPOSITION 4.6. *All the above families are p-definable.* □

Most "naturally occurring" families of polynomials seem to be p-definable. Some of them—e.g. the determinant—are p-computable, while others—e.g. the permanent—have defied attempts to find polynomial-time algorithms for them.

We recall the central conjecture that in some cases no such fast algorithms exist:

VALIANT'S HYPOTHESIS. *Over any field F, there exist p-definable families of polynomials that are not p-computable.* □

Valiant's hypothesis is the arithmetic analogue of Cook's hypothesis "$P \neq NP$", which refers to Boolean computations. If Valiant's hypothesis were false, then PER would be p-computable, and indeed PER_n could be computed by a program of depth $O(\log^2 n)$ and length $n^{O(1)}$. The general simulations of arithmetic computations over $\mathbb{Q}$ by Boolean computations (Ibarra & Moran, 1983; von zur Gathen, 1985, Corollary 6.9) then yield Boolean circuits of polynomial size for the integer permanent. Since the integer permanent is complete for $\#P$ (Valiant, 1979*b*), we have for the non-uniform versions of the Boolean complexity classes

$$P(\textit{NON-UNIFORM}) \neq NP(\textit{NON-UNIFORM}) \Rightarrow$$
$$NC^2(\textit{NON-UNIFORM}) \neq \#P(\textit{NON-UNIFORM}) \Rightarrow$$
$$\text{Valiant's hypothesis over } F,$$

when F is $\mathbb{Q}$ or a finite field. (In characteristic two, we use the p-complete family of Hamiltonian cycle polynomials instead of PER in this remark.) For a formal relation with the "uniform" Boolean conjecture, we would have to consider "uniform families $(Q_n)_{n \in \mathbb{N}}$ of straight-line programs", where a Turing machine, say probabilistic and polynomial-time bounded, can produce a description of Q_n on input n in unary (see von zur Gathen, 1986). The Boolean class R consists of the languages accepted by such Turing machines. Then a slightly stronger form of $P \neq NP$, namely $R \neq NP$ (or $R \neq \#P$) implies Valiant's hypothesis over $\mathbb{Q}$, since a uniform family of straight-line programs for the permanent would yield a random polynomial-time bounded Turing machine for the integer permanent. No converse implication is clear, and indeed it is hoped that Valiant's arithmetic hypothesis is "easier" to prove than Cook's Boolean hypothesis.

DEFINITION 4.7. A family f is *p-complete* if and only if

(i) f is p-definable, and
(ii) every p-definable family is a p-projection of f. □

PROPOSITION 4.8. *Let f be a p-complete family over F. Then Valiant's hypothesis holds over F if and only if f is not p-computable.* □

An interesting question is whether the classes of p-computable and of p-definable families are closed under some natural mathematical operations, such as:

1. gcd,
2. factorisation,
3. substitution,
4. taking coefficients,

5. derivative,
6. integral,
7. definition.

Proposition 2.14 has noted that both classes are closed under p-projections. In the sparse representation, von zur Gathen & Kaltofen (1985) showed that irreducible factors may grow more than polynomially in size. Kaltofen (1986) surprised us by showing that this does not happen for straight-line programs: the p-computable polynomials are closed under gcd's and factorisation. The factorisation algorithm works over those fields over which univariate polynomials can be factored efficiently; these include finite fields and the field of rational numbers. (An unsolved technical problem still is to extract pth roots of polynomials over a finite field of characteristic p.) Kaltofen's results are "uniform" in the following sense: There is a probabilistic Turing machine, which takes as input a binary representation of two programs, a list of constants used, and a supply of random elements from a large enough finite subset of the field. It outputs a binary representation of a program computing the gcd of the two polynomials computed by the two input programs, using the given constants. The running time is polynomial in the input size (which, over $\mathbb{Q}$, has to be defined carefully) plus the degrees of the two input polynomials, both for the new program and for the Turing machine. Similarly, Kaltofen can produce the irreducible factors of a polynomial, and also the reduced numerator and denominator if the input program computes a rational function.

These results provide justification for the condition of "p-bounded degree" for p-computable (or p-definable) polynomials. If no such degree bound is imposed, then even deciding whether the gcd of two univariate polynomials over $\mathbb{Q}$ is non-trivial is NP-hard (Plaisted, 1984).

For the proper notion of *substitution*, one has to consider arrays $g = (g_{nm})_{m \leqslant n \in \mathbb{N}}$, which can be substituted into a family of polynomials in the obvious way. Both complexity classes are closed under this type of substitution.

The *coefficient* in $f \in F[x_1, \ldots, x_n]$ of the monomial $\mathbf{x}^e = x_1^{e_1} \cdots x_n^{e_n}$ (with $e \in \mathbb{N}^n$) is the unique polynomial $g \in F[x_1, \ldots, x_n]$ for which there exists a polynomial

$$h = \sum_{a \in \mathbb{N}^n} h_a x^a$$

such that

$$f = g \cdot \mathbf{x}^e + h,$$
$$\forall\, i \leqslant n\ (e_i > 0 \Rightarrow x_i \text{ does not occur in } g),$$
$$\forall\, a \in \mathbb{N}^n\ (h_a \neq 0 \Rightarrow \exists\, i\ (e_i \neq 0 \text{ and } a_i \neq e_i)).$$

EXAMPLE 4.9.

1. The coefficient of $1 = x_1^0 \cdots x_n^0$ in f is f.
2. The coefficient of x_{12} in DET_2 is $-x_{21}$.
3. The coefficient of x_1 in $\mathrm{POWERSUM}_3$ is 0.
4. Consider the family $q = (q_n)_{n \in \mathbb{N}}$ with

$$q_n = \prod_{1 \leqslant i \leqslant n} \sum_{1 \leqslant j \leqslant n} x_{ij} y_j.$$

The coefficient of $y_1 \cdots y_n$ in q_n is PER_n. (This is already in Hammond, 1879.) This family q is p-expressible, but—under Valiant's hypothesis—the coefficient is not p-computable.

In contrast, the class of p-definable families is closed under the operation of taking coefficients.

The p-computable functions are closed under *partial derivatives* $\partial f/\partial x_i$ and *integrals* $\int f \mathrm{d}x_i$; see Baur & Strassen (1982) for a surprisingly strong result about derivatives. However, it makes sense now to consider p-bounded application of these operators, and then p-computability may get lost:

$$\frac{\partial}{\partial y_1} \frac{\partial}{\partial y_2} \cdots \frac{\partial}{\partial y_n} q_n = \mathrm{PER}_n,$$
$$\left(\frac{3}{2}\right)^n \int_{-1}^{1} \cdots \int_{-1}^{1} (y_1 \cdots y_n q_n)\, \mathrm{d}y_1 \cdots \mathrm{d}y_n = \mathrm{PER}_n.$$

However, the class of p-definable families is closed under these operations.

Finally, consider the operation "g defines h" of Definition 4.1 (i). Then Valiant's hypothesis is equivalent to the statement that the p-computable families are not closed under this operation. It turns out that the p-definable families are closed under this "definition". An arithmetic analogue $P_0 \subseteq P_1 \subseteq \cdots$ of the Boolean hierarchy of Meyer & Stockmeyer (1972) would let P_0 consist of the p-computable polynomials, and P_i would be composed of those polynomials that can be defined by polynomials in P_{i-1}. Then the above shows that this hierarchy collapses: $P_1 = P_2 = \cdots$.

The upshot is that among our two natural complexity classes, the p-computable families are somewhat vulnerable, but the p-definable ones are robust under the mathematical operations considered above.

5. *P*-complete Families

In this section we prove Valiant's central result that the permanent and Hamiltonian circuit families are p-complete. We start with an auxiliary structure, called "coupled permanents" (and proposed by Volker Strassen). Afterwards, we only have to show how to express these coupled permanents by ordinary permanents respectively by Hamiltonian cycle polynomials.

Let G be a (complete directed edge-weighted) graph on node set N with weights from a commutative ring R (i.e. a matrix from $R^{N \times N}$). A set

$$P \subseteq \binom{N \times N}{2}$$

is called a set of *couples of edges* of G if for all $a = \{(u, v), (u', v')\}$, $b \in P$ we have $u \neq u'$, $v \neq v'$, and $(a \neq b \Leftrightarrow a \cap b = \emptyset)$. An edge of G corresponds to a position in the matrix, and a couple is an unordered pair. The two conditions say that the two edges of a couple originate and terminate in different nodes, and that any edge may occur in at most one couple of a set of couples. For an edge $e = (u, v) \in N \times N$ and $\sigma \in \mathrm{Sym}\ G$ we write "$e \in \sigma$" if and only if $\sigma(u) = v$; this is consistent with the interpretation of σ as a subset of $N \times N$. The set $\mathrm{Sym}\ (G; P)$ of *coupled permutations* is

$$\mathrm{Sym}\ (G; P) = \{\sigma \in \mathrm{Sym}\ G\colon \forall\, \{e, e'\} \in P\ (e \in \sigma \Leftrightarrow e' \in \sigma)\},$$

and the *coupled permanent* $\mathrm{per}\ (G; P)$ is

$$\mathrm{per}\ (G; P) = \sum_{\sigma \in \mathrm{Sym}\,(G;P)} p_\sigma(G).$$

Thus $\mathrm{per}\ G = \mathrm{per}\ (G; \emptyset)$.

LEMMA 5.1. *Suppose that $g \in F[x_1, \ldots, x_n]$ has expression size e, and that g defines h. Then there exists a matrix B and a set Q of couples of edges of B such that*

(2.1) $\operatorname{per}(B; Q) = h$,
(2.2) $B \in (F \cup \{x_1, \ldots, x_n\})^{u \times u}$, *where* $u \leqslant 4e + 2n + 4$,
(2.3) $\#Q \leqslant e + 1$.

PROOF. We will use the following construction. Let R be a polynomial ring over F (or a commutative F-algebra), x an indeterminate over R, $A \in (R \cup \{x\})^{s \times s}$ a matrix in which x occurs exactly m times, and Q a set of couples of edges of A. We construct a matrix $C = \gamma(A, x)$, containing A as a full subgraph, and a set $P = \pi(A, x)$ of couples of edges of C such that

(3.1) $\operatorname{per}(C; Q \cup P) = (\operatorname{per}(A; Q))(0) + x(\operatorname{per}(A; Q))(1)$,
(3.2) $C \in (R \cup \{x\})^{t \times t}$, where $t = s + 2m + 2$,
(3.3) any element of $R \setminus \{0, 1\}$ occurs an equal number of times in A and C, x occurs exactly once in C, and $\#P = m$.

("C contains A as a full subgraph" means that we identify each node of A with a node of C. Although the edge weights in A and C of corresponding edges need not be equal, it is clear how to interpret Q as a set of couples of edges of C. Recall that all our graphs are complete.)

Then we prove the lemma by an easy induction: We let φ be an expression for g of size e, $Q_0 = \emptyset$, and $C_0 = \mu(\varphi) \in X^{2e+2}$ the matrix from Proposition 3.4, where $X = F \cup \{x_1, \ldots, x_n\}$, and $\operatorname{per} C_0 = g$. Let m_i denote the number of occurrences of x_i in C_0. We get a matrix C_i and a set Q_i for $1 \leqslant i \leqslant n$ by letting $R_i = F[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$, $C_i = \gamma(C_{i-1}, x_i)$ and $Q_i = Q_{i-1} \cup \pi(C_{i-1}, x_i)$.

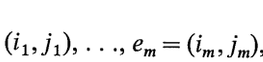
Properties (3.1), (3.2), (3.3) translate inductively into:

(4.1) $\operatorname{per}(C_i; Q_i) = \sum_{e \in \{0,1\}^i} g(e_1, \ldots, e_i, x_{i+1}, \ldots, x_n) x_1^{e_1} \cdots x_i^{e_i}$,
(4.2) $C_i \in X^{t_i \times t_i}$, where $t_i = 2e + 2 + 2(m_1 + \ldots + m_i) + 2i$,
(4.3) for $1 \leqslant i, j \leqslant n$, the number of occurrences of x_j in C_i is m_j if $i < j$, and 1 otherwise.

Furthermore, $\#Q_i = m_1 + \ldots + m_i$.

It is easy to see that a formula of size e can have at most $e+1$ occurrences of variables, so that $m_1 + \ldots + m_n \leqslant e + 1$. Then with $B = C_n$ and $Q = Q_n$ the claims of Lemma 5.1 hold. The construction of C and P proceeds as follows. C consists of A plus (disjointly) the graph C_1 on nodes $\{u_0, \ldots, u_{m+1}, v_1, \ldots, v_m\}$ (Fig. 8).

All edges drawn have weight 1, except that $C_1(u_0, u_1) = x$; besides these edges, every node u_k or v_k has a self-loop: $C_1(u_k, u_k), C_1(v_k, v_k) = 1$. Each edge weight of A is repeated in C, except that each occurrence of x is replaced by 1. For $m = 0$, C_1 is:



We number the occurrences of x in A as $e_1 = (i_1, j_1), \ldots, e_m = (i_m, j_m)$, and let $d_k = (u_k, u_{k+1})$ for $1 \leqslant k \leqslant m$. Then

$$P = \{\{d_k, e_k\} : 1 \leqslant k \leqslant m\}$$

is a set of couples of edges for C. Properties (3.2) and (3.3) obviously hold, and it remains to prove (3.1). This is clear if $m = 0$, and we now assume $m \geqslant 1$.

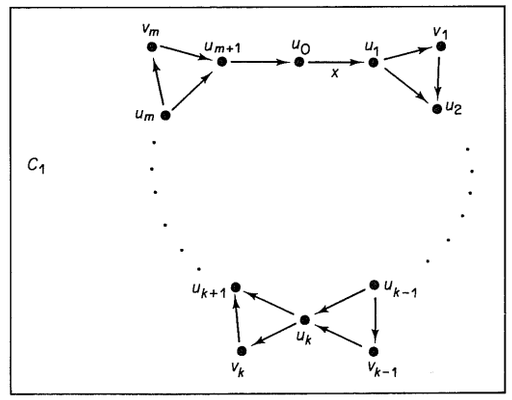
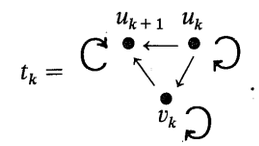


Fig. 8.

Figure 9 shows the construction for the example $x_5 + 6$ from section 3. Returning to the proof of (3.1), we consider $\sigma \in \operatorname{Sym} C$ and one triangle



We say that σ *moves in* t_k if and only if $\sigma(u_k) \neq u_k$. We then have for all $\sigma \in \operatorname{Sym} C$ with

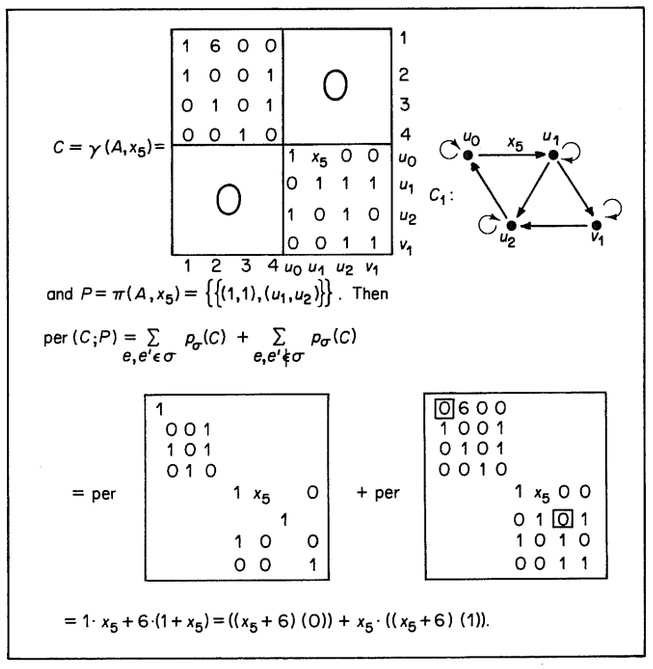


Fig. 9.

$p_\sigma(C) \neq 0$ that

$$\exists\, k \leqslant m\ \sigma \text{ moves in } t_k$$
$$\Leftrightarrow (\forall\, k \leqslant m\ \sigma \text{ moves in } t_k) \text{ and } \sigma(u_0) = u_1$$

(since $m \geqslant 1$). This follows by inductively applying the "domino theorem": a move (="revolution") in one triangle forces a move in each neighbouring triangle.

Consider

$$S = \{\sigma \in \mathrm{Sym}(C; Q \cup P) \colon p_\sigma(C) \neq 0\},$$

the mapping

$$\alpha \colon S \to \mathrm{Sym}(A; Q)$$
$$\sigma \mapsto \sigma \restriction (\text{nodes of } A),$$
$$T = \alpha(S),$$

and for $M \subseteq \{1, \ldots, m\}$

$$S_M = \{\sigma \in S \colon \forall\, k \leqslant m\ (\sigma(u_k) = u_{k+1} \Leftrightarrow k \in M)\},$$
$$T_M = \alpha(S_M).$$

Note that $(S_M)_{M \subseteq \{1,\ldots,m\}}$ is a partition of S.

Then the following hold for all $\tau \in \mathrm{Sym}(A; Q)$:

$$\tau \in T \Leftrightarrow p_\tau(A) \neq 0,$$
$$\tau \in T_M \Leftrightarrow p_\tau(A) \neq 0 \quad \text{and} \quad \forall\, k \leqslant m\ (\sigma(i_k) = j_k \Leftrightarrow k \in M).$$

Consider $M \neq \emptyset$ and $\tau \in T_M$. Then

$$\#\alpha^{-1}(\tau) = 1,$$
$$x \cdot p_\tau(A(1)) = p_\sigma(C) \text{ if } \alpha(\sigma) = \tau.$$

Here we use that σ moves in t_k if $k \in M$ and $\sigma \in S_M$. As an example, if $m = 3$, $M = \{2, 3\}$, and $\alpha(\sigma) \in T_M$, then σ operates on C_1 as in Fig. 10.

Now consider $\tau \in T_\emptyset$. Then $\alpha^{-1}(\tau) = \{\sigma_0, \sigma_1\}$, where σ_0 uses all self-loops on C_1, and σ_1 on C_1 is as in Fig. 11.

We have

$$p_\tau(A) = p_\tau(A(1)) = p_\tau(A(0)) = p_{\sigma_0}(C),$$
$$x \cdot p_\tau(A(1)) = p_{\sigma_1}(C).$$

We also have that

$$\sum_{\tau \in T_\emptyset} p_\tau(A(0)) = (\mathrm{per}\,(A; Q))(0),$$
$$\sum_{\substack{M \subseteq \{1,\ldots,m\} \\ \tau \in T_M}} p_\tau(A(1)) = (\mathrm{per}\,(A; Q))(1).$$

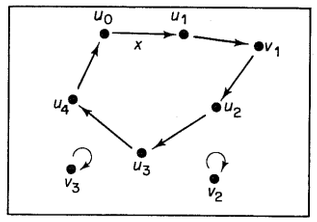


Fig. 10.

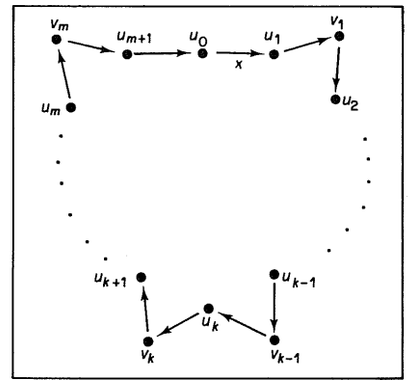


Fig. 11.

Assembling all this, we get

$$\begin{aligned}\mathrm{per}\,(C; Q \cup P) &= \sum_{\sigma \in \mathrm{Sym}(C; Q \cup P)} p_\sigma(C) = \sum_{M \subseteq \{0,\ldots,m\}} \sum_{\sigma \in S_M} p_\sigma(C) \\ &= \sum_{M \neq \emptyset} \sum_{\tau \in T_M} x \cdot p_\tau(A(1)) + \sum_{\tau \in T_\emptyset} [p_\tau(A(0)) + x \cdot p_\tau(A(1))] \\ &= (\mathrm{per}\,(A; Q))(0) + x \cdot (\mathrm{per}\,(A; Q))(1).\end{aligned}$$

Thus properties (3.1), (3.2), (3.3) are proven. □

REMARK 5.2. We can make the coupled permanents into a family of polynomials. For $n \in \mathbb{N}$, we have indeterminates x_{ij} and $y_{\{e,e\prime\}}$ for $i, j \in \mathbb{N}$ and

$$\{e, e'\} \in \binom{N \times N}{2},$$

where $N = \{1, \ldots, n\}$. Then

$$c_n = \sum_{\sigma \in \mathrm{Sym}\, N} \prod_{i \in N} x_{i,\sigma(i)} \prod_{\substack{i,j,k \in N \\ \sigma(j) \neq k}} y_{\{(i,\sigma(i)),(j,k)\}}$$

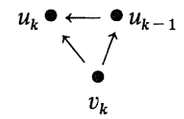
is the nth "coupled permanent polynomial". If $A \in F^{n \times n}$ is a matrix and

$$Q \subseteq \binom{N \times N}{2}$$

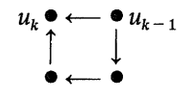
a set of couples of edges of A, then per $(A; Q)$ equals c_n evaluated at $x_{ij} = A_{ij}$ and

$$y_p = \begin{cases} 0 & \text{if } p \in Q, \\ 1 & \text{otherwise.} \end{cases}$$ □

REMARK 5.3 (C. Greither). Let us replace in the construction of C each triangle



by a square



and, if m is even, add an extra node

$$\bullet_{u_{m+1}} \longrightarrow \bullet_{u_{m+2}} \longrightarrow \bullet_{u_0}$$

between u_{m+1} and u_0. Then each permutation with non-zero contribution to the permanent is even. Therefore the permanent equals the determinant, and it follows that an analogue of Lemma 5.1 also holds for "coupled determinants". □

THEOREM 5.4. *Let F be a field of characteristic different from two. Then* PER *is p-complete over F.*

PROOF. The main step is the following construction which removes one couple from a coupled permanent. Let $G \in R^{t\times t}$ be a matrix over a commutative F-algebra R, Q a set of couples of edges of G, and $\{e, e'\}$ a couple of edges of G with $\{e, e'\} \notin Q$. We will get a graph $H = \eta(G, \{e, e'\})$, which contains G as a full subgraph, such that

(5.1) $\operatorname{per}(H; Q) = \operatorname{per}(G; Q \cup \{\{e, e'\}\})$,
(5.2) $H \in R^{r\times r}$, where $r = t+3$.

Given this construction, we prove the theorem as follows. PER is p-definable by Proposition 4.5. Let f be an arbitrary p-definable family of polynomials over F, and g, h as in Definition 4.1. It is sufficient to prove that h is a p-projection of PER, since then by Proposition 2.14 (i) also f is such a p-projection, and therefore PER p-complete.

Let e_n be the expression size of g_n. By assumption, e_n is a p-bounded function of n.

Fix some $n \in \mathbb{N}$. By Lemma 5.1, there exist $B_0 \in X^{u\times u}$ and a set Q_0 of couples of edges of B_0, where $X = F \cup \{x_1, \ldots, x_n\}$, such that $h_n = \operatorname{per}(B_0; Q_0)$ and $u \leqslant 4e_n + 2n + 4$. Let $q = \#Q_0$ and $Q_0 = \{c_1, \ldots, c_q\}$. Then $q \leqslant e_n + 1$. For $1 \leqslant i \leqslant q$ we get graphs $B_i = \eta(B_{i-1}, c_i)$ and sets $Q_i = \{c_{i+1}, \ldots, c_q\}$ such that

(6.1) $\operatorname{per}(B_i; Q_i) = h$,
(6.2) $B_i \in X^{(u+3i)\times(u+3i)}$.

In particular, $Q_q = \emptyset$, $B_q \in X^{s\times s}$ with $s \leqslant 7e_n + 2n + 7$, and $h_n = \operatorname{per}(B_q)$. Thus h is a p-projection of PER.

The basic graph for the construction with properties (5.1) and (5.2) is the following "coupler" K on 3 nodes a, b, c (Fig. 12). The edges drawn have weight 1, unless otherwise specified. If $K(U \mid V)$ denotes K with rows U and columns V removed, then the following hold:

(7.1) $\operatorname{per} K = 1$,
(7.1) $\operatorname{per} K(b, c \mid a, c) = 1$,

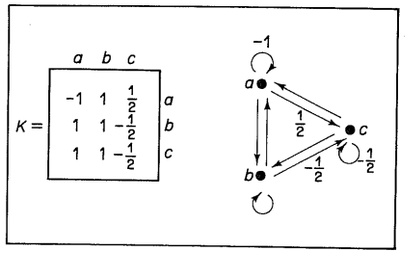


Fig. 12.

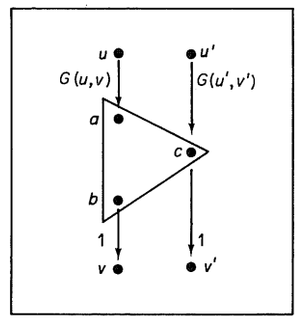


Fig. 13.

(7.3) $\operatorname{per} K(b \mid a) = 0$,
(7.4) $\operatorname{per} K(b \mid c) = 0$,
(7.5) $\operatorname{per} K(c \mid a) = 0$,
(7.6) $\operatorname{per} K(c \mid c) = 0$.

Now consider G, Q, e, e' as above, and write $e = (u, v)$ and $e' = (u', v')$, with $u \neq u'$ and $v \neq v'$. The graph (or matrix) $H = \eta(G, \{e, e'\})$ lives on the nodes of G plus the three nodes of K. All edges of G and K occur with the same weight in H, except that $H(u, v) = H(u', v') = 0$. The interconnecting edges are as in Fig. 13.

Properties (5.2) and (5.3) obviously hold. We now let

$$S = \{\sigma \in \operatorname{Sym}(H; Q) : p_\sigma(H) \neq 0\},$$

and partition S into six parts (Fig. 14).

Figure 14 illustrates exactly which connections between u, v, u', v' and a, b, c are required for each $\sigma \in S_k$. We first note that any $\sigma \in S$ uses a matching number of in/out edges between G and K. Inspection shows that this implies

$$\sigma \in \bigcup_{1\leqslant k\leqslant 6} S_k,$$

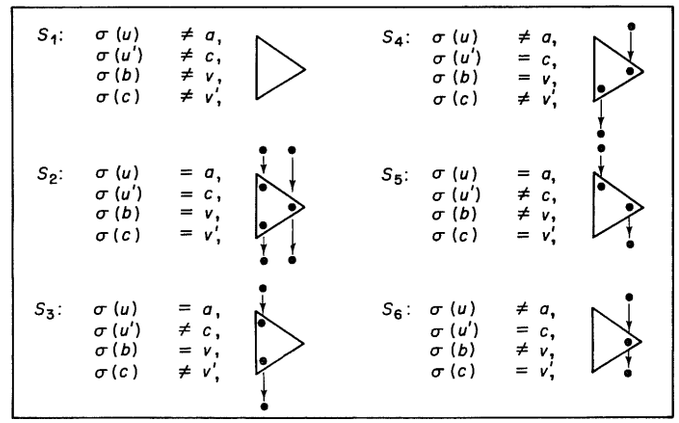


Fig. 14.

and therefore

$$S = \bigcup_{1 \leqslant k \leqslant 6} S_k$$

is a disjoint union. We now have six claims.

(8.1) $$\sum_{\sigma \in S_1} p_\sigma(H) = \sum_{\substack{\tau \in \mathrm{Sym}(G;Q) \\ e, e' \notin \tau}} p_\tau(G),$$

(8.2) $$\sum_{\sigma \in S_2} p_\sigma(H) = \sum_{\substack{\tau \in \mathrm{Sym}(G;Q) \\ e, e' \in \tau}} p_\tau(G),$$

(8.k) $$\sum_{\sigma \in S_k} p_\sigma(H) = 0 \text{ for } 3 \leqslant k \leqslant 6.$$

Since $S_j \cap S_k = \emptyset$ for $j \neq k$, these claims will imply

$$\begin{aligned} \mathrm{per}\,(H;Q) &= \sum_{\sigma \in S} p_\sigma(H) \\ &= \sum_{\sigma \in S_1} p_\sigma(H) + \sum_{\sigma \in S_2} p_\sigma(H) \\ &\quad \sum_{\substack{\tau \in \mathrm{Sym}(G;Q) \\ e, e' \notin \tau}} p_\tau(G) + \sum_{\substack{\tau \in \mathrm{Sym}(G;Q) \\ e, e' \in \tau}} p_\tau(G) \\ &= \mathrm{per}\,(G; Q \cup \{\{e, e'\}\}), \end{aligned}$$

which is (5.1).

It is convenient to consider the following equivalence relation on S:

$$\begin{aligned} \rho \equiv \sigma &\Leftrightarrow \rho \text{ and } \sigma \text{ agree outside the coupler} \\ &\Leftrightarrow (\rho(i) \neq \sigma(i) \Rightarrow i, \rho(i), \sigma(i) \in \{a, b, c\}). \end{aligned}$$

For (8.1), we let $Q' = Q \cup \{\{e, e'\}\}$ and consider

$$\begin{aligned} &\alpha_1 : S_1 \to \mathrm{Sym}\,(G; Q) \\ &\alpha_1(\sigma) = \sigma \restriction (\text{nodes of } G). \end{aligned}$$

One checks that α_1 is well defined, has

$$T_1 = \{\tau \in \mathrm{Sym}\,(G; Q') : p_\tau(G) \neq 0 \text{ and } e, e' \notin \tau\}$$

as image, and the fibres are equivalence classes. For $\tau \in T_1$, we have

$$\begin{aligned} \sum_{\alpha_1(\sigma) = \tau} p_\sigma(H) &= \sum_{\alpha_1(\sigma) = \tau} p_{\sigma \restriction K}(K) \cdot p_\tau(G) \restriction \\ &= \Big[\sum_{\rho \in \mathrm{Sym}\,K} p_\rho(K)\Big] \cdot p_\tau(G) \\ &= \mathrm{per}\,K \cdot p_\tau(G) = p_\tau(G), \end{aligned}$$

by (7.1). Summing over $\tau \in T_1$ yields (8.1).

The remaining claims are proven in an analogous way. One considers the corresponding mappings:

$$\begin{aligned} &\alpha_2 : S_2 \to \mathrm{Sym}\,(G; Q) \\ &\alpha_2(\sigma)(i) = \begin{cases} v & \text{if } i = u, \\ v' & \text{if } i = u', \\ \sigma(i) & \text{otherwise,} \end{cases} \end{aligned}$$

$$\begin{aligned} &\alpha_3 : S_3 \to \mathrm{Sym}\,(G; Q) \\ &\alpha_3(\sigma)(i) = \begin{cases} v & \text{if } i = u, \\ \sigma(i) & \text{otherwise,} \end{cases} \\ &\alpha_4 : S_4 \to \mathrm{Sym}\,(G; Q) \\ &\alpha_4(\sigma)(i) = \begin{cases} v & \text{if } i = u', \\ \sigma(i) & \text{otherwise,} \end{cases} \\ &\alpha_5 : S_5 \to \mathrm{Sym}\,(G; Q) \\ &\alpha_5(\sigma)(i) = \begin{cases} v' & \text{if } i = u, \\ \sigma(i) & \text{otherwise,} \end{cases} \\ &\alpha_6 : S_6 \to \mathrm{Sym}\,(G; Q) \\ &\alpha_6(\sigma)(i) = \begin{cases} v' & \text{if } i = u', \\ \sigma(i) & \text{otherwise.} \end{cases} \end{aligned}$$

One checks that each α_k is well defined, and the fibres are equivalence classes. For $\tau \in \mathrm{Sym}\,(G; Q)$,

(9.k) $$\sum_{\alpha_k(\sigma) = \tau} p_\tau(H) = \lambda_k \cdot p_\tau(G),$$

where

$$\lambda_k = \begin{cases} 1 & \text{if } k \leqslant 2, \\ 0 & \text{otherwise,} \end{cases}$$

using equation (7.k). Summing (9.k) over $\tau \in \mathrm{Im}\,\alpha_k$, the claim (8.$k$) follows. □

The question of what kind of relations exist between permanent and determinant, in particular whether the permanent can be expressed as the determinant of a matrix, is a classical mathematical problem. Szegö (1913), answering a question posed by Pólya (1913), showed that for $n \geqslant 3$, there is no way of generalising

$$\mathrm{per} \begin{vmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{vmatrix} = \det \begin{vmatrix} x_{11} & -x_{12} \\ x_{21} & x_{22} \end{vmatrix},$$

i.e. of affixing $\pm$ signs to the indeterminate entries x_{ij} such that

$$\mathrm{per}\,(x_{ij}) = \det\,(\pm x_{ij}).$$

Marcus & Minc (1961) proved that one cannot relate certain permanental and determinantal functions by linear mappings. In particular, for $n \geqslant 3$, there are no linear forms f_{kl} in indeterminates x_{ij} $(1 \leqslant i, j, k, l \leqslant n)$ such that $\mathrm{per}\,(x_{ij}) = \det\,(f_{kl})$. Von zur Gathen (1987) proves that if the $n \times n$—permanent is a projection of the $m \times m$—determinant, then $m \geqslant 1{\cdot}06n$; Meshulam (1987) and Seress & Babai (1987) improve this to $m \geqslant \sqrt{2}n$. The previous results establish the following connection between this classical algebraic question and the "qp-variant" of Valiant's hypothesis (the "extended Valiant hypothesis").

COROLLARY 5.5. *Let F be a field of characteristic different from two. Then the following are equivalent:*

(*i*) *there exists a qp-definable family which is not qp-computable,*
(*ii*) PER *is not qp-computable,*
(*iii*) PER *is not a qp-projection of* DET.

PROOF. For "$(i)\Rightarrow(ii)$", one has to check that the proofs of this section go through with "qp" for "p". Since DET is qp-computable (even p-computable), "$(ii)\Rightarrow(iii)$" holds by Proposition 2.14 (ii). "$(iii)\Rightarrow(ii)$" follows from Theorem 3.1. "$(ii)\Rightarrow(i)$" is trivial. □

Our next goal is to prove that the family of Hamiltonian cycle polynomials is p-complete. We first show that "coupled Hamiltonian cycles" can simulate coupled permanents, and then that ordinary Hamiltonian cycles can simulate the coupled ones. So let G be an arbitrary (complete directed edge-weighted) graph on node set N, and

$$P \subseteq \binom{N \times N}{2}$$

a set of couples of edges of G. We have the set of *Hamiltonian cycles*

$$HC(G) = \{\sigma \in \mathrm{Sym}\ N : \sigma \text{ consists of one cycle}\},$$

the *coupled Hamiltonian cycles*

$$HC(G; P) = HC(G) \cap \mathrm{Sym}\ (G; P),$$

and the *coupled Hamiltonian* of G

$$\mathrm{hc}(G; P) = \sum_{\sigma \in HC(G;P)} p_\sigma(G).$$

The ordinary *Hamiltonian* of G is

$$\mathrm{hc}(G) = \mathrm{hc}(G; \emptyset).$$

If G_n is the graph on nodes $1, \ldots, n$ with an indeterminate x_{ij} as weight on the edge (i, j), then $HC_n = \mathrm{hc}(G_n)$ is the nth member in the Hamiltonian circuit family HC as defined in section 4.

THEOREM 5.6. *Over any field, HC is p-complete.*

PROOF. We start by showing that coupled Hamiltonian circuits can simulate coupled permanents, using the following construction, which was suggested by A. Möbus.

Given a graph $G \in R^{s\times s}$, where R is a commutative ring, and

$$Q \subseteq \binom{s \times s}{2},$$

we construct a graph $H \in R^{t\times t}$ and

$$P \subseteq \binom{t \times t}{2}$$

such that

(10.1) $\mathrm{hc}(H; P) = \mathrm{per}\ (G; Q)$,
(10.2) $t = 4s^2 + 2s$, and $\#P = \#Q$,
(10.3) every element of $R\backslash\{0, 1\}$ occurs an equal number of times in G and H.

We arrange the positions (i, j) in G $(1 \leqslant i, j \leqslant s)$ arbitrarily in a circle, say alphabetically with $(1, 1)$ after (s, s). For every i, $1 \leqslant i \leqslant s$, we have two nodes r_i and c_i in H (for "row i" and "column i"), and for every i, j, $1 \leqslant i, j \leqslant s$, four nodes v_{ij}, w_{ij1}, w_{ij2}, w_{ij3}, with edge-

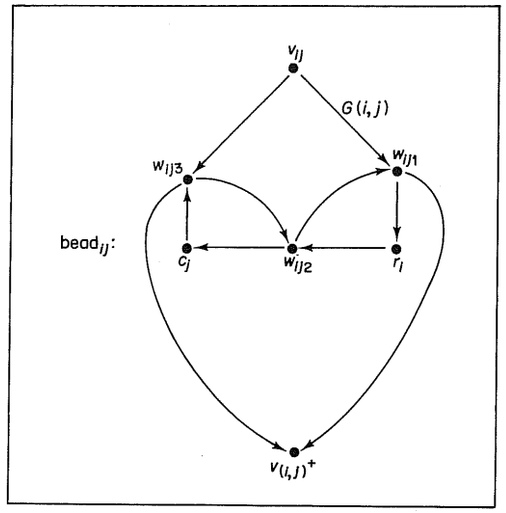


Fig. 15.

weights as in Fig. 15, where $(i, j)^+$ is the successor of (i, j). All edges drawn have weight 1, except $H(v_{ij}, w_{ij1}) = G(i, j)$. Note that the same r_i and c_j occur in different beads. All beads are strung together on a "Möbus band" H (Fig. 16). Claims (10.2) and (10.3) obviously hold.

We first note that there are only two ways that a Hamiltonian circuit $\sigma \in HC(H)$ can traverse any bead (Fig. 17).

For an edge $e = (i, j)$ of G, we have the corresponding edge

$$d_e = (v_{ij}, w_{ij1})$$

in H. We define

$$P \subseteq \binom{t \times t}{2}$$

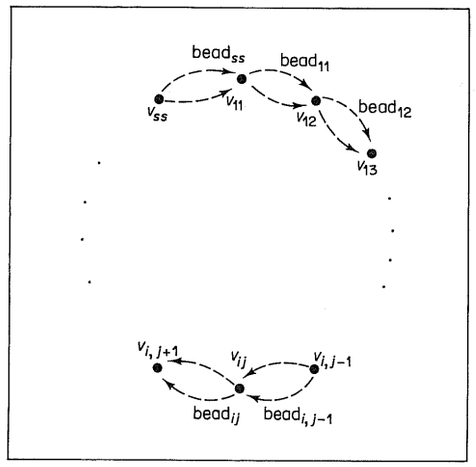


Fig. 16.

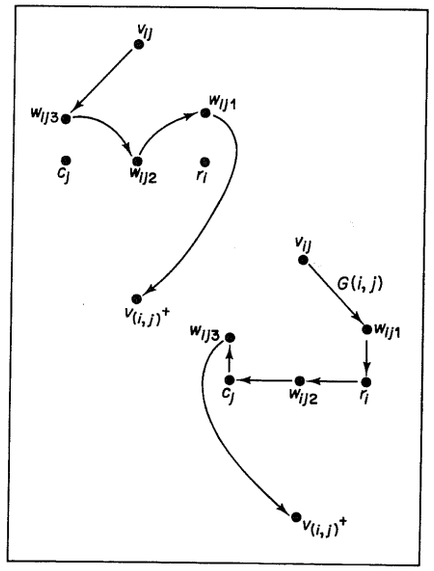


Fig. 17.

as the set of couples corresponding to Q in a natural way:

$$P = \{\{d_e, d_{e'}\} : \{e, e'\} \in Q\}.$$

We get a mapping

$$\alpha : HC(H; P) \to \mathrm{Sym}\,(G; Q),$$
$$\alpha(\sigma)(i) = j \quad \text{if} \quad \sigma(v_{ij}) = w_{ij1}.$$

One checks that $\alpha(\sigma)$ is a permutation in Sym (G), since for each i, σ passes through r_i exactly once, and through one c_j using the bead (i, j). It is then clear that also $\alpha(\sigma) \in \mathrm{Sym}\,(G; Q)$, and

$$p_\sigma(H) = p_{\alpha(\sigma)}(G).$$

The claim (10.1) is proven.

The next task, simulating coupled Hamiltonians by ordinary Hamiltonians, is solved by the following construction. We will consider "special graphs" H, which have a special edge $e_0 = (u, v)$ such that $H(u, j) = H(i, v) = 0$ for $j \neq v$, $i \neq u$, and $H(u, v) = 1$. Every $\sigma \in HC(H)$ with $p_\sigma(H) \neq 0$ contains e_0. For the construction, we are given a special graph $H \in R^{t \times t}$, where R is a commutative ring, and

$$P \subseteq \binom{t \times t}{2}, \qquad \{e, e'\} \in \binom{t \times t}{2} \backslash P,$$

such that $P \cup \{\{e, e'\}\}$ is a set of couples of edges of H, in which e_0 does not occur. We construct a special graph $C = \gamma(H, \{e, e'\})$, which contains H as a full subgraph, and such that

(11.1) $\mathrm{hc}(C; P) = \mathrm{hc}(H; P \cup \{\{e, e'\}\})$,
(11.2) $C \in R^{r \times r}$, where $r = t + 6$,
(11.3) every element of $R \backslash \{0, 1\}$ occurs an equal number of times in H and C.

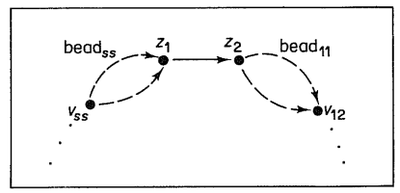


Fig. 18.

Given this construction, we prove the theorem just as in Theorem 5.4. In Proposition 4.6 we remarked that HC is p-definable. Using the notation n, e_n, $X = F \cup \{x_1, \ldots, x_n\}$, $B_0 \in X^{u \times u}$ and Q_0 from the proof of Theorem 5.4, it is sufficient to show that per $(B_0; Q_0)$ is a projection of HC_n for some small n, by Lemma 5.1. From (10.1), (10.2), and (10.3) we get $H \in R^{t \times t}$, where $t = 4u^2 + 2u$, and

$$P \subseteq \binom{t \times t}{2},$$

such that $\mathrm{hc}(H; P) = \mathrm{per}\,(B_0; Q_0)$. We make H into a special graph H_0 by splitting node v_{11} into two nodes z_1 and z_2, with special edge $e_0 = (z_1, z_2)$ (Fig. 18). Let $p = \# P$ and $P = \{c_1, \ldots, c_p\}$. Then $p = \# Q_0 \leqslant e_n + 1$. We get special graphs $H_0, H_1, \ldots, H_p$ and sets $P_0, P_1, \ldots, P_p$ by setting

$$P_i = \{c_{i+1}, \ldots, c_p\},$$
$$H_i = \gamma(H_{i-1}, c_i).$$

Properties (11.1), (11.2), (11.3) yield that for all $i \leqslant p$

(12.1) $\mathrm{hc}(H_i, P_i) = \mathrm{hc}(H_0, P_0) = \mathrm{per}\,(B_0, Q_0)$,
(12.2) $H_i \in R^{r_i \times r_i}$, where $r_i = 4u^2 + 2u + 6i$,
(12.3) every element of $R \backslash \{0, 1\}$ occurs an equal number of times in H_i and B_0.

In particular, per $(B_0; Q_0) = \mathrm{hc}(H_p)$ and $H_p \in R^{r \times r}$, where

$$r \leqslant (8e_n + 4n)^2 + 142e_n + 68n + 78.$$

It remains to construct C with properties (11.1), (11.2), (11.3). Assume that $e = (u, v)$, $e' = (u', v')$, and $e_0 = (u_0, v_0)$ is the special edge of H. C has the nodes of H plus 6 new nodes $a_1, a_2, a_3, b, c_1, c_2$. The edge weights are as in Fig. 19. All edges drawn have weight 1, unless otherwise specified. For nodes i, j of H, we have $C(i, j) = H(i, j)$ unless

$$(i, j) \in \{e, e'\}; \qquad C(u, v) = C(u', v') = 0.$$

The special edge of C is (u_0, c_1). Properties (11.2) and (11.3) obviously hold. Let

$$S = \{\sigma \in HC(C; P) : p_\sigma(C) \neq 0\},$$
$$S_1 = \{\sigma \in S : \sigma(a_2) = a_3\},$$
$$S_2 = S \backslash S_1.$$

One checks that any $\sigma \in S_1$ has the form in Fig. 20 and any $\sigma \in S_2$ has the form in Fig. 21.

Let $P' = P \cup \{\{e, e'\}\}$. We have a mapping

$$\alpha_1 : S_1 \to HC(H; P')$$

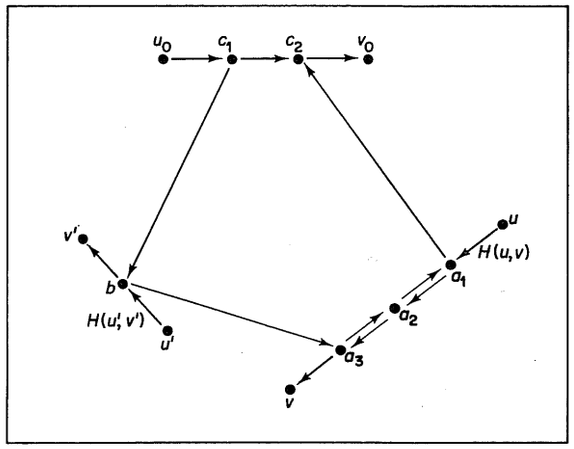


Fig. 19.

$$\alpha_1(\sigma)(i) = \begin{cases} v & \text{if } i = u, \\ v' & \text{if } i = u', \\ v_0 & \text{if } i = u_0, \\ \sigma(i) & \text{otherwise.} \end{cases}$$

Then $\#\alpha_1^{-1}(\tau) = 1$ for any $\tau \in im\alpha_1$, and if $\alpha_1(\sigma) = \tau$, then $p_\sigma(C) = p_\tau(H)$. Similarly, we get

$$\alpha_2 : S_2 \to HC(H; P')$$

$$\alpha_2(\sigma)(i) = \begin{cases} v_0 & \text{if } i = u_0, \\ \sigma(i) & \text{otherwise.} \end{cases}$$

Again, $\#\alpha_2^{-1}(\tau) = 1$ for $\tau \in im\alpha_2$, and if $\alpha_2(\sigma) = \tau$, then $p_\sigma(C) = p_\tau(H)$. Also, any $\tau \in HC(H; P')$ with $p_\tau(H) \neq 0$ is in $im\alpha_1 \cup im\alpha_2$. Claim (11.1) finally follows. □

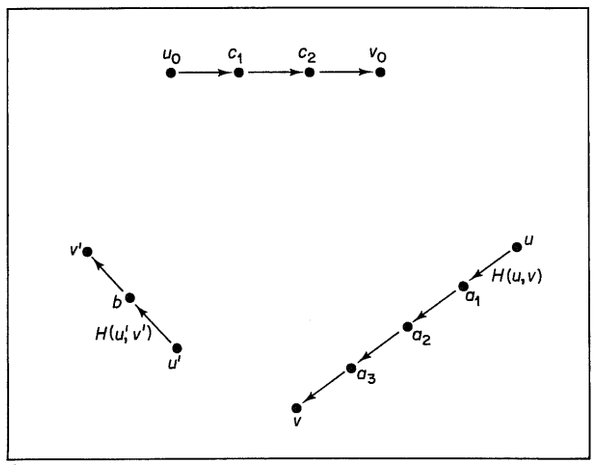


Fig. 20.

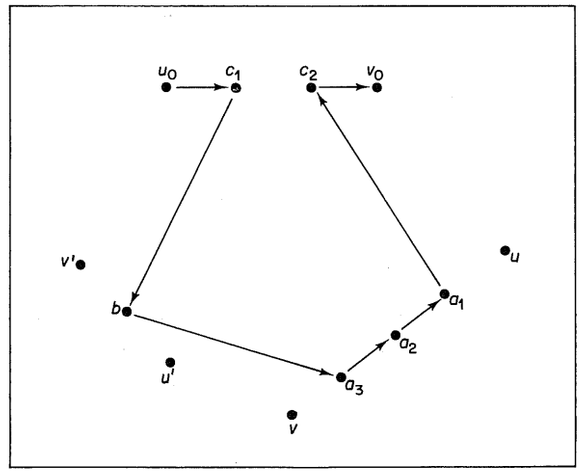


Fig. 21.

COROLLARY 5.7. *Under Valiant's hypothesis, neither* PER (*in characteristic different from two*) *nor HC are p-computable.*

Many thanks go to the participants of the DMV seminar, in particular to Volker Strassen and Michael Clausen, to the audience of a course in Toronto, in particular to Steve Cook, Faith Fich, and Charlie Rackoff, to the editor, Bruno Buchberger, and to the referees, for many helpful discussions and comments.

References

Baur, W., Strassen, V. (1982). The complexity of partial derivatives. *Theor. Comp. Sci.* **22,** 317–330.

Berkowitz, S. J. (1984). On computing the determinant in small parallel time using a small number of processors. *Inf. Pro. Lett.* **18,** 147–150.

Binet, M. J. (1813). Mémoire sur un système de formules analytiques, et leur application à des considérations géométriques. *J. Ecole polytechnique* **9,** 280–354.

Blokhuis, A., Seidel, J. J. (1984). An introduction to multilinear algebra and some applications. *Philips J. Res.* **39,** 111–120.

Borodin, A. (1982). Structured vs. general models in computational complexity. In: *Logic and Algorithmic,* Symposium in honour of Ernst Specker. *Enseign. Math.* **30,** 47–65.

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

Borodin, A., Munro, I. (1975). *The Computational Complexity of Algebraic and Numeric Problems.* New York: Elsevier.

Brent, R. P. (1974). The parallel evaluation of general arithmetic expressions. *J. Assoc. Comp. Mach.* **21,** 201–206.

Cauchy, A. L. (1813). Mémoire sur les fonctions qui ne peuvent obtenir que deux valeurs égales et de signes contraires par suite des transpositions opérées entre les variables qu'elles renferment. *J. Ecole polytechnique* **10,** 29–112.

Caviness, B. F., Epstein, H. I. (1978). A note on the complexity of algebraic differentiation. *Inf. Pro. Lett.* **7,** 122–124.

Chistov, A. L. (1985). Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic. Proc. Int. Conf. Foundations of Computation Theory. *Springer Lec. Notes Comp. Sci.* **199,** 63–69.

Cook, S. A. (1971). The complexity of theorem proving procedures. Proc. 3rd Ann. ACM Symp. Theory of Computing, 151–158.

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

Eberly, W. (1984). Very fast parallel matrix and polynomial arithmetic. Proc. 25th Ann. IEEE Symp. Foundations of Computer Science, Singer Island FL, 21–30. Tech. Rep. No. 178/85, Department of Computer Science, University of Toronto.

Fich, F., von zur Gathen, J., Rackoff, C. (1986). Complete families of polynomials. Manuscript.

Freeman, T. S., Imirzian, G., Kaltofen, E. (1986). A system for manipulating polynomials given by straight-line

programs. Tech. Rep. 86–15, Dept. Computer Science, Rensselaer Polytechnic Institute, Troy NY, September 1986.
Garey, M. R., Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness.* San Francisco: W. H. Freeman.
von zur Gathen, J. (1985). Irreducibility of multivariate polynomials. *J. Comp. Syst. Sci.* **31,** 225–264.
von zur Gathen, J. (1986). Parallel arithmetic computations: a survey. Proc. 12th Int. Symp. Math. Foundations of Computer Science, Bratislava. *Springer Lec. Notes Comp. Sci.* **233,** 93–112.
von zur Gathen, J. (1987). Permanent and determinant. To appear in *Linear Algebra and its Applications.*
von zur Gathen, J., Kaltofen, E. (1985). Factoring sparse multivariate polynomials. *J. Comp. Syst. Sci.* **31,** 265–287.
von zur Gathen, J., Seroussi, G. (1986). Boolean circuits versus arithmetic circuits. Proc. 6th Int. Conf. Computer Science, Santiago, Chile, 171–184.
Hammond, J. (1879). Question 6001. *Educ. Times* **32,** 179.
Hartmann, W. (1983). Zwei Probleme aus der Berechnungskomplexität. Dissertation, Universität Zürich.
Heintz, J., Sieveking, M. (1980). Lower bounds for polynomials with algebraic coefficients. *Theor. Comp. Sci.* **11,** 321–330.
Hyafil, L. (1979). On the parallel evaluation of multivariate polynomials. *SIAM J. Comp.* **8,** 120–123.
Ibarra, O. H., Moran, S. (1983). Probabilistic algorithms for deciding equivalence of straight-line programs. *J. Assoc. Comp. Mach.* **30,** 217–228.
Jung, H. (1985). On probabilistic time and space. Proc. 12th Int. Coll. Automata, Languages, and Programming. *Springer Lec. Notes Comp. Sci.* **194,** 310–317.
Kalorkoti, A. (1985). The formula size of the determinant. *SIAM J. Comp.* **14,** 678–687.
Kaltofen, E. (1986). Uniform closure properties of *p*-computable functions. Proc. 18th Ann. ACM Symp. Theory of Computing, Berkeley CA, 330–337.
Karp, R. M. (1972). Reducibility among combinatorial problems. In: (Miller, R. E., Thatcher, J. W., eds) *Complexity of Computer Computations*, pp. 85–104. New York: Plenum Press.
Kung, H. T. (1976). New algorithms and lower bounds for the parallel evaluation of certain rational expressions and recurrences. *J. Assoc. Comp. Mach.* **23,** 252–261.
van Lint, J. H. (1982). The van der Waerden conjecture: two proofs in one year. *Math. Intelligencer* **4,** 72–77.
Marcus, M., Minc, H. (1961). On the relation between the determinant and the permanent. *Illinois J. Math.* **5,** 376–381.
Meshulam, R. (1987). Private communication.
Meyer, A. R., Stockmeyer, L. J. (1972). The equivalence problem for regular expressions with squaring requires exponential space. Proc. 13th Ann. IEEE Symp. Switching and Automata Theory, 125–129.
Miller, G. L., Ramachandran, V., Kaltofen, E. (1986). Efficient parallel evaluation of straight-line code. Proc. Aegean Workshop on Computing, VLSI Algorithms and Architectures, Attica, Greece.
Minc, H. (1978). *Permanents.* Encyclopedia of Mathematics and its Applications, vol. 6. Addison-Wesley, Reading MA.
Ostrowski, A. M. (1954). On two problems in abstract algebra connected with Horner's rule. Studies in Math. and Mech. presented to Richard von Mises, pp. 40–48. New York: Academic Press.
Plaisted, D. A. (1984). New NP-hard and NP-complete polynomial and integer divisibility problems. *Theor. Comp. Sci.* **31,** 125–138.
Pólya, G. (1913). Aufgabe 424. *Arch. Math. Phys.* **20**, 271.
Reif, J. (1986). Logarithmic depth circuits for algebraic functions. *SIAM J. Comp.* **15,** 231–242.
Ryser, H. J. (1963). *Combinatorial Mathematics.* Washington: Math. Assoc. America.
Savage, J. (1976). *The Complexity of Computing.* New York: John Wiley and Sons.
Seress, A., Babai, L. (1987). Private communication.
Skyum, S., Valiant, L. G. (1985). A complexity theory based on Boolean algebra. *J. Assoc. Comp. Mach.* **32,** 484–502.
Strassen, V. (1972). Berechnung und Programm. I. *Acta Inf.* **1,** 320–335.
Strassen, V. (1973*a*). Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten. *Numer. Math.* **20,** 238–251.
Strassen, V. (1973*b*). Vermeidung von Divisionen. *J. reine u. angew. Math.* **264,** 182–202.
Strassen, V. (1974). Polynomials with rational coefficients which are hard to compute. *SIAM J. Comp.* **3,** 128–149.
Strassen, V. (1984). Algebraische Berechnungskomplexität. In: *Perspectives in Mathematics.* Basel: Birkhäuser Verlag.
Strassen, V. (1986). The work of L. G. Valiant. Proc. Int. Congress of Mathematicians, Berkeley CA.
Szegö, G. (1913). Zu Aufgabe 424. *Archiv Math. Phys.* **21,** 291–292.
Valiant, L. G. (1979*a*). Completeness classes in algebra. Proc. 11th Ann. ACM Symp. Theory of Computing, Atlanta GA, 249–261.
Valiant, L. G. (1979*b*). The complexity of computing the permanent. *Theor. Comp. Sci.* **8,** 189–201.
Valiant, L. G. (1982). Reducibility by algebraic projections. In: *Logic and Algorithmic,* Symposium in honour of Ernst Specker. *Enseign. Math.* **30,** 365–380.
Valiant, L., Skyum, S., Berkowitz, S., Rackoff, C. (1983). Fast parallel computation of polynomials using few processors. *SIAM J. Comp.* **12,** 641–644.

J. Symbolic Computation (1987) **4,** 173–190

Theory Links: Applications to Automated Theorem Proving

NEIL V. MURRAY AND ERIK ROSENTHAL †

Department of Computer Science, State University of New York at Albany, 1400 Washington Ave., LI 67A, Albany, New York 12222, U.S.A.

† *Department of Computer Science, Wellesley College, Science Center, Wellesley, Massachusets 02181, U.S.A.*

(*Received* 3 *April* 1986)

We develop the notion of *theory link*, which is a generalization of ordinary link to a set of literals that are simultaneously unsatisfiable relative to a given set of clauses. We show that theory links may be 'activated' in much the same manner as ordinary links when inferencing with respect to the given set of clauses. Several link deletion results are shown to hold for theory links, and some examples are presented using first-order theory links.

1. Introduction

In (Murray & Rosenthal, 1985a, 1985b, 1985c, 1987) we developed a graphical representation of NNF quantifier-free predicate calculus formulas and a new rule of inference, path resolution, which employs this representation. Stickel (1983, 1985a, 1985b) introduced theory resolution in which inferences depend on the existence of a 'black box' to implement a theory. Stickel designed theory resolution to be "a method of incorporating specialized reasoning procedures in a resolution theorem prover so that the reasoning task will be effectively divided into two parts: special cases ... are handled efficiently by specialized reasoning procedures, while more generalized reasoning is handled by resolution."

Path resolution operations hinge on the discovery of subgraphs (called resolution chains) which have the special property that all their c-paths contain a link. Many results from path resolution go through when we consider a generalization of link which we call a *theory link.* Intuitively, an ordinary link is a set of two c-connected (conjoined) literals such that under **no** assignment can both be true; a theory link is a set of n c-connected literals such that under no T-assignment, i.e., an assignment satisfying the axioms of theory T, can all be true. These specialized theory links can then be used in resolution-like procedures.

Finding a large resolution chain is hard in general, being essentially a subdeduction, i.e., the theorem proving problem on a (possibly non-explicit) subformula. One major advantage to the use of theory links is that they often represent large

0747–7171/87/050173+18