RFDUCIBILITY BY ALGEBRAIC PROJECTIONS

REDUCIBILITY BY ALGEBRAIC PROJECTIONS *

by L. G. VALIANT

ABSTRACT

Substitution as a notion of reduction between two polynomials or two Boolean functions is considered. It is shown that in a strong sense linear programming is a universal technique for computing discrete functions in polynomial time. The robustness of the notion of p-definability for polynomials is demonstrated by showing that alternative formulations, whether based on formula or program size, are equivalent. Also it is closed under most natural operations including substitution, taking coefficients and differentiation. These results facilitate the recognition of particular polynomials as p-definable. The polynomial analogue of the Meyer-Stockmeyer hierarchy collapses.

1. INTRODUCTION

The programming concept of a *subroutine* is well represented in theoretical computer science in the notion of *reducibility*. A function $A(\mathbf{x})$ is many-one reducible to function $B(\mathbf{y})$ if there is an easily computed transformation f such that $A(\mathbf{x}) = B(f(\mathbf{x}))$. A can be computed by computing f and then calling a subroutine for B. Traditionally this is the strictest notion considered. It is relaxed sometimes to allow several subroutine calls, or further computation after the call. In this paper we proceed in the opposite direction by considering reductions stricter still.

We say that $A(x_1, ..., x_n)$ is a *projection* of $B(y_1, ..., y_m)$ if after substituting for each y_i either an x_j or a constant, B equals $A(x_1, ..., x_n)$. Mathematically this notion has the obvious advantages of simplicity and of independence from any computational models. In programming terms it corresponds naturally to the concept of a *package* rather than subroutine,

* This article has already been published in *Logic and Algorithmic*, an international Symposium in honour of Ernst Specker, Zürich, February 1980. Monographie de L'Enseignement Mathématique N° 30, Genève 1982.

since the value of A can be obtained by calling B with the same inputs suitably reinterpreted. If a subroutine for B is available, A can be computed without further programming or precomputation on the input being required. The distinction between subroutines and packages can be of considerable practical importance as far as the effort required of a human user.

The results in this paper extend and complement those in [13], but can be read independently. There it was shown that the determinant is a universal function for all polynomials that can be computed fast sequentially or in parallel, and transitive closure is universal for Boolean functions computable fast in parallel. Here we complete this rough picture by showing that linear programming has the same universal role for Boolean functions that can be computed fast sequentially.

The concept of p-definability introduced in [13] serves to explain the difficulty of many intractable problems by providing an extensive class in which they are provably of maximal difficulty. In the polynomial case this suggests new techniques for identifying hard problems e.g. [6]. A shortcoming of the original treatment in [13] was that recognizing particular polynomials to be p-definable was sometimes possible only by indirect contrived means. The current paper remedies this by providing some useful equivalent definitions and various closure properties.

In the Boolean case p-definability provides an alternative approach to formulating such notions as NP, the Meyer-Stockmeyer hierarchy and polynomial space. It can be checked, for example, that the twenty-one NP-complete problems of Karp [7] are all p-projections of each other, and complete in our class. An important difference between our approach and the established one is that ours does not contain any assumptions about "Turing uniformity" (i.e. computational uniformity over infinite domains.) Thus, while this latter ingredient is a *sine qua non* in recursion theory and high-level complexity, it may be no more than an optional extra at the lower levels.

2. Definitions

Our notation is taken from [13] but is repeated here for completeness. We start with the case of polynomials.

Let F be a field and $F[x_1, ..., x_n]$ the ring of polynomials over indeterminates $x_1, ..., x_n$ with coefficients from F. P and Q will denote families of polynomials where typically

$$P = \{P_i \mid P_i \in F[x_1, \ldots, x_i], i \in X\},$$

where X is a set of positive integers. The arguments of P_i are exhibited sometimes as $P_i(x_1, \ldots, x_i)$ or $P_i(\mathbf{x})$ for short.

A *formula* f over F is an expression that is of one of the following forms: (i) "c" where $c \in F$, or (ii) "x_j" where x_j is an indeterminate, or (iii) "$(f_1 \circ f_2)$ where f_1 and f_2 are themselves formulae over F and $\circ$ is one of the two ring operators $\{+, \times\}$. The *size* of a formula is the number of operations of type (iii) needed in its construction, and is denoted by $|f|$. The *formula size* $|P_i|$ of polynomial P_i is the size of the minimal size formula that specifies it.

A *program* f over F is a sequence of instructions $v_i \leftarrow v_j \circ v_k$ $(i = 1, 2, \ldots, C)$ where (i) $j, k < i$, (ii) $\circ$ is one of the two ring operations $\{+, \times\}$, and (iii) if $j \leqslant 0$ then v_j is either an indeterminate x_m or a constant $c \in F$. The polynomial computed at v_i in the program is denoted by val (v_i) and its degree by deg (v_i). The size of a program is the number C of instructions. The *program size* $\|P_i\|$ of a polynomial P_i is the size of the minimal program that computes it.

Since formulae are just programs of a special form, in which each computed term can be used at most once, formula size is always at least as great as program size. A non-trivial converse relationship is due to Hyafil [5, 14].

A function from positive integers to positive integers we shall denote typically by t. Such a t is *p-bounded* if for some constant k, for all $n > 1$ $t(n) \leqslant n^k$. A family P has *p-bounded formula size* if for some p-bounded t for each i $|P_i| < t(i)$. P is *p-computable* iff for some p-bounded t for each i (a) $\|P_i\| < t(i)$ *and* (b) deg $(P_i) < t(i)$.

$Q_i \in F[y_1, \ldots, y_i]$ is a *projection* of $P_j \in F[x_1, \ldots, x_j]$ iff there is a mapping

$$\sigma: \{x_1, \ldots, x_j\} \to \{y_1, \ldots, y_i\} \cup F$$

such that $Q_i = P_j(\sigma(x_1), \ldots, \sigma(x_j))$.

Family Q is a *t-projection* of P if for each i for some $j < t(i)$ Q_i is the projection of P_j. It is the *p-projection* of P if it is the t-projection of P for some p-bounded t.

Among polynomial families that are generally regarded as intractable both mathematically and computationally, perhaps the simplest is the permanent [11] which is defined as follows.

$$\mathrm{Perm}_{n \times n}(x_{ij}) = \sum_{\pi} \prod_{i=1}^{n} x_{i, \pi(i)}$$

where summation is over the $n!$ permutations on n elements. This contrasts with the similar looking determinant which is tractable in both senses.

Another one is Hamiltonian Circuits:

$$HC_{n\times n}(x_{ij}) = \sum_{\pi} \prod_{i=1}^{n} x_{i,\,\pi(i)}$$

where summation is now over all $(n-1)!$ permutations consisting of a single cycle. Related to the latter are HG and $\#HG$ which are defined by

$$\sum_{\tau} N_{\tau} \cdot \prod_{x_{ij}\in\tau} x_{ij}$$

where summation is over those subsets τ of $\{ x_{ij} \mid 1 \leqslant i, j \leqslant n \}$ that *contain* a Hamiltonian circuit when interpreted as graphs. In HG $N_{\tau} = 1$. In $\#HG$ N_{τ} equals the number of Hamiltonian circuits in τ.

To treat Boolean computations we can use the same terminology as for polynomials except that $\{ +, \times \}$ are now interpreted as $\{$ or, and $\}$. For the above polynomials the graphical interpretation, where the value of x_{ij} denotes the presence or absence of edge (i, j), is natural. The permanent becomes the perfect matching function which is tractable [9]. HC, HG and $\#HG$ become identical and test for the presence of Hamiltonian circuits in a graph.

The Boolean versions of formulae, programs and projections differ only in the following ways: In formulae and programs an occurrence of an indeterminate x_i can now be either x_i or its negation $\bar{x}_i$, and constants need not occur at all. In a projection the mappings allowed are

$$\sigma : \{ x_1, \ldots, x_j \} \to \{ y_1, \bar{y}_1, y_2, \bar{y}_2, \ldots, y_i, \bar{y}_i \} \cup \{ 0, 1 \} .$$

The concept of degree is not defined and p-computability means just p-bounded program size. Lemma 18 in [4] ensures that this measure does correspond to the familiar notion of circuit size.

We shall be interested often in polynomials that have certain desired behaviour on $\{ 0, 1 \}$ inputs. In particular let $\mathrm{Sym}_n^r \in F[x_1, \ldots, x_n]$ be such that on any input from $\{ 0,1 \}^n$ it has value 1 or 0 according to whether exactly r of the inputs have value 1. A p-computable candidate for Sym_n^r is

$$(1 - T_n^n)\,(1 - T_n^{n-1}) \ldots (1 - T_n^{r+1})\,T_n^r$$

where T_n^i is the sum of the $\binom{n}{i}$ multilinear monomials of degree i, each with coefficient 1, i.e. the i'th elementary symmetric function.

3. p-DEFINABILITY

The concept of p-definability was introduced in [13] to characterize a large class of polynomials. Among naturally occurring polynomials of p-bounded degree it appears to contain a large majority. In this section we shall start to explore the extent of the class by considering various equivalent definitions of it. We start with the one given in [13] in its most simplified form.

Definition 1. A family P is *p-definable* over F iff *either* (a) $\exists Q$ over F of p-bounded formula size such that for all i

$$P_i = \sum_{\substack{(b_1, \ldots, b_i) \\ \in \{0,1\}^i}} \left[Q_i (b_1, \ldots, b_i) \prod_{b_k = 1} x_k \right] \qquad (\dagger)$$

or (b) P is the p-projection of a p-definable family.

If two polynomials P_i, Q_i are related as in part (a) of the definition we say that Q_i *defines* P_i. This relationship is to be interpreted as follows: P_i may or may not be a tractable polynomial but at least its coefficients are, i.e. there is a tractable Q_i whose values at the points $\{ 0, 1 \}^i$ are just the 2^i coefficients of P_i.

The permanent and determinant are widely recognised as being among the conceptually simplest polynomials. This is reflected here by the fact that part (a) of the above definition is sufficient to specify them. For example $\mathrm{Perm}_{n \times n} \{ x_{ij} \mid 1 \leqslant i, j \leqslant n \}$ is defined by

$$Q_{n \times n} = \left(\prod_{i=1}^{n} \sum_{j=1} y_{ij} \right) \left(\prod_{\substack{j = m \\ i \neq k}} (1 - y_{ij}\, y_{km}) \right).$$

Part (a) of the definition on its own, however, would be artificial and restrictive. Certainly only multilinear polynomials would be allowed. Also HC can be defined using (a) and (b) together (see Appendix 2) but apparently not with (a) alone.

Definition 1 is somewhat opaque. For example, it does not make clear even whether it covers all p-computable families. To resolve such questions the following formulation is useful.

Definition 2. A family P is *p-definable* over F iff *either* (a) $\exists Q$ over F that is p-computable such that for all i for some j $(0 \leqslant j \leqslant i)$

$$P_i(x_1, \ldots, x_i) = \sum_{\substack{b_{j+1}, \ldots, b_i \\ \in \{0,1\}^{i-j}}} \left[Q_i(x_1, \ldots, x_j, b_{j+1}, \ldots, b_i) \prod_{b_k = 1} x_k \right]$$

or (b) P is the p-projection of a p-definable family.

Later we shall see that this is indeed equivalent to Definition 1.

Remark 1. Every p-computable P is p-definable, for in Definition 2 we can take $Q_i = P_i$ and $j = i$.

Consider now a mathematically still simpler formulation that will be useful for proving closure properties.

Definition 3. A family P is *p-definable* over F if there is a p-computable Q and a polynomial t such that for all m there is an $i \leqslant t(m)$ such that

$$P_m(x_1, \ldots, x_m) = \sum_{\substack{b_{m+1}, \ldots, b_i \\ \in \{0,1\}^{i-m}}} Q_i(x_1, \ldots, x_m, b_{m+1}, \ldots, b_i) \quad .$$

PROPOSITION 1. *Definitions 2 and 3 are equivalent.*

Proof. Clearly P_m defined in Definition 3 can be translated into Definition 2 by taking the same defining Q_i, choosing $j = m$ and taking the projection $x_k = 1$ for $k = j + 1, \ldots, i$.

In the converse direction consider P_i as in Definition 2 (a). It clearly equals

$$\sum_{\substack{b_{j+1}, \ldots, b_i \\ \in \{0,1\}^{i-j}}} Q_i(x_1, \ldots, x_j, b_{j+1}, \ldots, b_i) \prod_{r=j+1}^{i} (x_r b_r + (1 - b_r))$$

which is of the form required in Definition 3 (but with a different Q_i !) □

For completeness and further simplicity we may also consider:

Definition 4. As Definition 3 but with Q restricted to p-bounded formula size.

PROPOSITION 2. *Definitions 1 and 4 are equivalent.*

Proof. Clearly Definition 1 implies Definition 4 exactly as Definition 2 implies Definition 3 (see proof of Proposition 1 above.)

To see the converse we use the form used in [13]. This is conveniently called Definition 1* as it is intermediate between Definitions 1 and 2. It is identical to Definition 1 except that line (†) is replaced by:

$$P_i = \sum_{\substack{b_{j+1}, \ldots, b_i \\ \in \{0,1\}^{i-j}}} \left[Q_i(x_1, \ldots, x_j, b_{j+1}, \ldots, b_i) \prod_{b_k = 1} x_k \right] \quad \text{for some } j \, .$$

Suppose now that a family P is p-definable in the sense of Definition 4. Then the argument in Proposition 1 showing that Definition 3 implies Definition 2 establishes that P is p-definable in the sense of Definition 1*. But Theorem 3 in [13] shows that any P so definable is the p-projection of HC and our Appendix 2 shows that HC is p-definable in the sense of Definition 1. The result follows. □

In Appendix 1 it will be shown that Definition 3 implies Definition 4. Together with Propositions 1 and 2 this will establish:

THEOREM 1. *Definitions 1, 2, 3 and 4 are all equivalent.*

4. CLOSURE PROPERTIES

A p-definable family P is *complete* over F if every family that is p-definable over F is the p-projection of P. It is known that several famous polynomials such as the permanent, hamiltonian circuits, the monomer-dimer polynomial and certain reliability problems are all complete for appropriate fields [6, 13]. In fact the projections required to establish these facts are all *strict projections* (i.e. no two indeterminates map to the same indeterminate). Hence these superficially dissimilar polynomials are related in the closest possible way: each one can be obtained from any other by fixing some indeterminates and renaming the others.

In the light of the simplicity of its completeness class the robustness of the notion of p-definability is perhaps remarkable. It can be explored conveniently by listing the operations under which it is closed.

First we consider the operation of substitution. The polynomials to be substituted can be viewed conveniently as an array.

Definitions. R is a *family array* over F if it is a set $\{ R^{m,n} \mid n \leqslant m \}$ of polynomials over F where $R^{m,n}$ has m indeterminates. It has *p-bounded degree* if for some p-bounded t $\deg(R^{m,n}) \leqslant t(m)$.

The various definitions of p-definability have analogues that are equivalent to each other for family arrays. For the current purpose it is best to adapt the fourth one:

Definition. Family array R is *p-definable* iff there is a p-bounded t such that for all m, n there is a T with formula size less than $t(m)$ such that

$$R^{m,n} = \sum_{\mathbf{b}} T(\mathbf{x}, \mathbf{b}) .$$

THEOREM 2. *If family P and array R are p-definable over F then so is the family* $P(R) = \{P_m(R^{m,1}, R^{m,2}, \ldots, R^{m,m})\}$

Proof. Consider the two polynomials:

$$S_i^1(\mathbf{x}) = \sum_{\mathbf{b}} Q^1(\mathbf{x}, b_1, \ldots, b_k) \quad \text{and} \quad S_j^2(\mathbf{y}) = \sum_{\mathbf{c}} Q^2(\mathbf{y}, c_1, \ldots, c_r).$$

If $k \geqslant r$ then their product is

$$\sum_{\mathbf{b}} \sum_{\mathbf{c}} Q^1(\mathbf{x}, b_1, \ldots, b_k) \cdot Q^2(\mathbf{y}, c_1, \ldots, c_r)$$

and their sum

$$\sum_{\mathbf{b}} Q^1(\mathbf{x}, b_1, \ldots, b_k) + Q^2(\mathbf{y}, b_1, \ldots, b_r)\, b_{r+1} \ldots b_k .$$

It follows by induction on the construction of formulae that if S is any family with p-bounded formula size then $S(R)$ is p-definable. Now choose S to be the family defining P. A typical member of $P(R)$ is

$$P_m(R) = \sum_{\mathbf{d}} S(R^{m,1}, \ldots, R^{m,m}, \mathbf{d}).$$

It follows by Theorem 5 that $P(R)$ is also p-definable. □

Remark 2. Closure of p-definability under addition ensures that Perm + 1 is p-definable. Since Perm is complete it follows that Perm + 1 is the p-projection of Perm. No direct proof of this is known and it is noteworthy that the corresponding question as to whether Det + 1 is the p-projection of Det appears to be open.

Remark 3. Reliability polynomials such as those considered in [6] can be recognised as p-definable by first considering distinct indeterminates p, q for each edge, and then substituting $q = 1 - p$.

The *coefficient* in $P_n \in F[x_1, \ldots, x_n]$ of the monomial $m = x_1^{i_1} \ldots x_n^{i_n}$ is the unique polynomial Q_n such that (i) $P_n = mQ_n + R_n$, (ii) Q_n and m have no indeterminate in common, and (iii) each monomial in R_n differs from m in the exponent of at least one indeterminate occurring in m.

The following closure property strengthens Proposition 9 in [13].

THEOREM 3. *If P is p-definable and R is a family such that for some p-bounded t, for each i, R_i is a coefficient in $P_{t(i)}$ then R is p-definable also.*

Proof. Suppose that $P_{t(i)}$ is the projection of

$$U = \sum_{\mathbf{b}} Q_j(\mathbf{b}) \prod_{b_k = 1} x_k$$

under σ. If R_i is the coefficient of $m = \prod y_s^{i_s}$ in $P_{t(i)}$ then it is the projection under σ of the sum of the coefficients in U of all products $\prod_{b_k=1} x_k$ such that for each s with $i_s \geqslant 1$.

$$|\{k \mid b_k = 1 \quad \text{and} \quad \sigma(x_k) = y_s\}| = i_s .$$

It therefore follows that R_i is a projection under σ' of

$$\sum_{\mathbf{b}} Q_j(\mathbf{b}) \prod_{s=1} \operatorname{Sym}^{i_s}(b_r \mid \sigma(x_r) = y_s) \prod_{b_k=1} x_k ,$$

where Sym is the polynomial defined in §2, and σ' modifies σ by mapping each element of

$$\{x_k \mid \sigma(x_k) = y_s \quad \text{and} \quad i_s \geqslant 1\}$$

to unity. □

THEOREM 4. *If P is p-definable then so are*

(i) $\{\partial P_i / \partial x_j \mid P_i \in P, \quad \text{any} \quad j\}$,

(ii) $\{\int P_i \, dx_j \mid P_i \in P, \quad \text{any} \quad j\}$, and

(iii) *the result of any p-bounded number of applications to P of differentiation or integration.*

Proof (i). Suppose that P_i is the projection of

$$\sum_{\mathbf{b}} Q_n(\mathbf{b}) \prod_{b_k=1} y_k$$

under $\sigma : \{y_k\} \to \{x_m\} \cup F$. For each power x_j^q of x_j we will take its coefficient, multiply it by $qz_1 \ldots z_{q-1}$ where $z_1, \ldots, z_{q-1}$ are new indeterminates, and finally project the original x_j to one and the new z's to x_j. Let $S = S_1 + S_2 + \ldots + S_d$ where $d = \deg(P_i)$ and $S_q(\mathbf{b}, \mathbf{c})$ equals:

$$q \cdot \operatorname{Sym}^q(b_r \mid \sigma(y_r) = x_j) \cdot \operatorname{Sym}_{q-1}^{q-1}(c_1, \ldots, c_{q-1}) \cdot \operatorname{Sym}_{d-q+1}^{0}(c_q, \ldots, c_d)$$

Then $\partial P_i / \partial x_j$ is the projection of

$$\sum Q_n(\mathbf{b}) \, S(\mathbf{b}, \mathbf{c}) \prod_{b_k=1} y_k \prod_{c_s=1} z_s .$$

Parts (ii) and (iii) follow by similar arguments. □

Finally we note that while p-definable families are rich in closure properties the p-computable ones are apparently not. Numerous natural mathematical operations seem not to preserve tractability. We can explore this phenomenon formally by showing that some easy polynomials become

complete when so operated on. A most convenient starting point is the following family T which is of p-bounded formula size:

$$T_{n^2+n} = \prod_{k=1}^{n} \sum_{i=1}^{n} x_{k,i}\, y_i \ .$$

Clearly (i) the coefficient of $y_1 \ldots y_n$ in T_{n^2+n},

(ii) $$\frac{\partial}{\partial y_1}\frac{\partial}{\partial y_2}\cdots\frac{\partial}{\partial y_n} T_{n^2+n} \ , \text{ and}$$

(iii) $$\left(\frac{3}{2}\right)^n \int_{-1}^{1} \cdots \int_{-1}^{1} [y_1 \ldots y_n\, T_{n^2+n}]\, dy_1 \ldots dy_n$$

all equal Perm $\{ x_{k,i} \}$.

In contrast, it is easy to see that all the other operations that we have considered preserve p-computability. This is immediate in the case of substitution. It can be shown to be true for $\partial P/\partial x_i$ and $\int P dx_i$ by considering a program for P, and decomposing it according to the powers of x_i at each instruction in the manner of [12].

5. A Non-existent Hierarchy

By analogies with recursion theory we can attempt to define the following hierarchy:

Definition. PD^0 = class of p-computable polynomial families. For $i > 0$ $P \in PD^i$ iff P is defined by some $Q \in PD^{i-1}$ in the sense of Definition 3.

That this hierarchy collapses in this algebraic case is easy to see:

Theorem 5. *For any* F *and any* $i > 0$ $PD^i = PD^{i+1}$.

Proof. It is clearly sufficient to prove $PD^1 = PD^2$. If $P \in PD^2$ then for each m

$$P_m(\mathbf{x}) = \sum_{\mathbf{b}} Q_i(\mathbf{x}, \mathbf{b})$$

where for some $R \in PD^0$ for each i

$$Q_i(\mathbf{x}, \mathbf{b}) = \sum_{\mathbf{c}} R_j(\mathbf{x}, \mathbf{b}, \mathbf{c}) \ .$$

Hence

$$P_m(x) = \sum_{\mathbf{b},\,\mathbf{c}} R_j(\mathbf{x}, \mathbf{b}, \mathbf{c})$$

which shows that $P \in PD^1$. □

We can attempt to generalise the definition of the above vacuous hierarchy by allowing the number of "alternations" to increase with the number of indeterminates.

Let t be any polynomial. Define $t\text{-}D^0$ to be the class of t-computable families. For $i > 0$ let $t\text{-}D^i$ be the class of families that are defined by some family in $t\text{-}D^{i-1}$ in the sense of Definition 3. Finally PD^* is the class of all families P such that for some t

$$P = \{P_i \mid P_i = Q_i \quad \text{for some} \quad Q \in t\text{-}D^{t^{(i)}}\}.$$

THEOREM 6. $PD^* = PD^1$

Proof. Similar to previous theorem. □

The above two results should be contrasted with the Boolean case where they still hold formally, but are no longer natural. The above definition of the successive levels PD^i is only natural if each level is a robust closure class. In Boolean algebra, however, PD^i is not known to be closed under complementation for any $i \geqslant 1$. Analogues of PD^i and PD^* where complementation is allowed at each level of alternation are not known to collapse, and are merely finite versions of the Meyer-Stockmeyer hierarchy, and PSPACE respectively [10].

A simple application of Theorem 5 is in recognising such polynomials as $\# HG$ as being p-definable. An intriguing open question is whether HG itself is p-definable for each F. If it is not then $P \neq NP$ (see Proposition 4 in [13]). If it is then the Meyer-Stockmeyer hierarchy and PSPACE can be simulated within p-definable families of polynomials.

6. UNIVERSALITY OF LINEAR PROGRAMMING

Here we consider a Boolean function family LP that corresponds to a linear programming problem and show that every p-computable family is the p-projection of it. Thus for computing discrete functions in polynomial time a package for LP for each input size is sufficient and no further programming is required. If we fix certain of the arguments of LP_i according to the particular function and input size being computed, the package becomes a program for the required function. That LP is itself p-computable follows from the recent result of Khachian [8].

The reader should note that several tractable problems in combinatorial optimisation are already known to have linear programming formula-

tions [9]. Our result shows that this is a universal phenomenon. It is related to the result in [3].

We define $LP_{2n(n+1)}$ to be the following Boolean function of arguments $\{ a_{ij}, b_{ij}, e_i, d_i \mid 1 \leqslant i, j \leqslant n \}$:

$$LP(a_{ij}, b_{ij}, e_i, d_i) = 1$$

if and only if the set of inequalities

$$\sum (\tilde{a}_{ij} x_j - \tilde{b}_{ij} x_j) \geqslant \tilde{e}_i - \tilde{d}_i$$

has a solution in real numbers, where each number $\tilde{a}_{ij}$, $\tilde{b}_{ij}$, $\tilde{e}_i$, $\tilde{d}_i$ is 1 or 0 according to whether the corresponding Boolean variable a_{ij}, b_{ij}, e_i, d_i is 1 or 0.

THEOREM 7. *Any p-computable family P of Boolean functions is the p-projection of LP.*

Proof. Consider some $P_m \in P$ with indeterminates $y_1, \ldots, y_m$, and a minimal program for it. The latter consists of a sequence of instructions of the form $v_i \leftarrow v_j \wedge v_k$ and $v_i \leftarrow v_j \vee v_k$, where $1 \leqslant i \leqslant C$ and each v_n with $n \leqslant 0$ equals some y_r or $\bar{y}_r$.

For any fixed assignment of truth values to $y_1, \ldots, y_m$ we can define a set E_0 of linear inequalities:

$$E_0 = \{ x_r \leqq 0 \mid r < 0 \text{ and } v_r \text{ has value } 0 \}$$
$$\cup \{ x_r \geqslant 1 \mid r < 0 \text{ and } v_r \text{ has value } 1 \}$$

For each sequence $v_1, v_2, \ldots, v_i$ we define E_i by induction from E_0:

$$E_i = \begin{cases} E_{i-1} \cup \{ x_j - x_i \geqslant 0, x_k - x_i \geqslant 0, x_i + 1 - x_j - x_k \geqslant 0 \} \\ \qquad \text{if} \quad v_i \leftarrow v_j \wedge v_k \ , \\ E_{i-1} \cup \{ x_j + x_k - x_i \geqslant 0, x_i - x_j \geqslant 0, x_i - x_k \geqslant 0 \} \\ \qquad \text{if} \quad v_i \leftarrow v_j \vee v_k \end{cases}$$

Claim 1. For any i, j $(j < i)$ every solution of E_i has $x_j \leqslant 0$, or every solution of E_i has $x_j \geqslant 1$.

Proof. The claim is true for E_o by definition. Assume inductively that it is true for E_{i-1}. (a) If $v_i \leftarrow v_j \wedge v_k$ then $x_j \leqslant 0$ implies that $x_i \leqslant 0$ since $x_j - x_i \geqslant 0$. Similarly if $x_k \leqslant 0$. In the remaining case $x_j, x_k \geqslant 1$ inequality $x_i + 1 - x_j - x_k \geqslant 0$ ensures that $x_i \geqslant 1$. (b) If $v_i \leftarrow v_j \vee v_k$ then $x_j \geqslant 1$

implies that $x_i \geqslant 1$ since $x_i - x_j \geqslant 0$. Similarly if $x_k \geqslant 1$. If $x_j, x_k \leqslant 0$ then $x_j + x_k - x_i \geqslant 0$ ensures that $x_i \leqslant 0$. □

Claim 2. If val $(v_i) = 0$ then $E_i \cup \{ x_i \leqslant 0 \}$ has a solution. If val $(v_i) = 1$ then $E_i \cup \{ x_i \geqslant 1 \}$ has a solution.

Proof. By induction on i it is easy to see that the point

$$x_j = \begin{cases} 1 & \text{if val } (v_j) = 1 \\ 0 & \text{if val } (v_j) = 0 \end{cases}$$

for $1 \leqslant j \leqslant i$ is a solution of E_i. □

Claim 3. If for some $i, j\ (j \leqslant i)$ $E_i \cup \{ x_j \geqslant 1 \}$ has a solution in reals then val $(v_j) = 1$.

Proof. By Claim 1, if $E_i \cup \{ x_j \geqslant 1 \}$ has a solution then $E_i \cup \{ x_j \leqslant 0 \}$ has no solution. Hence by Claim 2 val $(v_j) = 1$. □

Finally we observe that the given program of size C for P_m translates to $3C + 2m$ inequalities in E_C, of which the $2m$ of E_o depend on the values of $y_1, \ldots, y_m$, while the remaining $3C$ are fixed. It remains to note that P_m is the projection under σ of $LP_{2n(n+1)}$ for $n = 3C + 2m$, where σ maps $3C$ of the inequalities to those of $E_C - E_o$, and the remaining $2m$ values of i as follows. If v_i equals y_j or $\bar{y}_j$ then: $\sigma(a_{ik}) = \sigma(b_{ik}) = 0$ if $j \neq k$, $\sigma(d_i) = 0$, $\sigma(a_{ij}) = \sigma(e_i) = v_i$, $\sigma(b_{ij}) = \bar{v}_i$. □

ACKNOWLEDGEMENTS. It is a pleasure to thank Volker Strassen and Mark Jerrum for suggesting corrections and simplifications on a first draft of this paper.

APPENDIX 1

We show here that in the concept of p-definability it is immaterial whether the defining polynomials allowed are the p-computable ones or merely those of p-bounded formula size. We shall suppose that the family P is p-definable in the sense of Definition 3, i.e.

$$P_n(x_1, \ldots, x_n) = \sum_{b \in \{0,1\}^{m-n}} Q_m(x_1, \ldots, x_n, b_{n+1}, \ldots, b_m)$$

It will suffice to prove that any p-computable family, such as Q, is p-definable in the sense of Definition 4. By Theorem 5 it then follows that P itself is also p-definable in the sense of Definition 4.

It is known that any p-computable family of homogeneous polynomials has homogeneous program size at most polynomially larger than its unrestricted program size [12]. The inductive proof to follow assumes the former measure throughout and supports homogeneity. We shall assume that Q_m is itself homogeneous. If it were not then we would consider each of its homogeneous components separately in the same way.

Suppose that $Q_m(x_1, ..., x_m)$ has degree d and a minimal program ρ of complexity C. Let U be the subset of the computed terms $\{v_i\}$ such that (i) $\deg(v_i) > d/2$ and (ii) $v_i \leftarrow v_j \times v_k$ with $\deg(v_j) \leqslant d/2$ and $\deg(v_k) \leqslant d/2$. Let W be the subset $\{v_j\}$ such that $v_i \leftarrow v_j \times v_k$ or $v_j \leftarrow v_k \times v_j$ for some $v_i \in U$. For convenience rename the elements of U and W by $\{u_1, ..., u_r\}$ and $\{w_1, ..., w_s\}$ respectively.

Claim 1. There is a polynomial $S_{m+r+1}(x_1, ..., x_m, e_0, ..., e_r)$ of degree $\lfloor d/2 \rfloor + 1$ and homogeneous program complexity at most $2C + d$ such that

$$Q_m(\mathbf{x}) = \sum_{i=1}^{r} \text{val}(u_i) \cdot \text{compl}_i$$

where $\text{compl}_i = S_{m+r+1}(\mathbf{x}, \mathbf{e})$ when $e_0 = e_i = 1$ and $e_j = 0$ for $0 \neq j \neq i$.

Proof. In ρ replace each occurrence of u_i on the right hand side of an assignment by an occurrence of $e_i e_0^{\deg(u_i) - \lfloor d/2 \rfloor - 1}$. (Actually this would be simulated by a subprogram that raises e_0 to every power and multiplies by e_i as appropriate.) □

Claim 2. There is a polynomial $T_{n+s+1}(x_1, ..., x_m, c_0, ..., c_s)$ of degree $\lfloor d/2 \rfloor + 1$ and homogeneous program complexity at most $3C + d$ such that for each i $(1 \leqslant i \leqslant s)$

$$\text{val}(w_i) = T_{m+s+1}(\mathbf{x}, \mathbf{c})$$

when $c_0 = c_i = 1$ and $c_j = 0$ for $0 \neq j \neq i$.

Proof. Delete from ρ every instruction with degree greater than $d/2$. Add a subprogram equivalent to the set of instructions

$$z_i \leftarrow w_i \times c_i c_0^{\lfloor d/2 \rfloor - \deg(w_i)}$$

for $i = 1, ..., s$. Add further instructions to sum $z_1, ..., z_s$. □

Now for each i $\text{val}(u_i) = \text{val}(w_j)\,\text{val}(w_k)$ for some j, k specified by ρ. Hence each of the r additive contributions to Q_m is some product

$$T_{m+s+1}(\mathbf{x}, \mathbf{c})\, T_{m+s+1}(\mathbf{x}, \mathbf{c}')\, S_{m+r+1}(\mathbf{x}, \mathbf{e})$$

where $(\mathbf{c}, \mathbf{c}', \mathbf{e})$ is a fixed (0, 1)-vector of $2s+r+3$ elements. But any such vector can be specified by a conjunction of $2s+r+3$ Boolean literals. Consider the disjunction of the r such conjunctions and let $R(\mathbf{c}, \mathbf{c}', \mathbf{e})$ be the polynomial that simulates this Boolean formula at (0, 1) values. Then clearly

$$Q_m(x) = \sum T(\mathbf{x}, \mathbf{c})\, T(\mathbf{x}, \mathbf{c}')\, S(\mathbf{x}, \mathbf{e})\, R(\mathbf{c}, \mathbf{c}', \mathbf{e}) ,$$

where summation is over $(\mathbf{c}, \mathbf{c}', \mathbf{e}) \in \{ 0, 1 \}^{2s+r+3}$.
Let $A(C, d)$ be the upper bound over every homogeneous polynomial having degree d and homogeneous program complexity C, of the minimal size of formula needed to define it in Definition 4. Then the above recursive expression ensures that

$$A(C, d) \leqslant 3A(3C+d, \lfloor d/2 \rfloor + 1) + 0(C) .$$

Clearly also $A(C, 1) \leqslant 2C$. Hence if d is p-bounded in m then so is the solution to this recurrence. □

Appendix 2

For completeness we describe here a direct proof of the p-definability of HC in the sense of Definition 1. $HC_{n\times n}(x_{i,j})$ will be the projection under

$$\sigma(u_{k,m}) = 1 \quad \text{for} \quad 1 \leqslant k, m \leqslant n$$

of the polynomial in $\{ x_{i,j}, u_{k,m} \}$ defined by

$$Q_{n\times n}(y_{i,j}) \cdot Q_{n\times n}(z_{k,m}) \cdot R^1 \ldots R^n$$

with the association $y_{i,j} \leftrightarrow x_{i,j}$ and $z_{k,m} \leftrightarrow u_{k,m}$. Here $Q_{n\times n}$ is the polynomial that defines the permanent in §3. Its first occurrence with argument y plays exactly the same role as in the permanent and ensures a cycle cover. The intention of $z_{k,m}$ is to denote whether the k^{th} node in the circuit (starting from node 1, say) is node m. $Q_{n\times n}(z_{k,m})$ ensures that this intention is realised. For each k R^k captures the fact that if $z_{k,m}$ and $z_{k+1,r}$ are both 1 then $y_{m,r}$ must be also. In Boolean notation we require

$$y_{m,r} \vee (\bar{z}_{k,m} \vee \bar{z}_{k+1,r}) .$$

As is well known such Boolean formulae can be simulated by polynomials at $\{ 0, 1 \}$ values (e.g. see Proposition 2 in [13]). To guarantee just one monomial for each cycle we fix $R^1 = z_{11}$. □

REFERENCES

[1] BORODIN, A. and I. MUNRO. *The Computational Complexity of Algebraic and Numeric Problems*. American Elsevier, New York (1975).

[2] COOK, S. A. The complexity of theorem proving procedures. *Proc. 3rd ACM Symp. on Theory of Computing* (1971), 151-158.

[3] DOBKIN, D., R. J. LIPTON and S. REISS. Linear programming is log-space hard for *P*. *Inf. Proc. Lett. 8*: 2 (1979), 96-97.

[4] FISCHER, M. The complexity of negation-limited networks. *MAC Technical Memorandum 65*, MIT (1975).

[5] HYAFIL, L. On the parallel evaluation of multivariate polynomials. *Proc. 10th ACM Symp. on Theory of Computing*. (1978), 193-195.

[6] JERRUM, M. R. Applications of algebraic completeness to problems of network reliability and monomer-dimer systems. *Report CSR-45*, Computer Science Dept. Edinburgh University (1979). Also *SIAM J. on Comput.* (to appear).

[7] KARP, R. M. Reducibility among combinatorial problems. In *Complexity of Computer Computations* (R. E. Miller and J. W. Thatcher, eds.) Plenum Press, New York (1972).

[8] KHACHIAN, L. G. A polynomial algorithm in linear programming. *Soviet Math. Dokl 20* (1979), No. 1, 191-194.

[9] LAWLER, E. L. *Combinatorial Optimisation: Networks and Matroids*. Holt, Rinehart and Winston, New York (1976).

[10] MEYER, A. R. and L. J. STOCKMEYER. The equivalence problem for regular expressions with squaring requires exponential space. *Proc. 13th IEEE Symp. on Switching and Automata Theory* (1972), 125-129.

[11] MINC, H. *Permanents*. Addison Wesley (1978).

[12] STRASSEN, V. Vermeidung von Divisionen. *J. Reine und Angewandte Mathematik 264* (1973), 182-202.

[13] VALIANT, L. G. Completeness classes in algebra. *Proc. 11th ACM Symp. on Theory of Computing* (1979), 259-261.

[14] —— Computing multivariate polynomials in parallel. *Inf. Proc. Letters 11* (1980), 44-45 and *12* (1981), 54.

(Reçu le 25 avril 1981)

L. G. Valiant

Computer Science Department
Edinburgh University
Scotland