5. A NON-EXISTENT HIERARCHY

Objekttyp: **Chapter**

Zeitschrift: **L'Enseignement Mathématique**

Band (Jahr): **28 (1982)**

Heft 1-2: **L'ENSEIGNEMENT MATHÉMATIQUE**

PDF erstellt am: **05.06.2024**

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

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-