ON SETS HAVING ONLY HARD SUBSETS

P. FLAJOLET & J.M. STEYAERT

I.R.I.A. 78 Rocquencourt FRANCE

Abstract : We investigate properties of sets having no infinite subset in a given family of sets. We study the case when this family is defined by a complexity measure or one of the usual complexity notions in automata or recursive function theory.

Introduction :

The aim of this paper is to construct sets which have only "hard" non trivial subsets when we take for hard the various notions introduced in complexity theory. As independently noticed by Constable [3] and the authors [5] this concept is strongly reminiscent of the concept of immune set in recursive function theory ; indeed, an immune set is an infinite set that has no infinite recursively enumerable (r.e.) subset.

In the case of abstract complexity classes, Constable shows the existence of such sets using a diagonal argument based on list processing. We proceed differently and use a more natural method closer to Post's original construction of an immune set, as described in Rogers [13]. In section 1 we give a procedure for obtaining $\mathcal{B}$-immune sets when $\mathcal{B}$ is any denumerable class of subsets of N, the set of non negative integers. In section 2 we study the case when $\mathcal{B}$ is a complexity class in the scope of Blum's complexity theory [1] ; we then consider classes of resource bounded Turing machines and give an upper bound on the resource needed to perform the algorithm. As a corollary, we show that there exist exponential (w.r.t. time recognition) sets having no infinite polynomial subsets ; this enables us to conjecture a similar property for the set of prime numbers.

Finally the construction turns out to apply to other complexity notions in the field of language theory or subrecursive programming languages. In section 3, we give abstract conditions under which the basic construction applies to subrecursive classes. In a similar setting, we derive a class of undecidability results dealing with membership problem (i.e. determining whether an element in a larger class belongs to a smaller one). Applications to properties about program size are briefly sketched.

1. The basic construction.

In this section we give a construction of an infinite set $I \subset N$ which has no infinite set in a denumerable class $\mathcal{B}$ of subsets of N. We first notice that I has no infinite subset belonging to $\mathcal{B}$ iff its complement intersects all the infinite elements of $\mathcal{B}$; this follows from equivalence between (1) and (2) below :

J. Loeckx (ed.), *Automata, Languages and Programming*

(1) $\forall A$ infinite $[A \subset I \rightarrow A \notin \mathcal{B}]$

(2) $\forall A$ infinite $[A \in \mathcal{B} \rightarrow A \cap \bar{I} \neq \emptyset]$

The problem of finding such an I always has a trivial solution which consists in taking I finite. Putting these solutions away we define :

Definition : Let $\mathcal{B}$ be a class of subsets of N :

- A set is *$\mathcal{B}$-immune* iff it is infinite and has no infinite subset which belongs to $\mathcal{B}$.
- A set is *$\mathcal{B}$-simple* iff it is co-infinite and intersects every infinite element of $\mathcal{B}$.

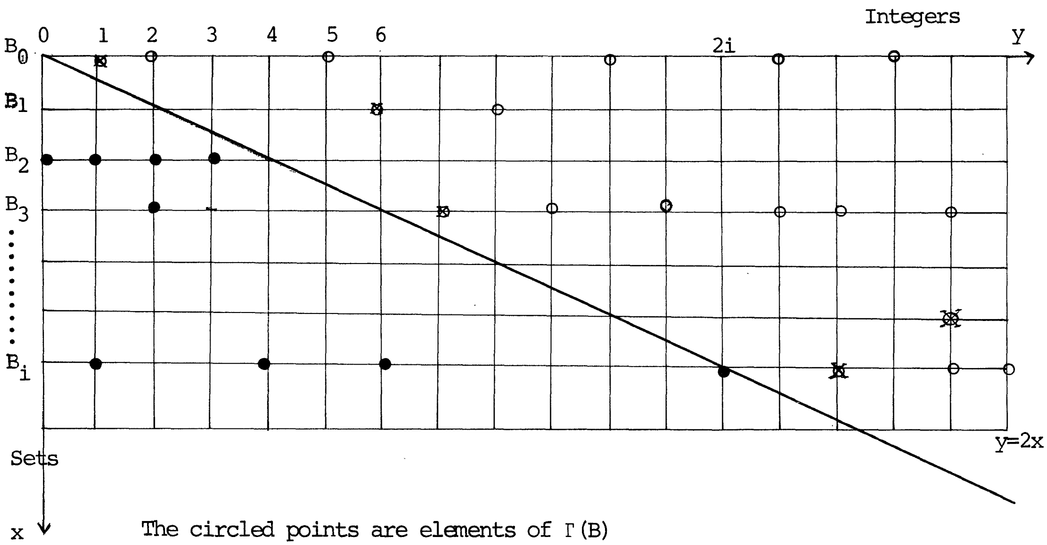
These two concepts are clear extensions of the corresponding notions in recursive function theory (cf Rogers [13]). In our framework, an immune set in the classical sense is an r.e-immune set and a simple set is an r.e-simple set which is itself r.e. . Notice that this extension includes as a subcase the one introduced independently by Constable [3].

We now turn to the basic construction which is adapted from Post's original one, as described in Rogers.

Let $\mathcal{B}$ be a denumerable class of subsets of N and $\{B_i\}$ be an enumeration of $\mathcal{B}$. A $\mathcal{B}$-simple set S can be obtained by choosing an integer in every infinite element of $\mathcal{B}$, simultaneously ensuring that we keep $\bar{S}$ infinite. We consider the set

$$\Gamma(B) = \{(x,y)/y \in B_x \wedge y > 2x\}$$

and obtain a $\mathcal{B}$-simple set S by a *choice* in $\Gamma(B)$ according to the first coordinate, i.e. for each x we select a y such that $(x,y) \in \Gamma(B)$ if there exists some.



The circled points are elements of $\Gamma(B)$

The crossed points are those selected in $\Gamma(B)$

Figure 1

S intersects every infinite element of $\mathcal{B}$; furthermore in the interval $[0,2n]$ we may choose only elements of $\mathcal{B}_0\mathcal{B}_1\ldots\mathcal{B}_n$ and thus S is co-infinite ; therefore S is $\mathcal{B}$-simple and $\bar{S}$ is $\mathcal{B}$-immune.

Having in mind the case when $\mathcal{B}$ is recursive † we take as a choice operator, a boundary operator : that is, for each x, we select the smallest y, if any, such that $(x,y)\in\Gamma(B)$ (cf Fig 1).

The $\mathcal{B}$-simple set S is thus defined by

$$y\in S \quad \text{iff} \quad \exists x[(x,y)\in\Gamma(\mathcal{B})\wedge\forall t<y\,(x,t)\notin\Gamma(\mathcal{B})]$$
$$\text{iff} \quad \exists x[y\in B_x\wedge y>2x\wedge\forall t<y\neg(t\in B_x\wedge t>2x)$$

The existential quantifier in front can be changed into a bounded quantifier ; this follows from the fact that $\exists x[y>2x\wedge\ldots]$ is equivalent to $\exists x<y[y>2x\wedge\ldots]$

Hence :

(1) $$y\in S \quad \text{iff} \quad \exists x<y[y\in B_x\wedge y>2x\wedge\forall t<y\neg(t\in B_x\wedge t>2x)]$$

By the same token we obtain the relation defining a $\mathcal{B}$-immune set $I=\bar{S}$ as :

(2) $$y\in I \quad \text{iff} \quad \forall x<y[(y\in B_x\wedge y>2x)\rightarrow\exists t<y\,(t\in B_x\wedge t>2x)]$$

It is clear that if the class $\mathcal{B}$ is recursive we define a $\mathcal{B}$-immune set which is recursive.

Formula (2) yields a procedure which, given a recursive enumeration B of $\mathcal{B}$ and an integer y, decides whether y does or does not belong to the set I defined by (2).

Algorithm I :

```
boolean procedure immune (y)
   integer x,y,t ;
   boolean im ;
begin
   immune := true ;
   for x:=o step 1 until (y-1) ÷2 do
       begin
            if B(x,y)
               then begin im:= false ;
               for t:=2x+1 step 1 until y-1 do im:=im or B(x,t) ; end
               else im:= true ;
            immune:= immune and im ; end ;
end proc immune ;
```

Remark : It should be noticed that the main property of the *threshold* relation $y>2x$

---

† a class $\mathcal{B}$ is recursive (subrecursive...) if it has a recursive (subrecursive...) enumeration, i.e. the predicate $B(x,y)\equiv y\in B_x$ is recursive (subrecursive).

used in defining $\Gamma(B)$ is that function $y=2x$ has a more-than-linear growth. Hence any function having this property can be used instead ; for example $y=x^2$ or $y=2^x$. A variant of algorithm I with the threshold relation $y>2^x$ will be used later.

2. Application to complexity classes of sets.

In the first part of this section we show that the basic construction applies to complexity classes defined in any measure that satisfies Blum's axioms [1].

Definition (Blum) : An abstract complexity measure is a couple of binary partial recursive functions (ϕ,Φ) such that :

(0) ϕ is an acceptable numbering of one place recursive functions

(1) $\mathrm{dom}\ \phi = \mathrm{dom}\ \Phi$

(2) the graph of Φ is recursive

Let $C(t)$ be the class of sets whose complexity is bounded by t almost everywhere (a.e)

$C(t) = \{A/\ \exists i \phi_i = \text{car } A \wedge \Phi_i \leq t \text{ a.e}\}$, where car $A(x)$ = *if* $x\in A$ *then* 0 *else* 1.

Since classes $C(t)$ are not recursive in general we shall often be lead to consider larger classes $D(t)$ which are recursive and contain classes $C(t)$ as subclasses.

Fact 1 : For all recursive t, there exists a class of sets $D(t)$ which is recursive and contains the complexity class $C(t)$.

Let s be a recursive function which takes infinitely often (i.o) any value : $\forall x \ \overset{\infty}{\exists} y \ \ s(y)=x$. Consider the binary relation $V_t(x,y)$ defined by $V_t(x,y) \equiv \Phi(s(x),y) \leq \max(x,t(y)) \wedge \phi(s(x),y)=0$. $V_t(x,y)$ is clearly recursive and enumerates a class $D(t)$ which contains $C(t)$.□

The following facts may be helpful to get deeper insight on the significance of immune sets.

Fact 2 · For all recursive t, there exists a set whose complexity is greater than t infinitely often (i.o).

This is essentially Blum's compression theorem which states that arbitrarily complex sets exist.□

Fact 3 : For all recursive t and all infinite recursive A, there exists B, subset of A, whose complexity is greater than t i.o.

This means that an infinite recursive set always has subsets of arbitrarily high complexity.

Let α be a unary recursive function such that $\alpha(A)=N$. The function α can be defined in the following way : let $A=\{a_0 a_1 \ldots a_n \ldots\}$; $a_{n-1}<p\leq a_n \Longrightarrow \alpha(p)=n$, the a_i being in increasing order. Let V_t be the recursive enumeration of the class $D(t)$ described in fact 1. The set B is defined by

$$B(x) \equiv A(x) \wedge \neg V_t(\alpha(x),x)$$

We then have $B\subset A$ and a straightforward diagonal argument shows that $B\notin D(t)$ and thus $B\notin C(t)$.□

Fact 4 : Some complex sets have infinite subsets of very low complexity.

Take, for instance, the set of non primes which is NP and admits the set of even integers as an "easy" subset.□

We now state

Proposition 1 : For all recursive t, there exists a recursive $C(t)$-immune set.

Proof : The basic algorithm does not apply directly since $C(t)$ is not recursive in general. We thus construct a $D(t)$-immune set using the recursive relation V_t. Since $C(t) \subset D(t)$ this set is also $C(t)$-immune.□

The next step consists in showing that the complexity of that $C(t)$-immune set can be bounded above uniformly in t.However, in order to avoid conflict with Borodin's gap theorem, we must further assume that t is self-computable, i.e. $t=\Phi_i$ for some i. (and t is total).

Theorem 1 : There exists a recursive r such that for every increasing self-computable t satisfying $\forall x\ t(x)\geq x$, $C(r\circ t)$ contains a $C(t)$-immune set.

Proof : It proceeds through standard methods of complexity theory. First notice that in the construction of proposition 1, an index for the $C(t)$-immune set can be obtained effectively from an index of t. Let ξ be the total recursive function s.t. :

ϕ_i total $\Longrightarrow \xi(i)$ is an index for the characteristic function of the $C(\phi_i)$-immune set constructed above.

We apply now an extended combining lemma (cf. Hartmanis-Hopcroft [8]). We define

$$p(i,n,m)=\begin{cases}\Phi_{\xi(i)}(n) \text{ if } \forall k\leq n \quad \Phi_i(k)\leq m\\ \\ 0 \qquad \text{otherwise}\end{cases}$$

Let $h(n,m)=\max\limits_{i\leq n} p(i,n,m)$. Clearly $h(n,m)$ is increasing in n and satisfies $\Phi_{\xi(i)}(n)\leq h(n,\Phi_i(n))$ whenever ϕ_i is total and $n\geq i$.

The proof is completed by taking $r(x)=h(x,x)$.□

We now turn to complexity classes defined by time or tape bounded Turing machine computations. The complexity of computations is measured by the amount of resource necessary to process all inputs of length n. A straight application of algorithm I would not give good results in the case of time bounded classes since the number of integers of length less than n is exponential in n and we should diagonalize over all of them. We shall thus diagonalize only on integers whose binary representation is a sequence of ones. This is in fact equivalent to applying diagonalisation on integers represented in unary notation and then transferring the results for Turing machines with binary encoded inputs.

Notations : UTIME(t) - resp. BTIME(t) - is the class of sets given in unary - resp. binary - notation which are recognizable within time bound t by some multitape Turing machine. The tape bounded classes UTAPE and BTAPE are defined similarly.

We also need an analogue of self computable fonctions : a function t is time-constructible iff it can be computed in unary notation within time t; tape constructible functions are defined similarly.

We shall use a slightly modified version of a simulation theorem of Hennie-Stearns [9].

Lemma 1 : Let t be a time constructible function. There exist a numerisation δ of $t(n)$ time bounded multitape Turing machines working on unary inputs, a function $c(x) \leq k.\log^2 x$, and a universal machine U which, given the description $\delta(T)$ of machine T and input x in unary notations yields the result of T upon x in time $c(\delta(T)).t(x).\log t(x)$.

Proof (outlined) : We first describe the numerisation δ.

Let T be a multitape Turing machine. The transition table of T is changed into a string on a fixed alphabet ; this string is then encoded on N using successive applications of the function $\gamma(x,y) = 2^x(2y+1)$. We thus obtain an integer $\delta(T)$ which represents the machine T. By construction, the length of the transition table of T is at most $d = \log \delta(T)$. Furthermore it is easy to see that decoding $\delta(T)$ requires at most $l.\delta(T)$ step on a Turing machine, for some constant l.

The universal machine U is basically the one described by Hennie-Stearns. Given inputs $\delta(T)$ and x, it simulates T on x using Hennie-Stearns technique but, instead of having the description of T stored in its control, U has to explore it on one of its tapes. Since every symbol of T is represented by at most d cells on U and since U has to explore the description of T at every simulation step, U requires at most $k.d^2.t.\log t$ steps to simulate t steps of T.□

Proposition 2 : Let t be any increasing time-constructible function.

$$\mathrm{UTIME}(x^{1+\varepsilon}\, t(x) \log t(x)) \text{ contains a UTIME } (t(x))\text{-immune set.}$$

Proof : Lemma 1 provides us with a universal relation $U(i,j)$ for UTIME(t).Checking membership of y to the immune set given by algorithm I requires about y^2 computations of U. This number can be improved to $y.\log y$ using the variant of algorithm I with $y > 2^x$ as a threshold relation. Let J be the UTIME(t)-immune set defined by this construction ; determining whether $y \in J$ requires computations of $U(i,j)$ with $i < \log y$ and $2^i < j \leq y$. We recall that $U(i,j)$ is computable in time $\log^2 i.t(j).\log t(j)$.Due to additive properties of computation time, $y \in J$ can thus be evaluated in time $\tau(y)$ such that :

$$\tau(y) = \sum_{\substack{i \leq \log y \\ i < j \leq y}} \log^2 i.t(j).\log t(j)$$

$$\leq y \,.\, \log y \,.\, (\log \log y)^2 \,.\, t(y) \,.\, \log t(y)$$

since t is increasing.

Hence $\tau(y) = O(y^{1+\varepsilon} t(y).\log t(y))$.□

Theorem 2 : Let t be any increasing time constructible function.

Then BTIME $(x^{1+\varepsilon}\hat{t}(x) \log t(x))$ contains a BTIME $(t(x))$-immune set.

Proof : Let J be defined as above. Consider $K = \{2^n - 1 / n \in J\}$. The key property is that unary representation of elements in J coincide with binary representations of elements in K. K clearly belongs to $BTIME(x^{1+\epsilon}t(x)\log t(x))$; furthermore K is BTIME(t) immune. Assume to the contrary that there exists some infinite $K'\subset K$ which belongs to BTIME(t). The set $J'=\{\lceil \log p\rceil / p\in K'\}$ whose unary representations coincide with binary representations for K' would then be in UTIME(t), contradicting the fact that J is UTIME(t)-immune.□

An analogous and even simpler construction can be done for tape classes. Since there exists a universal $c(\delta(T)).l(x)$ tape bounded machine which is universal for $l(x)$ tape bounded Turing machine we get :

Theorem 2': Let l be an increasing tape constructible function. Then $BTAPE(l(x).\log x)$ contains a $BTAPE(l(x))$-immune set.

Notice that algorithm I requires hardly more resource than a plain diagonal argument.

As a corollary we obtain :

Corollary : There exist exponential time recognisable sets having no polynomial time infinite subsets.

Taking $t(x)=x^{\log x}$ we obtain in $BTIME(x.\log^4 x.x^{\log x})$ a $BTIME(x^{\log x})$ immune set. Noticing that

$$\bigcup_{p\in N} BTIME(x^p) \subset BTIME(x^{\log x})$$

and

$$BTIME(x.\log^4 x.x^{\log x}) \subset BTIME(2^x)$$ completes the proof.□

The set of primes which is exponential time recognisable has been conjectured not to be polynomial ; the above corollary shows possibility of a stronger conjecture which might account for the failure of various attempts at finding easy sublaws to the law of prime numbers.

Conjecture : The set of prime numbers admits no infinite polynomial time recognisable subset.

As pointed out by J. Berstel, this conjecture is linked to the existence of an infinity of Mersenne primes which are of the form $M_p=2^p-1$, where p is prime. The Lucas-Lehmer criterion states that :

Lucas Criterion : Let p be an odd prime number ; then $M_p=2^p-1$ is prime if $L_{p,p-2}=0$ where the sequence $L_{p,0}=4$ and $L_{p,n+1}=(L_{p,n}^2-2)$ modulo M_p.

A Polynomial time algorithm for Mersenne prime recognition can easily be derived from Lucas Criterion. Hence, if true, our conjecture would imply that the set of Mersenne primes be finite.

3. Application to subrecursive classes.

In the former section, we considered abstract complexity classes and resource bounded classes. In this section, we study the case of classes of formal languages and subrecursive classes. Finally, we derive abstract conditions for membership problems to be unsolvable.

Let Σ be a finite alphabet : Reg $[\Sigma]$, CF $[\Sigma]$ and CS $[\Sigma]$ will denote respectively the class of regular sets, the class of context-free languages and the class of context-sensitive languages over Σ. The classical languages $L_0=\{a^n b^n / n\in N\}$ and $L_1=\{a^n b^n c^n / n\in N\}$ are seen to be respectively Reg-immune and CF-immune by application of the usual star lemmas.

In the case of subrecursive classes, such algebraic lemmas do not hold in general. However, one can consider Algorithm I in a way readily suited for application to most subrecursive classes.

Thoerem 3 : Let $\mathcal{B}$ be denumerable class of subsets of N and $\mathcal{C}$ be a class of predicates over N ; suppose $\mathcal{C}$ satisfies both conditions :

(a) $\mathcal{C}$ contains a universal relation for $\mathcal{B}$ (i.e. a relation which enumerates $\mathcal{B}$) and the binary relation $\lambda uv.\ u>2v$.

(b) $\mathcal{C}$ is closed under explicit transformations (i.e. changes of variables), boolean operations and bounded quantification.

Then $\mathcal{C}$ contains a $\mathcal{B}$-immune set.

Proof : Conditions *(a)* and *(b)* ensure that Algorithm I can be applied to $\mathcal{B}$ and that the resulting set will belong to $\mathcal{C}$.□

Conditions of theorem 3 are fulfilled when we take couple $(\mathcal{C},\mathcal{B})$ to be (recursive, primitive recursive) or in general $(\mathcal{R}^{n+1},\mathcal{R}^n)$ or for $n\geq 3$ $(\mathcal{E}^{n+1},\mathcal{E}^n)$ where $\mathcal{R}^n$ and $\mathcal{E}^n$ are respectively Peter and Grzegorczyk classes ; hence :

Corollary : There exists a primitive recursive set which is elementary recursive-immune.

For all n, $\mathcal{R}^{n+1}$ contains an $\mathcal{R}^n$-immune set.

For all $n\geq 3$, $\mathcal{E}^{n+1}$ contains an $\mathcal{E}^n$-immune set.

Theorem 3 also applies to classes of twoway multihead finite automata : indeed, for all k, there exists in the 2k+8 head class-A^{2k+8}- a set which is immune for the k head class -A^k- (cf [6]).

We now use the notion of immune sets to formulate in an abstract setting general conditions under which the finiteness problem for a recursive class $\mathcal{C}$ reduces to the membership problem (of elements in $\mathcal{C}$ to a smaller recursive class $\mathcal{B}$). More precisely, let $\mathcal{B}$ and $\mathcal{C}$ be two denumerable classes of subsets of N with $\mathcal{B}\subset\mathcal{C}$; let C be an enumeration of $\mathcal{C}$.

The membership problem MEMBER $[C;\mathcal{B}]$ is defined by : MEMBER $[C;\mathcal{B}](x)$ iff $C_x\in\mathcal{B}$. The finiteness problem for C - denoted FINITE [C]-corresponds to the special case of the membership problem when $\mathcal{B}$ coincides with the class $\mathcal{F}$ of finite subsets of N.

We shall suppose throughout $\mathcal{C}$ is recursive, $\mathcal{B}$ has a recursive enumeration B (the properties do not actually depend on which enumeration is chosen). In that case, the membership problem is written :

MEMBER $[C;\mathcal{B}](x)$ iff $\exists y\, \forall t\, [C_x(t) \equiv B_y(t)]$

It is thus a Σ_2-predicate in the sense of Kleene's arithmetical hierarchy.

The finiteness problem for C is often Σ_2-complete. It suffices that B allow a simulation of the computation sequences of some universal machine class (Turing-machines, Register Machines...) ; for instance $\mathcal{B}$ can be taken to be the simpler Grzegorczyk class $\mathcal{E}^0$ or the class of context-sensitive languages or the class of multihead finite automata recognisable languages.

Theorem 4 below gives sufficient conditions for FINITE [C] to reduce to MEMBER $[C;\mathcal{B}]$, and thus provides a useful tool for showing membership problems to be Σ_2-complete.

<u>Definition</u> : Let H and K be subsets of N ; we define the set composition of H and K-denoted $H*K$-, as the set of those elements in H whose rank in the natural enumeration of H belongs to K.

Let $H = \{h_0, h_1, h_2, \dots\}$ with $h_0 < h_1 < h_2 < \dots$

Then $H*K = \{h_j / j \in K\}$.

<u>Theorem 4</u> : If $\mathcal{B}$ and $\mathcal{C}$ satisfy the following conditions :

(a) $\mathcal{B}$ contains all finite sets : $\mathcal{F} \subset \mathcal{B}$

(b) $\mathcal{C}$ contains a $\mathcal{B}$-immune set

(c) $\mathcal{C}$ enumerated by C is effectively closed under set composition

then FINITE [C] reduces to MEMBER $[C;\mathcal{B}]$. Hence if FINITE [C] is Σ_2-complete then MEMBER $[C;\mathcal{B}]$ also is.

<u>Proof</u> : $\mathcal{C}$ enumerated by C is effectively closed under set composition if there exists a (total) recursive function τ which represents set composition w.r.t. C, i.e. such that $C_i * C_j = C_{\tau(i,j)}$. Let Δ be a $\mathcal{B}$-immune set ; take S to be an arbitrary set in $\mathcal{C}$ and consider $\Delta * S$. Two cases arise :

- if S is finite, $\Delta * S$ is a finite subset of Δ ; hence from *(a)* $\Delta * S \in \mathcal{B}$.

- if S is infinite, $\Delta * S$ is an infinite subset of Δ which does not belong to $\mathcal{B}$ by immunity of Δ.

Now let d and i be indices for Δ and S :

$\Delta = C_d$ and $S = C_i$

$S = C_i \in \mathcal{F}$ iff $\Delta * S = C_{\tau(d,i)} \in \mathcal{B}$.

This shows that FINITE [C] reduces to MEMBER $[C;\mathcal{B}]$.

Thus, if FINITE [C] is Σ_2-complete, MEMBER $[C;\mathcal{B}]$ also is. $\Box$

Let E^n, R^n and A^n denote recursive enumerations of $\mathcal{E}^n, \mathcal{R}^n$ and $\mathcal{A}^n$. We can state :

<u>Corollary</u> : (i) The problems MEMBER $[E^{n+1}, \mathcal{E}^n]$ $(n \geq 3)$ and MEMBER $[R^{n+1}; \mathcal{R}^n]$ are Σ_2-complete.

(ii) The problem MEMBER $[A^{2n+8}; A^n]$ is Σ_2-complete, for all n.

As a consequence, determining the place of a set in one of the three infinite hierarchies $\{\mathcal{E}^n\}$, $\{\mathcal{R}^n\}$ or $\{A^n\}$ is in each case a strictly Δ_3 problem ; this strengthens previous results of Meyer-Ritchie [12] and extends Rosenberg's on one-way finite automata [14].

Set composition which has been defined on sets of integers can clearly be extended to languages on some finite alphabet by considering words as n-ary representations of integers ; CS $[\Sigma]$ is effectively closed under set composition ; furthermore the finiteness problem for CS $[\Sigma]$ is known to be Σ_2-complete ; hence :

<u>Corollary</u> (Cudia [4]) : Problems of determining whether a context-sensitive language is regular or context-free are Σ_2-complete.

As seen above, undecidability results are derived through successive applications of theorems 3 and 4. These theorems can be combined together giving :

<u>Proposition</u> : Let $\mathcal{B}$ and $\mathcal{C}$ be to recursive classes of sets such that $\mathcal{B} \subset \mathcal{C}$. Assume :

(a) $\mathcal{B}$ contains all finite sets

(b) $\mathcal{C}$ is closed under boolean operations, explicit transformations and bounded quantification.

(c) $\mathcal{C}$ contains an enumeration for $\mathcal{B}$ and a threshold relation of the kind $\lambda uv.\ \dot{u} > 2v$

(d) $\mathcal{C}$ with enumeration C is effectively closed under set composition.

Then FINITE $[\mathcal{C}]$ reduces to MEMBER $[\mathcal{C}; \mathcal{B}]$.

<u>Proof</u> : Conditions *(b)* and *(c)* imply the existence in $\mathcal{C}$ of a $\mathcal{B}$-immune set by theorem 3 ; this fact together with *(a)* and *(d)* yields the result by theorem 4.□

These abstract conditions compare to a similar and independent result of Lewis [10].

It also shows that if $\mathcal{C}$ is a basis for r.e. sets then MEMBER $[\mathcal{C}; \mathcal{B}]$ is Σ_2-complete. Furthermore, this property is hereditary downwards, that is, under the above hypotheses, MEMBER $[\mathcal{C}; \mathcal{B}']$ is Σ_2-complete for all recursive classes $\mathcal{B}'$ such that $\mathcal{F} \subset \mathcal{B}' \subset \mathcal{B}$.

We now turn to properties of program compactness ; in [2] Blum shows that descriptions of primitive recursive programs can be made arbitrarily more compact using "while statement". We strengthen this result and show that immune sets together with undecidability results enable us to derive properties of program compactness in subrecursive languages of different powers ; we obtain that the program compactification already holds on finite sets.

Using the standard notion of size measure as axiomatically introduced in [2] we have :

<u>Theorem 5</u> : Let $\mathcal{C}$ and $\mathcal{B} \subset \mathcal{C}$ be two recursive classes. Assume that conditions of theorem 4 hold for $\mathcal{C}$ and $\mathcal{B}$, that FINITE $[\mathcal{C}]$ is Σ-Π and let | | and | | be size measu-

res for C and B. Then C allows arbitrary program compactification for B on finite sets ; i.e :

$$\forall f \text{ rec. } \exists C_i \text{ finite } \forall j[B_j=C_i \rightarrow |j|_B > f(|i|_C)]$$

Proof : Assume to the contrary that an f exists such that :

$$\forall C_i[C_i \text{ finite} \rightarrow \exists j[|j|_B \leq f(|i|_C) \wedge B_j = C_i]]$$

Using notations of theorem 4 we would have C_i finite iff $C_{\tau(d,i)}$ finite iff $\exists j[|j|_B \leq f(|\tau(d,i)|_C) \wedge B_j = C_{\tau(d,i)}]$

Hence FINITE [C] would not be Σ_2-Π_2.□

This result can be reshaped to yield the following result of Meyer [11] established by different techniques

Proposition : For every subrecursive language B, one can construct a language C primitive recursive in B such that C allows arbitrary program compactification for B on finite sets.

References :

[1] Blum M. *"A machine-independent theory of the complexity of recursive functions"*, JACM 14 (1967),322-336.

[2] Blum M. *"On the size of machines"*, Inf. & Cont. 11 (1967),257-265.

[3] Constable R. *"Hierarchy theorems for axiomatic complexity"*, Computational complexity (Randall Rustin ed.) Algorithmics Press Inc. (1973),37-63.

[4] Cudia D. *"The degree hierarchy of undecidable problems of formal grammars"*, Proc. of 2nd ACM Symp. on theory of computing(1970) 10-21.

[5] Flajolet P., Steyaert J.M., *"Une formalisation de la notion d'algorithme de tri non récurrent"*, Thèse de 3° cycle. Université PARIS VII (1973).

[6] Flajolet P., Steyaert J.M., *"Decision problems for multihead finite automata"* Proc. of MFCS Symp. (1973), 225-230.

[7] Flajolet P., Steyaert J.M., *"Une généralisation de la notion d'ensemble immune"*, RAIRO R1 (1974), 37-48.

[8] Hartmanis J., Hopcroft J., *"An overview of the theory of computational complexity"*, JACM 18 (1971), 444-475.

[9] Hennie F., Stearns R., *"Two-tape simulation of multitape Turing machines"*, JACM 13 (1966), 533-546.

[10] Lewis F., *"On unsolvability in subrecursive classes of predicates"*, Harvard University report (1972).

[11] Meyer A, *"Program size in restricted programming languages"*, Inf. & Cont. 21 (1972) 382-394.

[12] Meyer A., Ritchie D., *"Computational complexity and program structure"*, I.B.M. research report RC 1817 (1967).

[13] Rogers H., *Theory of recursive functions and effective computability*, Mc Graw Hill (1966).

[14] Rosenberg A., *"On multihead finite automata"*, I.B.M. Journal (1966), 388-394.