REVUE FRANÇAISE D'AUTOMATIQUE INFORMATIQUE
RECHERCHE OPÉRATIONNELLE. INFORMATIQUE THÉORIQUE

G. LONGO
M. VENTURINI ZILLI

Complexity of theorem-proving procedures : some general properties

Revue française d'automatique informatique recherche opérationnelle. Informatique théorique, tome 8, n° R3 (1974), p. 5-18

<http://www.numdam.org/item?id=ITA_1974__8_3_5_0>

R.A.I.R.O.
(8e année, R-3, 1974, p. 5 à 18)

COMPLEXITY OF THEOREM-PROVING PROCEDURES (1): SOME GENERAL PROPERTIES

par G. LONGO (*) et M. VENTURINI ZILLI (**)

Communicated by G. AUSIELLO

Abstract. – *Theorem-proving procedures are considered as a family of partial recursive functions satisfying some axioms which express the properties of soundness and of decidability for a solvable class of formulas including all propositions, i.e. ground sentences, and an infinite class of non ground sentences. By using whatever complexity measure satisfying Blum's axioms it is proved that such a family of functions is π_2-complete; contains « difficult » proof-procedures; is such that every its elements has some programs having the « oscillation property »; does not contain a « best » proof-procedure. Moreover a notion of approximation is defined according to which the set of all theorems of a first-order theory is approximable.*

I. INTRODUCTION

In many existing papers complexity-comparisons between some theorem-proving procedures can be found. But what it is meant by a theorem-proving procedure? Usually it is meant a family of programs for computing a 0-1 partial recursive function taking value « 1 » only on theorems of a first-order theory. Hence comparisons concern families of programs, families which sometimes contain just one program. Moreover several criteria are used for the evaluation of complexity : most of them are based on an arbitrary set of theorems used as a test set; some suit particular procedures (as *r-m-size* in [6]). It is worthnoting that usually the chosen criterium is not applied on what has been generated by a theorem-proving procedure, but

(*) ISI, Università di Pisa.
(**) I.A.C.-C.N.R., Roma.
(1) Work carried out at IAC within the project of mathematical informatics.

on proofs « cleaned » in some way, as it has been noted in [8] where a distinction is proposed between complexity and difficulty.

The variety of approaches and criteria seem to require a sufficiently general and adequate approach. What we think useful to this aim is to have a formal notion of theorem-proving procedure and to fixe for it a reasonable class of complexity measures, where « complexity » has the intended meaning of « difficulty » (as in [8]).

In [8] various trade-off phenomena concerning efficiency of resolution strategies have been examined : the existence of unavoidable oscillating behaviours clearly turns out and some notions of best proof-procedures are proposed and conjectured to be realistic.

The axiomatic approach proposed in [4] is, from a certain point of view, very general : in fact it just requires that the Turing machines representing the proof-procedures stop on theorems only; on the other hand the results in [4] concern Herbrand proof-procedures.

2. AXIOMS

If we think of programs for Herbrand or resolution proof-procedures (from which most of the known theorem-proving procedures derive), we see that they « stop » also on some (infinitely many) closed formulas which are non-theorems and which include all false propositions (i.e. false ground sentences).

Hence we claim that properties that axioms for theorem-proving procedures should capture are :

1) Soundness.

2) Convergence at least on the set of all propositions (P_0).

3) To be a decision procedure for some class of formulas which, besides P_0, contains infinitely many theorems and infinitely many non-theorems. Such an r.e. set of formulas will be called solvable.

Then given an enumeration of all closed well-formed formulas of first-order predicate calculus (P_1), let $x, y, z, \ldots$ vary on such an enumerated set and T be the set of all code numbers for theorems of P_1.

Let $\mathcal{S}$ be the set of partial recursive functions p taking values 0 and 1 such that :

A_1) $p(x) = 1 \Rightarrow x \in T$, $p(x) = 0 \Rightarrow x \in CT$

A_2) $D_p = \{ x \mid p(x) = 0 \} \supset P_{00}$ and $T_p = \{ x \mid p(x) = 1 \} \supset P_{01}$

A) $D_p - P_{00}$ and $T_p - P_{01}$ are both infinite

where P_{01} is the set of tautologies of P_0 and $P_{00} = P_0 - P_{01}$; $D_p \cup T_p$ is what we call solvable class.

From now on we consider $\mathfrak{F}$ as the set of functions computed by theorem-proving programs : then by a proof-procedure we mean each function of $\mathfrak{F}$.

It is worthnoting that convergence on non-theorems is a relevant property of a proof-procedure for at least two reasons : *i*) a sound negative answer on a non-theorem is an information which is not less important or less useful than a sound positive answer on a theorem; *ii*) it is reasonable to suppose that there is a connection between the complexity for obtaining proofs and the complexity for recognizing non-theorems. In fact, concerning Herbrand and resolution proof-procedures, it can be shown that if a program of theirs accepts a theorem with a certain complexity then a non-theorem strictly related to the accepted theorem is rejected with less than or nearly the same complexity. Let us call *oscillation property* such a property. (On the other hand if a program of either Herbrand or resolution proof procedure rejects a non-theorem with a certain complexity then it is easy to obtain a theorem from that non-theorem accepted with a complexity not exceeding the complexity of the non-theorem. This depends on the fact that all those programs reject a non-theorem after having exhausted all the work they can do on that formula.)

3. OSCILLATION PROPERTY FOR HERBRAND AND RESOLUTION PROOF-PROCEDURES

Suppose that a program of a resolution strategy succeeded in proving $y \in T$. Let S be the uncodified set of clauses corresponding to y. Consider the refutation tree generated by the program applied to y. Construct a non-theorem x in the following way :

STEP1. Starting from the bottom rename only one of the two leterals giving $\square$ by a predicate symbol not occurring in S and generalize it by putting a variable as one of its arguments if it has no variable.

STEP2. Go one level up and make ground by instantiation the complementary pair of literals resolved upon.

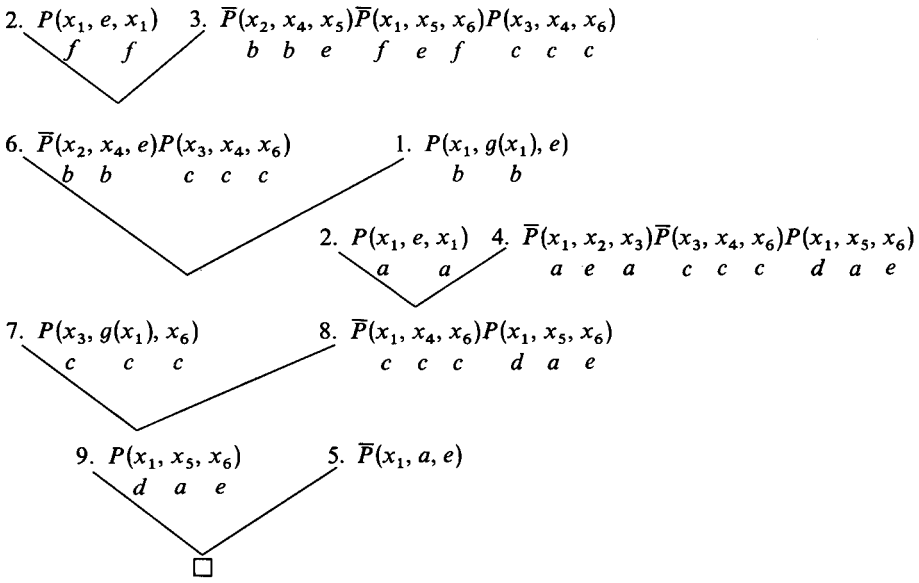
STEP3. Keep the renaming already done, use constants not already used and go to step 2.

STEP4. Let x be the code number for the set of clauses obtained taking the leaves of the tree modified by the preceding steps.

END

EXAMPLE :

$$\sim S \begin{cases} 1.\ P(x_1, g(x_1), e) \\ 2.\ P(x_1, e, x_1) \\ 3.\ \overline{P}(x_2, x_4, x_5)\overline{P}(x_1, x_5, x_6)P(x_3, x_4, x_6) \\ 4.\ \overline{P}(x_1, x_2, x_3)\overline{P}(x_3, x_4, x_6)P(x_1, x_5, x_6) \\ 5.\ \overline{P}(x_1, a, e) \end{cases}$$



$$\sim S' \begin{cases} 1.\ P(b, b, e) \\ 2.\ P(a, e, a) \\ 3.\ \overline{P}(b, b, e)\overline{P}(f, e, f)P(c, c, c) \\ 4.\ \overline{P}(a, e, a)\overline{P}(c, c, c)P(d, a, e) \\ 5.\ \overline{P}_1(x_1, a, e) \\ 6.\ P(f, e, f) \end{cases}$$

Then the considered program applied to x cannot add more resolvents to those included in the modified tree and then it works on x in an exhaustive way without obtaining $\square$ and without increasing complexity, if, for instance, as a complexity measure we define the number of resolvents or computation time etc... Hence such a program verifies the oscillation property.

A similar fact also holds for programs of Herbrand proof procedures. In fact suppose that one of those programs succeded in proving $y \in T$, that

is it constructed a set of ground clauses containing a contradiction. Then obtain a non-theorem x in the following way :

STEP1. Consider the subset H'_S of the Herbrand universe H_S used by the program for obtaining a contradiction and make ground in S all terms containing functions by substituting variables inside them with some constants of H'_S and put constants of H'_S in place of constants of $H_S - H'_S$.

STEP2. Rename by a predicate symbol not occurring in S only one of the two literals of the contradictory pair and generalize it substituting one of its arguments with a variable.

STEP3. Let x be the code number of the set of clauses of S modified by the preceding steps.

END

In this case it is easy to see that there exists an increasing total recursive function f such that the considered program applied on x works in an exhaustive way with complexity f-bounded by the complexity needed for proving y. Hence we may say that such a program verifies the f-oscillation property.

EXAMPLE :

$$\sim S \begin{cases} P(a, x) \\ \overline{P}(y, f(g(z))) \\ \overline{Q}(b)R(x) \end{cases}$$

$$H_S = \{ a, b, f(g(a)), f(g(b)), \ldots \}$$

$$\begin{array}{lll}
P(a,a) \wedge \overline{P}(a, f(g(a))) \wedge \overline{Q}(b)R(a) \wedge & \text{with} & a/x, a/y, a/z \\
\wedge\ P(a,a) \wedge \overline{P}(a, f(g(b))) \wedge \overline{Q}(b)R(a) \wedge & \text{with} & a/x, a/y, b/z \\
\wedge\ P(a,a) \wedge \overline{P}(b, f(g(b))) \wedge \overline{Q}(b)R(a) \wedge & \text{with} & a/x, b/y, b/z \\
\wedge\ P(a,b) \wedge \overline{P}(a, f(g(a))) \wedge \overline{Q}(b)R(b) \wedge & \text{with} & b/x, a/y, a/z \\
\wedge\ P(a,b) \wedge \overline{P}(b, f(g(a))) \wedge \overline{Q}(b)R(b) \wedge & \text{with} & b/x, b/y, a/z \\
\wedge\ P(a,b) \wedge \overline{P}(b, f(g(b))) \wedge \overline{Q}(b)R(b) \wedge & \text{with} & b/x, b/y, b/z \\
\wedge\ P(a, f(g(a))) \wedge \overline{P}(a, f(g(a))) \wedge \overline{Q}(b)R(f(g(a))) & \text{with} & f(g(a))/x, a/y, a/z
\end{array}$$

$$H'_S = \{ a, b, f(g(a)) \}$$

$$\sim S' \begin{cases} P_1(a, x) \\ \overline{P}(y, f(g(a))) \\ \overline{Q}(b)R(x) \end{cases}$$

$$H_{S'} = \{ a, b, f(g(a)) \}$$

The informal notion of complexity we used so far mainly refers to the total number either of resolvents or of instances obtained from a set of clauses, that is, as we already mentioned, it refers to a difficulty measure as time or the total number of steps.

In the sequel, as a complexity measure we assume any measure satisfying Blum's axioms [2], and if $\{ \varphi_i \}_{i \in N}$ is an acceptable Gödel numbering of all partial recursive functions, $\{ \Phi_i \}_{i \in N}$ will be the corresponding set of measure functions (1).

We now remark that for any complexity measure and for any function $p \in \mathcal{F}$, there are some of its programs such that for every y there exists $x \in D_p - P_{00}$, $x \geqslant y$ and x is rejected with a complexity not exceeding the complexity for accepting y (i.e. those programs have the oscillation property).

In fact let's first suppose that we are given an enumeration of all closed well formed formulas of predicate calculus; then, given any $p \in \mathcal{F}$, let T_p be the range of t, $D_p - P_{00}$ be the range of d and P_{00} the range of σ, with t, d and σ one-one total recursive functions. Suppose also that $T_p \cup D_p$ has been enumerated in the following way : compute $d(0)$, $d(1)$..., put $t(0)$ in place $2^{t(0)}$ and $\sigma(0)$ in $3^{t(0)}$; continue to compute $d(m)$, $d(m+1)$..., put $t(n)$ in place $2^{t(0)+\ldots+t(n)}$ and $\sigma(n)$ in place $3^{t(0)+\ldots+t(n)}$.

More formally $T_p \cup D_p = \{ \xi_1, \xi_2, \ldots \}$ with $t(x) = \xi_j$ iff $j = 2^{t(0)+\ldots+t(x)}$ and $\sigma(x) = \xi_k$ iff $k = 3^{t(0)+\ldots+t(x)}$. Given such a listing of $T_p \cup D_p$ a program φ_i for computing p is the following : let $y \in N$

STEP1. Enumerate $T_p \cup D_p$ as before : if $\exists j$ s.t. $y = \xi_j$ go to step 2 with $n = 0$, diverge otherwise.

STEP2. If $j = 2^{t(0)+\ldots+t(n)}$ give output 1 ; if $j < 2^{t(0)+\ldots+t(n)}$ give output 0 ; if $j > 2^{t(0)+\ldots+t(n)}$ go to step 3.

STEP3. Increase n by 1 and go to step 2.

END.

This program works like enumeration theorem proving procedures : but, instead of enumerating only theorems, it can also recognize, by the same technique, a class of non-theorems. Moreover (almost) every $y \in T$, $y = \xi_j$ is « preceded » by some non ground $x \in CT$, $x = \xi_h$, i.e. $h < j$, with $x > y$: in fact, since $2^{t(0)+,,,+t(n)} - 2n > t(n)$ (almost) always and, for $y = t(n)$ (i.e. $j = 2^{t(0)+,,,+t(n)}$) and $k < j$, there are at most $2n$ $\xi_k \in T \cup P_{00}$, then there exist some $\xi_h = x \in D_p - P_{00}$ s.t. $h < j$ and $x = d(m) > t(n) = y$.

Therefore the « amount of work » (the complexity) needed for computing $\varphi_i(y)$ includes the amount of work needed for computing $\varphi_i(x)$.

(1) Blum's axioms : 1) $\varphi_i(x) \downarrow \Leftrightarrow \Phi_i(x) \downarrow$
2) $\forall i, x, y$ $\Phi_i(x) = y$ is decidable.

4. UNSOLVABILITY OF $\mathfrak{F}$

In this paragraph we briefly present a few facts which tell us where the set $\Omega\mathfrak{F}$ (of all indices for functions in $\mathfrak{F}$) is located in the Kleene-Mostowski hierarchy (i.e. its degree of unsolvability) and that $\Omega\mathfrak{F}$ can be neither recursively enumerated nor recursively « represented ».

Lemma 1. $\forall x \exists p \in \mathfrak{F}$ such that $p(x)\downarrow$.

Proof. Let $p_0 \in \mathfrak{F}$ and define

$$p_1(y) = \begin{cases} 0 & \text{if } y = x \\ p_0(y) & \text{otherwise,} \end{cases}$$

$$p_2(y) = \begin{cases} 1 & \text{if } y = x \\ p_0(y) & \text{otherwise} \end{cases}$$

then $p_1 \in \mathfrak{F}$ iff $x \notin T$, and $p_2 \in \mathfrak{F}$ iff $x \in T$.

Q.E.D.

Calling, as usual, a representation of a set A of functions every set $B \subset \Omega A$ such that $\forall f \in A\ \exists i \in B \cap \Omega\{f\}$, we have :

Teorem 2 : A) There is no r.e. representation of $\mathfrak{F}$.

B) $\Omega\mathfrak{F}$ is a Π_2-complete set.

Proof. A) Supposing the contrary, for all x we may compute $\varphi_{l_i}(x)$ by a dovetailing technique on $i = 1, 2, \ldots$ Then by lemma 1 we could décide P_1.

Proof. B) $\Omega\,\mathfrak{F} \in \Pi_2$ can be seen, using Tarski-Kuratowski algorithm, from

$$\begin{aligned} i \in \Omega\mathfrak{F} \text{ iff } \forall x((\varphi_i(x) = 1 \Rightarrow x \in T) &\wedge (\varphi_i(x) = 0 \Rightarrow x \in CT) \\ &\wedge (x \in P_{01} \Rightarrow x \in T_{\varphi_i}) \\ &\wedge (x \in P_{00} \Rightarrow x \in D_{\varphi_i}) \\ &\wedge \exists y(y > x \wedge y \in T_{\varphi_i} - P_{01}) \\ &\wedge \exists z(z > x \wedge z \in D_{\varphi_i} - P_{00}) \\ &\wedge (\varphi_i(x)\downarrow \Leftrightarrow \varphi_i(x) \in \{0, 1\})) \end{aligned}$$

and from $\{ i \mid W_i \text{ infinite} \} \leqslant_m \Omega\mathfrak{F}$.

Q.E.D.

5. DIFFICULT PROOF-PROCEDURES

Let R be the set of total recursive functions and $\{ \Phi_i \}_{i \in N}$ any fixed complexity measure.

Lemma 3. $\forall f \in R$, $\forall L$ infinite r.e. set, $L \neq N$, $\exists g$, $0-1$ valued partial recursive function, such that $G = \{ x \mid g(x) = 1 \} \subset L$, G infinite and

$$\forall i \in \Omega \{ g \} \overset{\infty}{\forall} x \Phi_i(x) > f(x).$$

Proof. Let l be the semicharacteristic function of L. Only one of the following cases holds :

i) $\forall i \in \Omega \{ l \} \overset{\infty}{\forall} x \Phi_i(x) > f(x)$ and then the lemma is true.

ii) $\exists k \in \Omega \{ l \} \overset{\infty}{\exists} x \Phi_k(x) \leqslant f(x)$. Then define

$$d_{kj}(x) = \begin{cases} 0 & \text{if } \Phi_k(x) \leqslant f(x) \\ j \dot{-} x & \text{otherwise.} \end{cases}$$

Obviously $\forall j \overset{\infty}{\forall} x \, d_{kj}(x) = 0$ and a result in [1] easily gives, by a padding technique, that $\exists s \in R$, with $s(k, j) \in \Omega \{ d_{kj} \}$ and $s(k, j) > j$ for all j, and $\exists r \in R$, with r depending effectively on k and $r(x, y) > y$, such that

$$\forall j \overset{\infty}{\forall} x (\Phi_k(x) \leqslant f(x) \Rightarrow \Phi_{S(k,j)}(x) \leqslant r(x, f(x))).$$

Therefore :

$$(3.1) \qquad \forall j \overset{\infty}{\exists} x \in L \text{ s.t. } \varphi_{s(k,j)}(x) = 0 \quad \text{and} \quad \Phi_{s(k\,j)}(x) \leqslant r(x, f(x)).$$

Moreover, since $L \neq N$,

$$(3.2) \qquad \exists z \exists i_0 \text{ s.t. } \forall i, j > i_0 \quad i \neq j \Rightarrow \varphi_{s(k,i)}(z) \neq \varphi_{s(k,j)}(z),$$

that is $\{ \varphi_{s(k,i)} \}_{i \in N}$ is a set of programs for infinitely many distinct functions.

Construct now (see also a known technique for a result in [10])

$$g(x) = \begin{cases} 1 \dot{-} \varphi_i(x) & \text{if } \exists i \leqslant x \text{ , } i \text{ « not underlined »,} \\ & \text{s.t. } \Phi_i(x) \leqslant r(x, f(x)) \wedge l(x) = 1 \\ \uparrow \text{ otherwise} \end{cases}$$

and when $g(x) = 1 \dot{-} \varphi_i(x)$ underline the corresponding i.

Suppose now that $u \in \Omega \{ g \}$; if $x < u$ it can be $\Phi_u(x) \leqslant r(x, f(x))$. When $x \geqslant u$ and $x \in L$, $\Phi_u(x) \leqslant r(x, f(x))$ holds only if an index i, $i < u \leqslant x$, has been underlined in the computation of $g(x)$, but this is possible only for u values of x.

Moreover, thanks to 3.1 and 3.2, each $s(k, j)$ is underlined, soon or later, during the construction of g, and almost all $s(k, j)$ are underlined in correspondence to an input x on which $\varphi_{s(k,j)}$ assumes value zero (remember that $s(k, i) > i$) : that is, since for any x at most one index is underlined and, if it is an $s(k, j)$ index, it is underlined only when $x \geqslant s(k, j) > j$ (i.e. $j \dot{-} x = 0$). Toghether with $r(x, f(x)) > f(x)$ those facts give our thesis.

Q.E.D.

Lemma 4. Let i and e be s.t. $\nexists x \varphi_i(x) = 0 = \varphi_e(x)$; then

$$\exists t \in R \, \exists v \in R \quad , \quad v(x, y, z)$$

increasing with respect to y and z, s.t. :

$$\forall j \, \varphi_{t(i,j,e)}(x) = \begin{cases} 1 & \text{if} \quad \varphi_i(x) = 0 \wedge \varphi_j(x) = 0 \\ 0 & \text{if} \quad \varphi_e(x) = 0 \\ \uparrow & \text{otherwise} \end{cases}$$

and $$\overset{\infty}{\forall} x(\varphi_{t(i,j,e)}(x) = 1 \Rightarrow \Phi_{t(i,j,e)}(x) \leqslant v(x, \Phi_i(x), \Phi_j(x)).$$

Proof. Define

$$p(i, j, e, x, y, z) = \begin{cases} \Phi_{t(i,j,e)}(x) & \text{if} \quad \Phi_i(x) = y \wedge \Phi_j(x) = z \wedge \varphi_i(x) = 0 \\ & \qquad\qquad \wedge \varphi_j(x) = 0 \\ 0 \text{ otherwise} & \end{cases}$$

and $v(x, y, z) = 1 + \max\limits_{i,j,e \geqslant x} p(i, j, e, x, y, z)$. Then $\overset{\infty}{\forall} x$

$$v(x, \Phi_i(x), \Phi_j(x)) \geqslant \Phi_{t(i,j,e)}(x) \quad \text{if} \quad \varphi_{t(i,j,e)}(x) = 1.$$

Q.E.D.

Next results in this paragraph show that for any solvable class there exists a proof-procedure having all programs requiring an arbitrary amount of « resource » for recognizing theorems and non-theorems in the solvable class.

Theorem 5. $\forall h \in R \, \forall M$ solvable class $\exists p \in \mathfrak{F}$ s.t. $T_p = T \cap M$ and $D_p \subset CT \cap M$ and $\forall i \in \Omega \{ p \}$

$$\overset{\infty}{\forall} x(x \in D_p - P_{00} \Rightarrow \Phi_i(x) > h(x)).$$

Proof

$$\text{Let } q(x) = \begin{cases} 1 & \text{if} \quad x \in P_{00} \\ 0 & \text{otherwise} \end{cases}$$

and $j_0 \in \Omega \{ q \}$.

Take $f(x) = v(x, h(x), \Phi_{j_0}(x))$, with v as in Lemma 4, $L = M \cap CT$ and use Lemma 3 : there exist an r.e. infinite set $G \subset L$ and a partial recursive function g s.t.

$$G = \{ x/g(x) = 1 \} \quad \text{and} \quad \forall i \in \Omega \{ g \} \overset{\infty}{\forall} x \, \Phi_i(x) > f(x).$$

Define

$$p(x) = \begin{cases} 1 & \text{if } x \in M \cap T \\ 0 & \text{if } x \in P_{00} \cup G \\ \uparrow & \text{otherwise} \end{cases}$$

then $p \in \mathfrak{F}$ and, if $\exists i_0 \in \Omega \{ p \}$ such that $\overset{\infty}{\exists} x \in D_p - P_{00}$ with $\Phi_{i_0}(x) \leqslant h(x)$, then, for $e_0 \in \Omega \{ g \}$ and t as in Lemma 4, $t(i_0, j_0, e_0) \in \Omega \{ g \}$ and

$$\overset{\infty}{\exists} x \Phi_{t(i_0, j_0, e_0)}(x) \leqslant v(x, \Phi_{i_0}(x), \Phi_{j_0}(x)) \leqslant v(x, h(x), \Phi_{j_0}(x)) = f(x)$$

against the construction of g.

Q.E.D.

Theorem 6. $\forall h \in R \; \forall M$ solvable class $\exists p \in \mathfrak{F}$ s.t. $T_p \subset T \cap M$ and $D_p \subset CT \cap M$ and

$$\forall i \in \Omega \{ p \} \overset{\infty}{\forall} x [(x \in (T_p \cup D_p) - P_0) \Rightarrow (\Phi_i(x) > h(x))].$$

Proof. As for theorem 5 using Lemma 3 and 4 also for $L' = T \cap M$.

Q.E.D.

Notice that theorem 5 although weaker from a complexity point of view concerns a complete proof procedure when $M \supset T$.

REMARK. At the end of paragraph 3 we showed that, for any $p \in \mathfrak{F}$, there exist some programs for p having the oscillation property.

More formally, since $f : N \to N$ increasing implies $f(x) \geqslant x$, we have shown that : $\forall f \in R$, f increasing, $\forall p \in \mathfrak{F} \; \exists j \in \Omega \{ p \}$ s.t. $\forall y \; \exists x \in D_p - P_{00}$, $x \geqslant y$ and $\Phi_j(x) \leqslant f(\Phi_j(y))$. For any increasing $f \in R$ let's call H_f the set of all programs in $\Omega\mathfrak{F}$ which have the f-oscillation property : then H_f is a representation of $\mathfrak{F}$.

What we said in paragraph 3 about the oscillation property for Herbrand and resolution programs (where, in the case of Herbrand programs, we proved an f-oscillation property for a particular increasing $f \in R$) allows us to point our interest on H_f and to consider the following corollary to theorem 5 :

Corollary 7. $\forall f \in R$, f increasing, $\forall s \in R \; \forall M$ solvable class, $\exists p \in \mathfrak{F}$ with $T_p = T \cap M$, $D_p \subset CT \cap M$ and $\forall i \in \Omega \{ p \} \cap H_f \overset{\infty}{\forall} x \Phi_i(x) > s(x)$.

Proof. Set $h(x) = \max_{z \leqslant x} f(s(z))$ and use theorem 5 :

$$\exists p \in \mathfrak{I},\quad D_p \subset P_{00} \cup L,\quad \text{s.t. } \forall i \in \Omega \{ p \} \overset{\infty}{\forall} x \in D_p - P_{00}\Phi_i(x) > h(x).$$

Let's now suppose that $\exists j \in \Omega \{ p \} \cap H_f$ s.t. $\overset{\infty}{\exists} y \Phi_j(y) \leqslant s(y)$. Then, by definition of H_f, $\overset{\infty}{\exists} x \in D_p - P_{00}$ s.t. $\Phi_j(x) \leqslant f(\Phi_j(y)) \leqslant f(s(y)) \leqslant h(x)$ against the definition of p.

Q.E.D.

This means, among other things, that it is not significant to rely on a finite test set of formulas for evaluating complexity of theorem-proving programs.

6. NON OPTIMALITY

Ehrenfeucht and Rabin have shown (unpublished) that there is no « perfect » Herbrand proof procedure : that is a procedure which stops on $x \in T$ after having generated only a minimally contradictory set of instances of x. In [3] it is shown, using the preceding result, that there is no *best* Herbrand proof procedure, where best is called a procedure which terminates, $\forall x \in T$, after having generated a number of instances of x which is less than or equal to the number gererated by any other Herbrand proof procedure and, moreover, $\exists x \in T$ on which this number is strictly less than the number generated by any other Herbrand proof procedure.

Since the approach proposed in this paper notes the importance of negative answers a proof procedure may give, and the notion of irrelevancy is not reasonably definable for non-theorems, the above results cannot be extended to be compatible with our approach.

Then defining :

1) $p \in \mathfrak{I}$ is *stronger* than $p' \in \mathfrak{I}$ iff $D'_p \subset D_p$ and $T'_p \subset T_p$.

2) $p_0 \in \mathfrak{I}$ is *strongest* iff there is no stronger $p \in \mathfrak{I}$, we have :

Theorem 8. There is no strongest proof procedure.

Proof. Let $p_0 \in \mathfrak{I}$ be the strongest, then by Lemma 1 a stronger $p \in \mathfrak{I}$ can be defined.

Q.E.D

3) Let $f \in R$, $i \in \Omega\mathfrak{I}$ is no *worse by* f than $j \in \Omega\mathfrak{I}$ iff

$$\forall x(\varphi_j(x)\downarrow \Rightarrow \Phi_i(x) \leqslant f(x, \Phi_j(x))).$$

4) $p \in \mathfrak{P}$ is *better* than $p' \in \mathfrak{P}$ iff $\exists f \in R\ \forall j \in \Omega\ \{ p' \}$
$\exists i \in \Omega\ \{ p \}$ i is no worse by f than j.

5) p_0 is *best* iff there is no better $p \in \mathfrak{P}$.

Corollary 9. There is no best proof-procedure.

Proof. Since the best should be the strongest.
Q.E.D.

7. APPROXIMATION

On a computer a program for theorem-proving procedures runs within a bounded amount of « resource » (time, space, etc...) : then, given a program $i \in \Omega\mathfrak{P}$ and a total measure function (clock, ...) $\Phi_j \in \{ \Phi_i \}_{i \in N}$, what is actually computed may be written :

$$\varphi_{s(i,j)}(x) = \begin{cases} \varphi_i(x) & \text{if } \Phi_i(x) \leqslant \Phi_j(x) \\ \$ & \text{otherwise} \end{cases}$$

and the following recursive sets are defined :

$$A_{ij} = \{ x / \varphi_{s(i,j)}(x) = 1 \}$$
$$B_{ij} = \{ x / \varphi_{s(i,j)}(x) = 0 \}.$$

What one would expect is, informally, to approximate, in some way, the set T of theorems by the above recursive sets. This problem may be so formulated :

Def (Meyer, Lynch). Density of a r.e. set A is

$$dens\ (A) = \lim \frac{1}{n} |\{ 0, ..., n-1 \} \cap A| \text{ if it exists.}$$

Def Given $i \in \Omega\mathfrak{P}$ and $\Phi_j \in \{ \Phi_i \} \cap R$, the set $T_{\mathfrak{L}}$ of codified theorems of a first-order theory $\mathfrak{L}$ is *ij-approximable* iff dens $(A_{ij} \cup B_{ij}) = 1$.

Now, we need to refer to the theory $\mathfrak{L}$, because, as it is well known [11], the r.e. sets of theorems may stay on different m-degrees including the complete m-degree while excluding the m-degree of simple sets. Moreover there exist [7] creative sets which are not approximable according to the following definition :

Def (Meyer, Lynch). A r.e. set E is approximable iff $\exists A$, $\exists B$ recursive sets s.t. $A \subset E$, $B \subset CE$ and dens $(A \cup B) = 1$.

Let's first remark that a set is approximable iff it is *ij*-approximable for some i and j ($\Leftarrow$ obvious; $\Rightarrow$ take a function which is always the maximum

of the complexity of the characteristic functions of the recursive approximating sets : this will be the desired clock...).

Hence our problem is whether, for a given $\mathfrak{L}$, $T_{\mathfrak{L}}$ is approximable or not.

Def $A \equiv B$ iff $\exists\sigma$ recursive permutation s.t. $A = \sigma(B)$.

Lemma 10. $\forall r, 0 < r = \frac{m}{n} \leqslant 1\ \forall E$ r.e. not co-isolated (i.e. neither co-finite nor simple) $\exists D$, $D \equiv E$, s.t. dens $(D) = r$ and D is approximable.

Proof. Let E be r.e., $f \in R$ increasing s.t. $\lim \frac{n}{f(n)} = 0$ and $0 < m \leqslant n$.

Let Q be the set of the first m elements of any interval $[kn, (k+1)n[$, with $k \in N$: dens $(Q) = r = \frac{m}{n}$ and Q is recursive.

Now if $D = (Q - \{ f(x)/x \in N \}) \cup \{ f(x)/x \in E \}$ then D is r.e. and dens (D) = dens $(Q) = r$, because dens $(\{ f(x)/x \in N \}) = 0$.

Defining $A = Q - \{ f(x)/x \in N \}$ and $B = CQ - \{ f(x)/x \in N \}$, we see that A is reçursive, $A \subset D$ and dens $(A) = r$, while B is recursive,

$$B = CQ \cap C\{ f(x)/x \in N \} = C(Q \cup \{ f(x)/x \in N \}) \subset C(Q \cup \{ f(x)/x \in E \}) \subset CD$$

and dens $(B) = 1 - r$.

Hence dens $(A \cup B) = 1$.

Moreover :

i) $E \leqslant_1 D$ via f : in fact if $x \in E$ then $f(x) \in D$ and if $x \notin E$ then $f(x) \notin D$.

ii) $D \leqslant_1 E$ via a function g so defined :

$$g(y) = \begin{cases} x & \text{if } \exists x\, f(x) = y \wedge x \notin W_{i'} \wedge x \notin W_i \\ \varphi_{h'}(y) & \text{if } y \in Q - \{ f(x)/x \in N \} \vee (\exists x\, f(x) = y \wedge x \in W_{i'}) \\ \varphi_h(y) & \text{otherwise} \end{cases}$$

Where $W_i \subset CE$ and $W_{i'} \subset E$ are infinite recursive sets (E is not co-isolated) enumerated in an increasing order by respetively φ_h and $\varphi_{h'}$. Since $g \in R$ and $y \neq z \Rightarrow g(y) \neq g(z)$ $D \equiv_1 E$, that is [11] there exists (effectively) a recursive permutation σ s.t. $D = \sigma(E)$.

Q.E.D.

Theorem 11. $\forall \mathfrak{L}$ first-order logic $\forall r\ 0 < r = \frac{m}{n} \leqslant 1\ \exists\tau$, effective one-one enumeration of closed well formed formulas of $\mathfrak{L}$ s.t. dens $(T_{\mathfrak{L}}^{\tau}) = r$ and $T_{\mathfrak{L}}^{\tau}$ is approximable