The Intensional Content of Rice's Theorem *(Pearl)*

Andrea Asperti

Department of Computer Science, University of Bologna
Mura Anteo Zamboni 7, 40127, Bologna, ITALY
asperti@cs.unibo.it

Abstract

The proofs of major results of Computability Theory like Rice, Rice-Shapiro or Kleene's fixed point theorem hide more information of what is usually expressed in their respective statements. We make this information explicit, allowing to state stronger, complexity theoretic-versions of all these theorems. In particular, we replace the notion of extensional set of indices of programs, by a set of indices of programs having not only the same extensional behavior but also similar complexity (Complexity Clique). We prove, under very weak complexity assumptions, that any recursive Complexity Clique is trivial, and any r.e. Complexity Clique is an extensional set (and thus satisfies Rice-Shapiro conditions). This allows, for instance, to use Rice's argument to prove that the property of having polynomial complexity is not decidable, and to use Rice-Shapiro to conclude that it is not even semi-decidable. We conclude the paper with a discussion of "complexity-theoretic" versions of Kleene's Fixed Point Theorem.

Categories and Subject Descriptors F.4.1 [*Computability Theory*]; F.1.3 [*Machine-independent complexity*]

General Terms Theory

Keywords Recursion Theory, computability

1. Introduction

Recursive properties of *extensional* sets, i.e. index sets for partial recursive functions, have been extensively studied since the early days of Recursion Theory (Rice 1953; Dekker and Myhill 1958). Among *intensional* properties of programs, a major role is played by *complexity*. The theoretical research on effective properties of recursive functions under complexity assumptions focused, since the very beginning, on *complexity classes*, that is classes of recursive functions computable (almost everywhere) within a given bound of complexity t. Studies on complexity classes, mostly developed in the abstract framework promoted by Blum (Blum 1967), extensively investigated their order structure under set theoretic inclusion (Borodin 1972; McCreight and Meyer 1969), their recursive presentability and the computational quality of such a presentation (Lewis 1970; Young 1969; Landweber and Robertson 1972). It is important to understand that a complexity class is a set of functions, and not of indices; regarded as a set of indices, it is thus extensional by definition. If we are really interested in decidability properties of program complexity, complexity classes do not look like as an adequate instrument to perform the investigation.

The notion we propose here is that of *complexity clique*. A complexity clique (Section 2) is a set of indices for recursive functions closed w.r.t. indices of functions with same extension and similar complexity (defined in the only sensible way, namely up to constants). A *complexity clique* is not extensional, in general. On the other side, any extensional set is a *complexity clique* (the trivial "complexity" clique with arbitrary complexity).

Set-theoretic (Section 3) and, especially, recursive properties of complexity cliques are investigated. In particular, we generalize both Rice's Theorem (Section 4) and Rice-Shapiro's Theorem (Section 5), concluding with a discussion of Kleene's second fixed point Theorem from a complexity perspective (Section 6). Remarkably, all proofs remain essentially the same of their classical counterparts: in other words, all those proofs seem to hide more information of what is expressed in their classical statements, and the notion of Complexity Clique is precisely what is required to make this hidden information explicit (providing our original motivation for their introduction).

Following an old tradition, we try to stick to the appealing generality of Blum's axiomatic approach; however, axiomatization is not the focus of the paper, and we shall introduce new axioms with some liberality, as far as they are satisfied by standard complexity measures; in particular, we shall introduce additional axioms bounding the complexity of (linear) composition, of the s-m-n function, and finally of the universal machine.

Our generalization of Rice's Theorem and other enumeration techniques is also different from (and in some sense complementary to) the approach in (Kozen 1980): while Kozen investigates extensional properties in a sub-recursive setting, we are essentially concerned with decidability of complexity questions in a general recursive setting.

2. Similarity and Complexity Cliques

We shall work with n-ary partial recursive functions over natural numbers. If f is such a partial function, $dom(f)$ and $cod(f)$ respectively denote the domain and range of f. We write $f(n)\downarrow$ if $n \in dom(f)$. We use $\perp$ for the everywhere divergent function, that is $dom(\perp) = \emptyset$. Given two function f and g, we write $f \cong g$ when $f$and g have the same graph, i.e. $dom(f) = dom(g)$ and for any $n \in dom(f)$, $f(n) = g(n)$. We say a function is *finite* if it has a finite graph (i.e. $dom(f)$ is a finite set). Partial functions are or-

POPL '08 10-12 January 2008, San Francisco (Ca)

dered w.r.t the set-theoretic inclusion of their graphs: $f \leq g$ if and only if $graph(f) \subseteq graph(g)$. ω is the set of all natural numbers.

DEFINITION 1. *Let f and g be partial functions on natural numbers:*

1. *we say $f \in O(g)$ if and only if there exist n and c such that for any $m \geq n$, if $g(m) \downarrow$ then $f(m) \leq cg(m)$;*
2. *$f \in \Theta(g)$ if and only if $f \in O(g)$ and $g \in O(f)$.*

Note that if $f \in \Theta(g)$ the domains of f and g may only differ for a finite number of points. Let us also remark that, $f \in \Theta(g)$ if and only if $g \in \Theta(f)$; in particular, $f \in \Theta(g)$ is an equivalence relation.

DEFINITION 2. *(Blum 1967) A pair $\langle \phi, \Phi \rangle$ is a* computational complexity measure *if ϕ is a principal effective enumeration of partial recursive functions and Φ satisfies Blum's axioms:*

$$(a) \quad \phi_i(\vec{n}) \downarrow \leftrightarrow \Phi_i(\vec{n}) \downarrow$$
$$(b) \quad \text{the predicate } \Phi_i(\vec{n}) = m \text{ is decidable}$$

We say two programs are *similar* when they compute the same function and have the same complexity, up to constants:

DEFINITION 3. *Two programs i and j are* similar *(write $i \approx j$) if and only if*

$$\phi_j \cong \phi_i \wedge \Phi_j \in \Theta(\Phi_i)$$

THEOREM 4. *Similarity is an equivalence relation.*

Proof. Obvious, since similarity is defined as intersection of two equivalence relations.

We shall write $[i]_{\approx}$ for the equivalence class of i w.r.t. the similarity relation $\approx$.

DEFINITION 5. *Let $\langle \phi, \Phi \rangle$ be an abstract complexity measure. A set P of natural numbers is a* Complexity Clique*, if and only if for all i and j*

$$i \in P \wedge j \approx i \rightarrow j \in P$$

EXAMPLE 6. *The following are examples of Complexity Cliques:*

1. *$\emptyset$ and ω;*
2. *for any index i, $[i]_{\approx}$;*
3. *for any index i, $\{j | \Phi_j \in O(\Phi_i)\}$.*

On the other side the set $\{i | \Phi_i(a) = b\}$ (i.e. the programs whose complexity on input a is b) is not a Complexity Clique, in general.

3. Simple properties of Complexity Cliques

A set $A \subset \omega$ of indices is *extensional* if, for all i and j, $i \in A \wedge \phi_j \cong \phi_i \rightarrow j \in A$; this is equivalent to say that A is the counter-image (index set) under ϕ of some subset of partial recursive functions.

The notion of Complexity Clique is a strict generalization of the notion of *extensional set* well known in computability theory:

THEOREM 7. *Every extensional set is a Complexity Clique.*

Proof. Trivial.

Of course, the converse is not true: for instance, the class of programs with polynomial complexity is not *extensional*.

Any Complexity Cliques is a union of similarity classes, in particular:

THEOREM 8. *C is a Complexity Clique if and only if*

$$C = \bigcup_{i \in C} [i]_{\approx}$$

Proof. Trivial.

As a consequence, Complexity Cliques have very good algebraic properties; in particular:

THEOREM 9. *Complexity Cliques, equipped with the subset relation, are a Complete Boolean Lattice.*

Proof. Just note that $\overline{C} = \bigcup_{i \in \overline{C}} [i]_{\approx}$. The rest is easy.

The good set-theoretic properties of Complexity Cliques are to be compared with the very bad properties of Complexity Classes (stressing again the strong difference between the two notions). In particular:

1. for any complexity measure, Complexity Classes are not closed under union (Hartmanis and Stearns 1965; McCreight and Meyer 1969);
2. there are complexity measures whose Complexity Classes are not closed under intersection[1](Landweber and Robertson 1972)

Let us finally remark an important property about finite functions.

THEOREM 10. *Let C be a Complexity Class and let $i \in C$. If ϕ_i is* finite *then for any j such that $\phi_i \cong \phi_j$, $j \in C$.*

*Proof.*For finite functions, extensional equivalence implies similarity.

4. Generalized Rice's Theorem

DEFINITION 11. *A pair $\langle \phi, \Phi \rangle$ has the* s-m-n property *if for all positive natural numbers m and n there exists a recursive function s_m^n such that, for any i and all $x_1, \ldots, x_m$*

$$(a) \quad \phi_{s_m^n(i,x_1,\ldots,x_m)} \cong \lambda y_1, \ldots, y_n . \phi_i(x_1, \ldots, x_m, y_1, \ldots, y_n)$$
$$(b) \quad \Phi_{s_m^n(i,x_1,\ldots,x_m)} \in \Theta(\lambda y_1, \ldots, y_n . \Phi_i(x_1, \ldots, x_m, y_1, \ldots, y_n))$$

Equation (a) is the standard s-m-n theorem, while equation (b) states that the overhead introduced by the function s_m^n is at most a constant factor: in fact, in standard computational models, it amounts to the cost of copying the (fixed) parameters $x_1, \ldots, x_m$ and then calling ϕ_i - see e.g. Cutland (1986); Börger (1986); Odifreddi (1997).

Since the set or recursive functions is closed under composition, by the s-m-n theorem we may conclude that there exists a total computable function h such that $\phi_{h(i,j)} \cong \phi_i \circ \phi_j$. Axioms relating the complexity of $h(i,j)$ to the complexity of its components have been considered by Lischke (1975, 1976, 1977). For the purposes of our analysis, we need a tight bound:

DEFINITION 12. *A pair $\langle \phi, \Phi \rangle$ has the (linear) time-composition property if there exists a total computable function h such that*

$$(a) \quad \phi_{h(i,j)} \cong \phi_i \circ \phi_j$$
$$(b) \quad \Phi_{h(i,j)} \in \Theta(\Phi_i \circ \phi_j + \Phi_j)$$

We say that it has the (linear) space-composition property if equation (b) is replaced by

$$(c) \quad \Phi_{h(i,j)} \in \Theta(max\{\Phi_i \circ \phi_j, \Phi_j\})$$

Equation (b) says that the cost for computing $\phi_i(\phi_j(x))$ is, up to a constant factor, no more than the cost for computing $\phi_j(x)$ plus the cost for computing ϕ_i on input $\phi_j(x)$. Similarly, the space required for the same computation is the maximum among the

[1] An important exception are measures satisfying the parallel computation property (Landweber and Robertson 1972) (see also section 5).

space required for computing $\phi_j(x)$ and the space required for computing ϕ_i on input $\phi_j(x)$.

The previous equations may be extended to n-ary composition, provided that input variables are linearly partitioned among the components; if duplication of some input is required, we should eventually take this cost into account, adding a linear overhead.

A particular case of (multi-variable) linear composition of frequent use in computability theory is *sequencing* with absence of communication, i.e. the case of a function

$$\phi_{g(i,j)}(x,y) = \phi_j(x); \phi_i(y) = \begin{cases} \phi_i(y) & \text{if } \phi_j(x)\downarrow \\ \bot & \text{otherwise} \end{cases}$$

The aim is merely to force the evaluation of $\phi_j(x)$ before computing $\phi_i(y)$. According to our definitions

$$\Phi_{g(i,j)} \in \Theta(\lambda xy.(\Phi_j(x) + \Phi_i(y)))$$

for time and

$$\Phi_{g(i,j)} \in \Theta(\lambda xy.max\{\Phi_j(x), \Phi_i(y)\})$$

for space.

In combination with the s-m-n property, fixing the parameter x, we have, *for both time and space*:

LEMMA 13. *There exists a total computable function c such that*

$$(a) \quad \begin{cases} \phi_{c(i,j,x)}(y) = \phi_i(y) & \textit{if } \phi_j(x)\downarrow \\ \phi_{c(i,j,x)}(y) = \bot & \textit{otherwise} \end{cases}$$

$$(b) \quad \begin{cases} \Phi_{c(i,j,x)} \in \Theta(\Phi_i) & \textit{if } \phi_j(x)\downarrow \\ \Phi_{c(i,j,x)} = \bot & \textit{otherwise} \end{cases}$$

In fact, the previous Lemma is all we need for the generalization of Rice's Theorem to Complexity Cliques.

THEOREM 14. *Under the s-m-n (Def. 11) and the linear-composition (Def. 12) assumptions (either in time or space), a Complexity Clique P is recursive if and only if it is trivial, i.e. $P = \emptyset \vee P = \omega$.*

Proof. Assume P is recursive and let ϕ_p be its characteristic function. Suppose P is not trivial, so there exist a and b such that $\phi_p(a) = 1$ and $\phi_p(b) = 0$. Let m be the index of a program computing the everywhere divergent function; either $\phi_p(m) = 1$ or $\phi_p(m) = 0$. Let us consider the latter case, the other one being analogous.

Let K be a r.e. not recursive set, and let ϕ_k be its semi-decision function, i.e. $dom(\phi_k) = \mathcal{K}$. Consider the following function: $f(x,y) = \phi_k(x); \phi_a(y)$. By Lemma 13 there exists a total computable function c such that

$$(a) \quad \begin{cases} \phi_{c(a,k,x)}(y) = \phi_a(y) & \text{if } x \in \mathcal{K} \\ \phi_{c(a,k,x)}(y) = \bot = \phi_m(y) & \text{otherwise} \end{cases}$$

$$(b) \quad \begin{cases} \Phi_{c(a,k,x)} \in \Theta(\Phi_a) & \text{if } x \in \mathcal{K} \\ \Phi_{c(a,k,x)} = \bot = \Phi_m(y) & \text{otherwise} \end{cases}$$

Since P is a Complexity Clique,

$$\phi_p(c(a,k,x)) = \begin{cases} \phi_p(a) = 1 & \text{if } x \in \mathcal{K} \\ \phi_p(m) = 0 & \text{otherwise} \end{cases}$$

So, $\lambda x.\phi_p(c(k,a,x))$ is a characteristic function for $\mathcal{K}$; c is computable, hence ϕ_p cannot be.
The case $\phi_p(m) = 1$ is similar.

The knowledgeable reader will have certainly recognized, in part (a) of the above proof, the traditional Rice's argument. We just rephrased it to put in evidence the two elementary operations required by the transformation, namely composition and s-m-n. Providing a complexity bound for such operations is the key point that allows us to take complexity into account in a formal way. However, it is *clear* that, in concrete computational models, part (b) is always satisfied (that is, the program $\phi_x(x); \phi_a(y)$ has, for any given x, the same computational complexity as $\phi_a(y)$.

5. Generalized Rice-Shapiro's Theorem

Rice's Theorem provides a simple structural criterion for an extensional set to fall in Σ_0. Rice-Shapiro's Theorem[2] does the same for Σ_1. In a modern terminology, the result states that any completely r.e. set A of partial functions is *upward closed* and *compact*, i.e.

$$\phi_i \in A \Leftrightarrow \exists u \; finite, u \in A \wedge u \leq \phi_i$$

Let us start considering *upward closedness*, or monotonicity. The proof is not as standard as the one for Rice's Theorem, and not every proof is suitable for a complexity-theoretic generalization. We recall here the argument in Odifreddi (1997), that is particularly close to our approach. Suppose there exist two partial recursive functions ϕ_i and ϕ_j such that $\phi_i \in A$, $\phi_j \notin A$ and $\phi_i \leq \phi_j$. Let f be a recurtsive function such that

$$\phi_{f(e)}(x) = y \Leftrightarrow \phi_i(x) = y \vee (e \in K \wedge \phi_j(x) = y)$$

where K is an arbitrary r.e. non recursive set. Observe that the function f is well defined and computable just because $\phi_i \leq \phi_j$. Then

$$e \in \overline{K} \Leftrightarrow \phi_{f(e)} \in A$$

and since A is completely r.e. $\overline{K}$ would be r.e. too, that is a contradiction.
The key point of the above proof essentially consists in running in parallel two functions, namely $\phi_i(x)$ and $\phi_k(e); \phi_j(x)$ where ϕ_k is a semidecision function for K. In order to study the complexity of the composite function, we need some assumptions about the complexity of *parallel composition*. Remarkably, the subject was investigated by Landweber and Robertson (1972) a long time ago and for quite different reasons.

DEFINITION 15. *(Landweber and Robertson 1972) A pair $\langle \phi, \Phi \rangle$ has the* parallel computation *property if there exists a total computable function h such that*

$$(a) \quad \phi_{h(i,j)}(x) = \begin{cases} \phi_i(x) & \textit{if } \Phi_i(x) \leq \Phi_j(x) \\ \phi_j(x) & \textit{otherwise} \end{cases}$$

$$(b) \quad \Phi_{h(i,j)} \in \Theta(\lambda x.min\{\Phi_i(x), \Phi_j(x)\})$$

THEOREM 16. *Let P be a r.e. Complexity Clique. If $i \in P$, ϕ_i is finite and $\phi_i \leq \phi_j$ then $j \in P$.*

Proof. Suppose there exist i and j such that $i \in P$, ϕ_i is finite, $\phi_i \leq \phi_j$ but $j \notin P$. Let K be a r.e. not recursive set, and let ϕ_k be its semi-decision function, i.e. $dom(\phi_k) = \mathcal{K}$. By Lemma 13 there exists a total computable function c such that

$$\phi_{c(j,k,x)}(y) = \phi_k(x); \phi_j(y)$$

Let h be the function of the parallel computation property, and let us consider

$$\phi_{h(i,c(j,k,x))}(y) = \begin{cases} \phi_i(y) & \text{if } \Phi_i(y) \leq \Phi_{c(j,k,x)}(y) \\ \phi_{c(j,k,x)}(y) & \text{otherwise} \end{cases}$$

[2] There seem to be some controversy on the paternity of this result; in particular, it is attributed to Shapiro (1956) by Rogers (1987), pag.324, and to Myhill and Shepherdson (1955) by (Odifreddi 1997), pag.206.

We claim that

$$\phi_{h(i,c(j,k,x))}(y) = \begin{cases} \phi_i(y) & \text{if } x \notin \mathcal{K} \\ \phi_j(y) & \text{otherwise} \end{cases}$$

If $x \notin \mathcal{K}$ then $\phi_{c(j,k,x)}(y) \uparrow$ so, by definition of $\phi_{h(i,c(j,k,x))}$, either $\Phi_i(y) \downarrow$ and then $\phi_{h(i,c(j,k,x))}(y) = \phi_i(y)$ or otherwise both $\phi_{h(i,c(j,k,x))}(y)$ and $\phi_i(y)$ diverge.
Conversely, suppose $x \in \mathcal{K}$. If $\phi_j(y) \uparrow$, since $\phi_i \leq \phi_j$, also $\phi_i(y) \uparrow$ and hence $\phi_{h(i,c(j,k,x))}(y) \uparrow$. If $\phi_j(y) \downarrow$ then, even if $\Phi_i(y) \leq \Phi_{c(j,k,x)}(y)$ since $\phi_i \leq \phi_j$ we have $\phi_i(y) = \phi_j(y)$, and so in any case $\phi_{h(i,c(j,k,x))}(y) = \phi_j(y)$. The second claim is that

$$\Phi_{h(i,c(j,k,x))} \in \begin{cases} \Theta(\Phi_i) & \text{if } x \notin \mathcal{K} \\ \Theta(\Phi_j) & \text{otherwise} \end{cases}$$

Indeed if $x \notin \mathcal{K}$ then $\Phi_{c(j,k,x)} = \bot$ and

$$\lambda y.min\{\Phi_i(y), \Phi_{c(j,k,x)}(y)) = \Phi_i$$

Conversely, if $x \in \mathcal{K}$, by Lemma 13, $\Phi_{c(j,k,x)} \in \Theta(\Phi_j)$, and since by hypothesis ϕ_i is finite, $\lambda y.min\{\Phi_i(y), \Phi_{c(j,k,x)}(y)) = \Phi_{c(j,k,x)}$ almost everywhere, so

$$\Theta(\lambda y.min\{\Phi_i(y), \Phi_{h(i,c(j,k,x))}(y)\} = \Theta(\Phi_j)$$

In conclusion, since P is a Complexity Clique, having assumed $i \in P$ and $j \notin P$, we have

$$h(i, c(j, k, x)) \in P \leftrightarrow x \notin \mathcal{K}$$

and thus P cannot be r.e.

Let us remark that the hypothesis that ϕ_i is finite plays a crucial role in the previous proof. The point is that we have to compute a function similar to ϕ_i, if $x \notin \mathcal{K}$ and a function similar to ϕ_j if $x \in \mathcal{K}$. The parallel composition of ϕ_i and ϕ_j, when $x \in \mathcal{K}$, might be too fast for our purposes (it could belong to our complexity clique even if ϕ_j does not). Take for instance the case $\phi_i \cong \phi_j$ but Φ_i is uniformly faster then Φ_j: in this case the parallel composition of i and j gives an algorithm with the same complexity as i.

As a corollary of Theorem 16 we get the following alternative proof of Rice's Theorem (we repeat the statement both for completeness and for ease of reference to the new proof):

THEOREM 17. *A Complexity Clique P is recursive if and only if it is trivial, i.e. $P = \emptyset \vee P = \omega$.*

Proof. Let P be a recursive Complexity Clique. Then both P and $\overline{P}$ are r.e. and both are Complexity Cliques, since Complexity Cliques are closed under complementation. Let m be and index for the everywhere divergent function. Eventually, $m \in P$ or $m \in \overline{P}$. If $m \in P$ then by Theorem 16 the index of *any* function must be in P, so $P = \omega$. Similarly, if $\bot \in \overline{P}$ then $\overline{P} = \omega$.

Let us now come to the converse of Theorem 16, that is *compactness*. In this case, Odifreddi's proof runs as follows. Suppose A is a completely r.e. set of partial computable functions; assume $\phi_i \in A$ but no finite subfunction of ϕ_i is in A. Let $g(t)$ be the partial computable function such that

$$g(x,t) = \begin{cases} \uparrow & \text{if } \phi_x(x) \downarrow \text{ in less then } t \text{ steps} \\ 0 & \text{otherwise} \end{cases}$$

and consider the function $\phi_{f(e)}(x) = g(e, x); \phi_i(x)$. It is clear that $\phi_{f(e)} \cong \phi_i$ if $\phi_e(e) \uparrow$ and $\phi_{f(e)}$ is some finite subfunction of $\phi_i(x)$ if $\phi_e(e) \downarrow$. Hence, $\phi_{f(e)} \in A \Leftrightarrow e \in \overline{K}$, that is a contradiction.
We could probably argue that the complexity of $g(x,t)$, when it terminates, is at most linear in t. However, this seems to require some delicate assumptions on the complexity of the intrepreter, and we finally decided to follow a slightly different route.

First of all we need a slight generalization of the parallel composition property. The problem is that definition 15 does not allow to apply different continuations to the two parallel computations after termination of one of them.

DEFINITION 18. *A pair $\langle \phi, \Phi \rangle$ has the* generalized parallel computation *property if there exists a total computable function p such that for all i, i', j, j'*

$$(a) \quad \phi_{p(i,i',j,j')}(x) = \begin{cases} \phi_{i'}(\phi_i(x)) & \textit{if } \Phi_i(x) \leq \Phi_j(x) \\ \phi_{j'}(\phi_j(x)) & \textit{otherwise} \end{cases}$$

$$(b) \quad \Phi_{p(i,i',j,j')} \in \Theta\left(\lambda x. \begin{cases} \Phi_{h(i',i)}(x) & \textit{if } \Phi_i(x) \leq \Phi_j(x) \\ \Phi_{h(j',j)}(x) & \textit{otherwise} \end{cases}\right)$$

where h is the sequential composition function of Definition 12.

Even if the definition is a bit involved, the generalized parallel computation property looks like a natural extension of property 15: roughly, the idea is that, given two algorithms, we may "arbitrarily" fix a checkpoint in their respective code, run them in parallel and, according to which one reaches its checkpoint first, drop the other computation. In order to reduce the generalized property to the simpler one, it looks enough to state the existence of pairs and a test function (we must know which one of the two computations terminated first), but then we should also discuss the complexity of all these new notions, that would drive us a bit too far away.

THEOREM 19. *Let P be a r.e. Complexity Clique. If $i \in P$ and $\Phi_i \notin O(1)$ then there exists j such that ϕ_j is finite, $\phi_j \leq \phi_i$ and $j \in P$.*

Proof. Let K be a r.e. not recursive set, and let ϕ_k be its semi-decision function, i.e. $dom(\phi_k) = \mathcal{K}$. Let m be an index for the everywhere divergent function, and let I be an index for the identity. By the s-m-n property there exists a total computable function c such that

$$\phi_{c(x,k)}(y) = \phi_k(x)$$

$\phi_{c(x,k)}$ is either the everywhere divergent function if $x \notin \mathcal{K}$ or a constant function with complexity $\Phi_{c(x,k)} \in \Theta(\lambda y.\Phi_k(x)) = \Theta(1)$ otherwise. By the *generalized parallel computation* property,

$$(a) \quad \phi_{p(i,I,c(x,k),m)}(y) = \begin{cases} \phi_i(y) & \text{if } \Phi_i(y) \leq \Phi_k(x) \\ \uparrow & \text{otherwise} \end{cases}$$

$$(b) \quad \Phi_{p(i,I,c(x,k),m)} \in \Theta\left(\lambda y. \begin{cases} \Phi_i(y) & \text{if } \Phi_i(y) \leq \Phi_k(x) \\ \uparrow & \text{otherwise} \end{cases}\right)$$

If $x \notin \mathcal{K}$ then for any y $\Phi_i(y) \leq \Phi_k(x)$, hence $\phi_{p(i,I,c(x,k),m)}(y) = \phi_i(y)$, and the two functions also have the same complexity. If $x \in \mathcal{K}$, since $\Phi_i \notin O(1)$ then, $\Phi_i(y) \leq \Phi_k(x)$ only for a finite number of values for y, and hence $\phi_{p(i,I,c(x,k),m)}$ is a finite function.
If no index of finite sub-functions of ϕ_i is in P, we have

$$\phi_{p(i,I,c(x,k),m)} \notin P \leftrightarrow x \notin \mathcal{K}$$

and thus P cannot be r.e.

COROLLARY 20. *Let P be a r.e. Complexity Clique. If $i \in P$ and $\Phi_i \notin O(1)$ then for every j such that $\phi_j \cong \phi_i$ we have $j \in P$.*

Proof. By Theorem 19, there exists a finite sub-function $\phi_r \leq \phi_i$ such that $r \in P$, and by Theorem 16, any j such that $\phi_r \leq \phi_j$, *independently from its complexity* Φ_j, must belong to P.

COROLLARY 21. *No Complexity Clique of total functions and containing (indices of) programs with non constant complexity can be r.e.*

By the last corollary, we have e.g. that the class of programs with linear (polynomial, exponential, . . .) complexity is not r.e. (in fact, they are all Σ_2^0: see below).

Corollary 20 essentially says that every Complexity Clique in Σ_1^0 is (morally) an extensional set; in other words, all Complexity Cliques in Σ_1^0 have trivial complexity. Since the complement of an extensional set is also extensional, a similar result holds for Complexity Cliques in Π_1^0. Precisely:

COROLLARY 22. *Let P be a Complexity Clique in Π_1^0. If $i \in P$ then for every j such that $\Phi_j \notin O(1)$ and $\phi_j \cong \phi_i$ we have $j \in P$.*

Proof. Let $i \in P$ and consider j such that $\Phi_j \notin O(1)$ and $\phi_j \cong \phi_i$. If $j \notin P$ then $j \in \overline{P}$. Since $\overline{P}$ is a r.e. Complexity Clique, then we are in the hypothesis of Corollary 20 and we should have $i \in \overline{P}$, that is contradictory.

It is natural to wonder if there are Complexity Cliques with non trivial complexity in Σ_2^0.

THEOREM 23. *Let t be any total recursive function. Let C_t be the following Complexity Clique: $C_t = \{i | \Phi_i \in O(t)\}$. Then, $C_t \in \Sigma_2^0$.*

Proof. By definition, $\Phi_i \in O(t)$ if there exist n and c such that for any $m \geq n$, $\Phi_i(n) \leq ct(n)$. The relation $\Phi_i(n) \leq ct(n)$ is decidable by definition of complexity measure (Definition 2). Hence, C_t is Σ_2^0.

Since Σ_2^0 is closed under r.e. unions, many interesting Complexity Cliques (such as, for instance, the Cliques of programs with polynomial complexity) are in Σ_2^0.

6. Kleene's Second Fixed Point Theorem

Kleene's Second Fixed Point Theorem, in Rogers' formulation (Rogers 1987), states that for any total recursive function f there exists an indices i such that $\phi_i \cong \phi_{f(i)}$. The question is if we can always find a fixpoint i with the same complexity as $f(i)$.

In the general framework given by an arbitrary abstract complexity measure, the best we can prove is that the two complexity of i and $f(i)$ are related by a total recursive function. More precisely[3]:

THEOREM 24. *(Blum 1971) There exists a binary total recursive function h monotonically increasing in its second argument such that, for any total recursive function ϕ_i there exists an index m such that, for any x,*

$$\begin{array}{ll} (1) & \phi_m(x) = \phi_{f(m)}(x) \\ (2) & \Phi_m(x) \leq h(x, \Phi_{f(m)}(x)) \\ (3) & \Phi_m(x) \geq \Phi_{f(m)}(x) \end{array}$$

Fixing a particular model of computation, we may of course provide a better estimation for h. The interesting point is that, assuming the s-m-n property, the complexity of h uniquely depends from the complexity of the universal function ϕ_u:

THEOREM 25. *Let $\langle \phi, \Phi \rangle$ be an abstract complexity measure with the s-m-n property 11, and let u be an index for the universal function. Then for any total recursive function ϕ_i there exists an index m such that, for any x,*

$$\begin{array}{ll} (1) & \phi_m \cong \phi_{\phi_i(m)} \\ (2) & \Phi_m \in \Theta(\lambda y.\Phi_u(\phi_i(m), y) \end{array}$$

Proof. Consider the following computable function:

$$g(x, y) = \phi_{\phi_i(\phi_x(x))}(y) = \phi_u(\phi_u(i, (\phi_u(x, x))), y)$$

By the s-m-n theorem there exists a computable function s such that $\phi_{s(x)} = \lambda y.g(x, y)$ and, for any x,

$$\begin{array}{rcl} \Phi_{s(x)} & \in & \Theta(\lambda y.\Phi_u(\phi_u(i, (\phi_u(x, x))), y)) \\ & = & \Theta(\lambda y.\Phi_u(\phi_i(\phi_x(x)), y)) \end{array}$$

(x is a fixed parameter, hence the complexity of computing $\phi_i(\phi_x(x))$ does not matter). Since s is a total computable function there exists p such that $s = \phi_p$. $\phi_p(p)\downarrow$ since ϕ_p is total. Moreover we have:

$$\phi_{\phi_p(p)}(y) = g(p, y) = \phi_{\phi_i(\phi_p(p))}(y)$$

and

$$\Phi_{\phi_p(p)} = \Theta(\lambda y.\Phi_u(\phi_i(\phi_p(p)), y)$$

DEFINITION 26. *We say that a universal function ϕ_u is* fair *if for any x*

$$\lambda y.\Phi_u(x, y) \in \Theta(\Phi_x)$$

The cost of interpreting a program i by a *fair* universal machine may only introduce a constant slow-down factor c w.r.t. the direct computation of i. However, the constant c may depend on i. Following (Jones 1993) we say that the universal machine is *efficient* when the constant c is independent from the interpreted program.

COROLLARY 27. *Let $\langle \phi, \Phi \rangle$ be an abstract complexity measure with the $s-m-n$ property 11. If it admits a fair universal function u then for any total recursive function ϕ_i there exists an index m such that, for any x,*

$$\begin{array}{ll} (1) & \phi_m \cong \phi_{\phi_i(m)} \\ (2) & \Phi_m \in \Theta(\Phi_{\phi_i(m)}) \end{array}$$

Let us remark that the previous result says almost nothing about the "efficiency" of the fixpoint; it only says that, independently from ϕ_i we may always find a fixpoint m as (in-)efficient as $\phi_i(m)$ (that is not surprising: a fixpoint program is as efficient as the program defined by its body).

Not all computational models seem to admit fair universal machines. Discussing the case of multi-tape Turing machines, Blum (1971) suggests an upper bound for $\Phi_u(i, j)$ given by $\Phi_i(j)^2 + j$, where the square is due to the fact that the universal machine has a given number of tapes, but may need to simulate additional ones (if we admit that the universal machine has at least two tapes, the cost is likely to be reduced to $log(\Phi_i(j))\Phi_i(j) + j$)[4].

However, many models, comprising single tape Turing machines, lambda calculus and combinatory logic have fair universal machines. Unfortunately, this the kind of results that, belonging to the so called "folklore" of these subjects, cannot be properly quoted or elaborated. We hope someone will assume soon or later the burden to formally prove these important properties, and also hope that the scientific community will be so wise to accept these contributions. As a remarkable exception, the existence of *efficient* universal functions is proved by Jones (1993) for several classes of computational models defining problems in deterministic linear time (i.e. computational models with very restricted capabilities).

[3] See (Odifreddi 1999) for the analogous theorem in terms of Kleene's formulation of the Fixed Point Theorem. This has been also investigated in (Hansen et al. 1989) but no theoretical measure is given: the emphasis of the paper is more on practical, implementative issues.

[4] In our opinion, the notion of multi-tape machine as a foundational model of computation is arguable: indeed, we are used to work with a fixed machinery (a given processor), that amounts to fix a single universal machine with a given number of tapes, emulating all other programs.

The strong version of the Fixed Point Theorem of Corollary 27 (when it holds) is a major tool for the investigation of Complexity Cliques, since it can be used with the same confidence of the traditional theorem w.r.t. extensional sets. As a simple example, let us consider the traditional proof of Rice's Theorem making use of a fixed point.

THEOREM 28. *A Complexity Clique P is recursive if and only if it is trivial, i.e. $P = \emptyset \vee P = \omega$.*

Proof. Assume P is recursive and let ϕ_p be its characteristic function. Suppose P is not trivial, so there exist a and b such that $\phi_p(a) = 1$ and $\phi_p(b) = 0$. Let us consider the following computable function

$$g(x) = \begin{cases} a & \text{if } \phi_p(x) = 0 \\ b & \text{if } \phi_p(x) = 1 \end{cases}$$

If ϕ_p is total recursive, so is g, hence we may apply Corollary 27, concluding that there exists an index m such that

$$\begin{array}{l} \phi_m \cong \phi_{g(m)} \\ \Phi_m \in \Theta(\Phi_{g(m)}) \end{array}$$

In particular, if $\phi_p(m) = 0$ we have

$$\begin{array}{l} \phi_m \cong \phi_a \\ \Phi_m \in \Theta(\Phi_a) \end{array}$$

Since P is a Complexity Clique and $\phi_p(a) = 1$ it should also be $\phi_p(m) = 1$: contradiction.
Similarly, if if $\phi_p(m) = 1$ we have

$$\begin{array}{l} \phi_m \cong \phi_b \\ \Phi_m \in \Theta(\Phi_b) \end{array}$$

Since P is a Complexity Clique and $\phi_p(b) = 0$ it should also be $\phi_p(m) = 0$, that is again a contradiction.

7. Conclusions

In this paper, we introduced the notion of Complexity Clique: a set of indices for programs closed under a *similarity* relation defined taking into account their extension and complexity. Recursive properties of Complexity Cliques have been investigated, sharpening classical results of Recursion Theory, like Rice's and Rice-Shapiro's Theorems. In particular, we proved that all recursive Complexity Cliques are trivial, and all Complexity Cliques in Σ_1^0 and Π_1^0 are in fact (morally) extensional (i.e. are trivial w.r.t. complexity). In this way, we rephrase classical theorems in computability theory in order to establish properties of classes of programs defined by complexity conditions, concluding that *no non-trivial complexity property is semi-decidable*. For instance, the recent field of implicit computational complexity - see eg Dal Lago and Baillot (2006), Amadio (2005), citetAR02 and the biliography therein - studies criteria on programs implying some complexity properties. The results of this paper put in evidence that such criteria, whenever computable, can not be necessary and sufficient conditions. In other words, if polytime languages may be extensionally complete (computing all polynomial functions) they cannot be *intensionally* complete, that is express all polynomial *algorithms*.

The technical flavor of the paper follows Blum's axiomatic approach, but for the aims of the proofs we extended Blum's axioms with stronger assumptions, concerning the complexity of the s-m-n function, of sequential and parallel composition, and of the universal machine. Not all assumptions are needed for all results and the study of complexity seem to allow a deeper, fine grained, investigation even of traditional aspects of Recursion Theory (see e.g. the three different proofs of the generalized Rice's Theorem: Theorems 14-17-28).

It is possible (but not evident) that by relaxing the complexity condition in the notion of similarity (e.g. to recursive relatedness via some total recursive function h) we could prove (much) weaker results in the full generality of an arbitrary abstract complexity measure (i.e. without requiring any additional axiom). However, our axioms are *very* natural, and it is not clear if the effort would be worth the result.

From a strictly technical point of view, the requirement $i \notin O(1)$ in Theorem 19 (and followings) is annoying (more from an aesthetic point of view than a practical one). It would be nice to find a way to avoid it (or, alternatively, to prove that it is indeed an essential hypothesis).

We believe that most of the interesting problems investigated at the beginning of the seventies for Complexity Classes can also be studied, possibly more profitably, in the framework of Complexity Cliques. A typical example is recursive presentability, that is the problem to provide effective enumerations of representatives (up to extensional equivalence) for all elements in the given set (such as the problem of giving an effective enumeration of programs computing all functions with polynomial complexity). As already remarked by several authors - see e.g. Landweber and Robertson (1972); Young (1969) - , the problem does not make much sense for Complexity Classes: since complexity Classes are extensional you could enumerate "bad" programs (not within the given complexity bound) for "good" functions (functions admitting a program within the given complexity bound). The problem has been traditionally solved introducing a notion of *quality* for presentations (Landweber and Robertson 1972), but Complexity Cliques, due to their intensional nature, could provide an alternative and possibly more natural framework to work with.

One of the appealing aspects of Complexity Cliques is that, (at the contrary of Complexity Classes, for which even basic compositional properties fail) they have a very nice set-theoretic structure: they are a Complete Boolean Lattice (Theorem 9).

In a categorical perspective, Complexity Cliques, equipped with the similarity relation, are partial equivalence relations, and it looks natural to investigate them as a sub-category of PER (Freyd et al. 1992). The natural notion of morphism seems to be that of a complexity-preserving effective operator (Dekker and Myhill 1958); note that, however, in order to define a category you immediately need some additional assumptions on the nature of composition beyond Blum's axioms. As it is often the case, Category Theory may turn out to be the best tool to grasp the essence of the notions under investigation.

Finally, from the point of view of Abstract Complexity Theory, we hope that our work can help to revitalize a field that, lately, has been a bit starving. In particular, in (Asperti and Ciabattoni 1995) it was proved that any enumerated collection of functions containing projections, a universal function, and closed w.r.t. composition and the s-m-n construction is algorithmically complete. For this reason, an abstract approach based on complexity assumptions for these basilar constructions looks particularly appealing.

Acknowledgments

We are grateful to G.Longo, S.Martini, U.Dal Lago and Claudio Sacerdoti Coen for a few interesting discussions on the subject of this paper.

References

R. Amadio. Synthesis of max-plus quasi interpretations. Fundamenta Informaticae 65 (2005), pp 29–60. IOS Press.

A. Asperti, A. Ciabattoni. Effective applicative structures. In *Proceedings the sixth biannual conference on Category Theory in Computer Science (CTCS'95)*, volume 953, pp. 158–174, 1995.

A.Asperti, L.Roversi. Intuitionistic Light Affine Logic (Proof-nets, Normalization Complexity, Expressive Power). ACM Transactions on Computational Logic (TOCL), Volume 3 , Issue 1, January 2002, pp.137–175.

M. Blum. A machine-independent theory of the complexity of recursive functions. *Journal of ACM*, 14(2), pp.322–336, April 1967.

M. Blum. On effective procedures for speeding up algorithms. *Journal of ACM*, 18(2), pp.290–305, April 1971.

E. Börger. *Berechenbarkeit, Komplexität, Logik*. Friedr. Vieweg & Sohn, Braunshweig, 1986.

A. Borodin. Computational complexity and the existence of complexity gaps. *Journal of ACM*, 19, pp.158–174, March 1972.

N. J. Cutland. *Computability: An Introduction to Recursive Function Theory*. Cambridge University Press, Cambridge, UK, 1986.

U.Dal Lago, P.Baillot. On light logics, uniform encodings and polynomial time. Mathematical Structures in Computer Science, V.16, Issue 4, pp. 713 - 733, August 2006.

J. C. E. Dekker and J. Myhill. Some theorems on classes of recursively enumerable sets. *Trans. Amer. Math. Soc.*, 89, pp.25–59, 1958.

P. Freyd, P. Mulry, G. Rosolini, and D. Scott. Extensional pers. *Information and Computation*, 98(2), pp.211–227, June 1992.

T. A. Hansen, T. Nikolajsen, J. L. Träff, and N. D. Jones. Experiments with implementations of two theoretical constructions. In *Logic at Botik*, volume 363, pages 119–133, 1989.

J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc.*, 117, pp.285–306, 1965.

Neil D. Jones. Constant time factors *do* matter. In *Proceedings of STOC'93*, pp.602–611, 1993.

D. Kozen. Indexing of subrecursive classes. *Theoretical Computer Science*, 11, pp.277–301, 1980.

L. H. Landweber and E. L. Robertson. Recursive properties of abstract complexity classes. *Journal of ACM*, 19(2), pp.296–308, April 1972.

F. D. Lewis. Unsolvability considerations in computational complexity. In *Proceedings of Second ACM Symp. on Theory of Computing*, pp.22–30, 1970.

G. Lischke. Über die erfüllung gewisser erhaltungssätze durch kompliziertheitsmasse. *Zeit. Math. Log. Grund. Math.*, 21, pp.159-166, 1975.

G. Lischke. Natürliche kompliziertheitsmasse und erhaltungssätze i. *Zeit. Math. Log. Grund. Math.*, 22, pp.413-418, 1976.

G. Lischke. Natürliche kompliziertheitsmasse und erhaltungssätze ii. *Zeit. Math. Log. Grund. Math.*, 23, pp.193-200, 1977.

E. M. McCreight and A. R. Meyer. Classes of computable functions defined by bounds on computation. In *Proceedings of ACM Symp. on Theory of Computing, Marina del Rey, Calif.*, pp. 79–88, 1969.

J.Myhill, J.C.Shepherdson. Effective operations on partial recursive functions. *Zeit.Math. Log. Grund. Math.*, 1, pp.310-317, 1955.

P. G. Odifreddi. *Classical Recursion Theory: the Theory of Functions and Sets of Natural Numbers*, volume 125 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1997.

P. G. Odifreddi. *Classical Recursion Theory, Volume II*, volume 143 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 1999.

J. G. Rice. Classes of recursively enumerable set and their decision problems. *Trans. Amer. Math. Soc.*, 74, pp.358–366, 1953.

H. Rogers. *Theory of Recursive Functions and Effective Computability*. MIT press, 1987.

N.Shapiro. Degrees of computability. Transactions of the American Mathematical Society, 82, pp.281-299, 1956.

P. R. Young. Toward a theory of enumeration. *Journal of ACM*, 16(2), pp.328–348, 1969.