More intensional versions of Rice's Theorem

Jean-Yves Moyen*
Jean-Yves.Moyen@univ-paris13.fr

Jakob Grue Simonsen†
simonsen@diku.dk

Department of Computer Science, University of Copenhagen (DIKU)
Njalsgade 128-132, 2300 Copenhagen S, Denmark

Abstract

Classic results in computability theory are almost invariably extensional: they concern the behaviour of partial recursive functions rather than the programs computing them. We provide generalised versions of two classic results Rice's Theorem and the Rice-Shapiro Theorem and demonstrate how they may be applied to study intensional properties such as time and space complexity. In particular we obtain simple proofs of several striking negative results about overapproximations of intensional properties, for example that any decidable property of programs that (strictly) contains the set of polytime program must contain programs of arbitrarily high time complexity.

1 Introduction

A cornerstone of computability theory is Rice's Theorem [Ric53]: any non-trivial extensional set of programs is undecidable (extensionality roughly means that the set depends only on the partial functions computed by the program). This very generic formulation allows to prove undecidability of a variety of sets *e.g.* "programs that, on input 0, return 42", "programs that compute a bijection" or "programs that compute a non-total function".

Rice's Theorem showcases a fundamental dichotomy between programs and the partial functions they compute: it gives an undecidability criterion for sets of programs, but these sets are defined by the function computed by the programs. Underlying this dichotomy and the Theorem is the notion of *extensional equivalence* "two programs are equivalent iff they compute the same function". Rice's Theorem basically tells us that this equivalence is undecidable.

Even after 60 years, scant research has been made in *intensional* analogues of Rice's Theorem, i.e. undecidability results concerning *how* programs compute rather than *what* they compute. One exception is Asperti's work on *complexity cliques* [Asp08]. Roughly, Asperti refines the extensional equivalence relation into the equivalence relation "two programs are equivalent if they compute the same function with comparable (up to big-Θ) complexity" and then states the corresponding result that this equivalence relation is also undecidable.

In this work, we generalise Rice's Theorem in a different direction. Rather than trying to refine the extensional equivalence relation and find other, more precise, ones that are still undecidable, we will first lift the strict extensionality of sets and then generalise to any kind of equivalence between programs. The first generalisation is to remove the strict extensionality of the sets considered. We will only ask that the studied sets accept all the programs computing one given function, but we put no condition on programs computing other functions (they may or not be in the set). This allows to consider more intensional sets of programs. The second generalisation is to completely change the equivalence relations considerd. We must still impose very general conditions on such relations (via so-called *switching families*). This allows both to refine the equivalence (*i.e.*, Asperti's Result fits perfectly in the generalised setting) and to consider completely different equivalences.

Our work has implications for Implicit Computational Complexity (ICC). Classical ICC provides sound but incomplete criteria for a given property (such as computing a Ptime function). The criteria

*Supported by the Marie Skłodowska–Curie action "Walgo", program H2020-MSCA-IF-2014, number 655222
†Partially supported by the Danish Council for Independent Research *Sapere Aude* grant "Complexity via Logic and Algebra" (COLA).

are *sound* in the sense that being accepted certifies the complexity bound, but *incomplete* in the sense that some "good" programs are nonetheless rejected (but extensional completeness is usually required, e.g. every PTIME function is computed by some program accepted, but not all programs running in polynomial time are accepted). That is, a classical ICC criterion admit false negatives but guarantees the absence of false positives. In this sense, an ICC criterion can be seen as a certificate of good behaviour by the program. It is an *under*-approximation of the target set.

Consider the *non*-classical view: what if we try to characterise *over*-approximations rather than under-approximations? From an ICC point of view, that means a criterion that accepts false positives but no false negatives. Every program in the target set should be recognised but we will also accept some "bad eggs" in the process. The hope being that there will be few of them and that they won't be too bad. Typically, it would make sense to accept all the polytime programs plus "a few" exponential time ones. As long as there's not too many exponential ones and they're "only" exponential and not worse, this is still interesting. Typically, most accepted programs would run fast (e.g., in a couple of minutes) and some would be slower but not too slow.

Moreover, having over-approximation can be a good way to prove lower bounds on complexity. Classical complexity has been very good to provide upper bounds on complexity (e.g., SAT is in NP), but is bad at providing lower bounds (thus making separation results between complexity classes hard to prove). Classical ICC, via *under*-approximations, can only provide upper bounds: if the program is accepted, then it is guaranteed to have complexity at most the one of the criterion, but if the program is rejected, nothing is known. Having an *over*-approximation could help provide lower bounds: if the program is rejected, then it is guaranteed to have a "bad" behaviour.

However, our intensional versions of Rice's Theorem will show that over-approximations are patently not a viable path: There will, necessarily, always be *many* bad eggs and there will always be some *extremely* bad ones.

2 Notations and Rice's Theorem

We assume an unspecified, Turing-complete programming language, in the proofs we'll use an informal syntax for programs. We note $[\![\mathtt{p}]\!]$ the function computed by a program `p` and conversely $\mathcal{P}_f$ the set of all programs computing function f. We define the extensional equivalence, or Rice's equivalence, as $\mathtt{p}\mathfrak{R}\mathtt{q} \Leftrightarrow [\![\mathtt{p}]\!] = [\![\mathtt{q}]\!]$.

We say that a set of programs `P` is

- *non-trivial* if it is neither empty, nor the set of all programs.
- *extensional* if it is the union of classes of $\mathfrak{R}$;
- *partially extensional* for a set of functions F if it contains all the programs computing a function in F: $\mathcal{P}_F \subset \mathtt{P}$ (over approximation of $\mathcal{P}_F$). It does not need to be extensional for other functions;
- *extensionally complete* for a set a functions F if it can computes all these functions: $F \subset [\![\mathtt{P}]\!]$;
- *extensionally sound* for a set a functions F if it can compute only these functions: $[\![\mathtt{P}]\!] \subset F$ (under approximation of $\mathcal{P}_F$);
- an *ICC characterisation* of a set of functions F if it is both extensionally sound and complete for F, *i.e.* it computes exactly these functions: $[\![\mathtt{P}]\!] = F$;
- *extensionally universal* if it is extensionally complete for the set of computable partial functions (each computable partial function is computed by at least one program in `P`).

Note that any extensionally universal set of programs must be infinite as there are infinitely many computable functions.

Theorem 2.1 (Rice[Ric53])**.** *Any (non-trivial) extensional set of programs is undecidable.*

Sketch of proof. Let `p` be in the set and suppose that the infinite loop isn't. Consider the program `q'(x) = q(0); p(x)`. It computes the same thing as `p` (and hence is in the set) iff `q`(0) terminates (and loops otherwise), an undecidable property. □

Note for later reference, that Rice's Theorem could equivalently be formulated as: "Every decidable extensional set of programs is trivial".

3 The Rice and Rice-Shapiro Theorems, intensionally

We now provide intensional versions of two classic results; the proofs use simple tools from classical recursion theory, but are different from the proofs of the classical extensional versions.

3.1 Rice, intensional version

Definition 3.1 (Recursively separable)**.** Two sets A and B are said to be *recursively separable* (a concept due to Smullyan [Smu58]) if there is a decidable set C such that $A \subseteq C$ and $B \bigcap C = \emptyset$. The sets A and B are said to be *recursively inseparable* if they are not recursively separable.

That is, C may overapproximate A, but will never contain elements of B. A classical example of recursively inseparable sets (see [Pap94], Section 3.3) is:

Lemma 3.2. *Let* $A = \{\, \mathtt{p} \;:\; [\![\mathtt{p}]\!](0) = 0 \,\}$ *and* $B = \{\, \mathtt{p} \;:\; [\![\mathtt{p}]\!](0) \notin \{0, \bot\} \,\}$*. They are recursively inseparable.*

Our first generalisation of Rice's Theorem is to replace the "extensional" condition by partial extensionality.

Theorem 3.3. *Any non-empty decidable partially extensional set of programs is extensionally universal.*

That is, any decidable over-approximation of an extensional set of programs contains at least one program for any computable partial function. Among other, it must contain at least one program that always loops. Thus, for example, a decidable set of programs capturing all the programs that compute the constant function $x \mapsto 0$ must also capture (at least) one program that always loops. There is no hope of reaching intensional completeness and keeping extensional soundness.

Note that, contrary to Rice's Theorem, we only require the set of program to be non-empty rather than non-trivial. Indeed, the trivial set of all programs is certainly decidable, partially extensional and extensionally universal.

Sketch of proof. Suppose that the set is partially extensional for $[\![\mathtt{p}]\!]$ but contains no program computing $[\![\mathtt{q}]\!]$. Consider program `r'(x) = if r(0)=0 then p(x) else q(x)`. It is in the set if `r(0) = 0` and out if `r(0)` terminates on another value, thus deciding the set would recursively separate A and B above. □

Corollary 3.4 (Rice's Theorem)**.** *Any (non-trivial) extensional set of programs is undecidable.*

Sketch of proof. An extensional and extensionally universal set must be the set of all programs. □

This Theorem has as immediate corollaries several folklore results that are usually explained by simple ad-hoc arguments (e.g. "adding arbitrarily many dummy `x:=x` instructions")–the theorem provides a uniform means of proving these. It is also more general than Rice's Theorem.

Corollary 3.5 (Many programs)**.** *For any computable function f, there are infinitely many programs computing it.*

Proof. If $\mathcal{P}_f$ is finite, it is decidable. By construction, it is partially extensional for f, hence by the Theorem it must be extensionally universal and thus infinite, an absurdity. □

Note that this is a bit less trivial than what it appears: The corollary states that it is not possible to design a programming language with only finitely many possible implementations of a given function without losing Turing completeness.

Corollary 3.6 (Arbitrary size)**.** *For any N, every computable function is computable by a program of size larger than N.*

Proof. Let $\mathtt{P} = \{\, \mathtt{p} \;:\; |\mathtt{p}| \leq N \,\}$. It is finite, hence by previous Corollary cannot contain all the programs computing a given function. □

3.2 Rice-Shapiro, intensional version

We can apply the same idea to the Rice-Shapiro Theorem Theorem [MS55, Sha56]:

Theorem 3.7 (Rice-Shapiro)**.** *Let F be a recursively enumerable set of computable functions. $f \in F \Leftrightarrow$ there exists $g \in F$ with finite domain D such that $\forall d \in D, f(d) = g(d)$.*

The "power" of this Theorem comes from the following construction: Suppose that f is in F. As $f \in F$, there exists g with finite domain D in F that agrees on f on this finite domain. Now, consider **any** function f' that also agrees on g on this finite domain. Using the $\Leftarrow$ direction of the Theorem, $f' \in F$. Thus, any function that agrees with f on a finite domain D must be in F and the only thing one can do to compare functions (especially to determine whether a given program computes a given function or not) is to check a finite number of inputs.

The intensional version is:

Theorem 3.8. *Let $\mathtt{P}$ be a recursively enumerable partially extensional set of programs. Then, for any computable function f, there exists a program $\mathtt{p} \in \mathtt{P}$ such that $[\![\mathtt{p}]\!]$ differs from f only on finitely many inputs.*

Note that $\mathtt{P}$ here is recursively enumerable and not necessarily decidable as in Rice's Theorem. For example, if $\mathtt{P}$ is a recursively enumerable set of programs partially extensional for a total function (e.g., the constant function $x \mapsto 0$), then by choosing f to be the totally undefined function, we can conclude that $\mathtt{P}$ must contain a program that loops on all but finitely many inputs, by choosing $f : x \mapsto 1$ we can conclude that $\mathtt{P}$ must also contain a program that return 1 on all but finitely many inputs.

4 Beyond Rice's Theorem

The second, and broader, generalisation we make is to choose other equivalence relations than the Rice relation $\mathfrak{R}$.

4.1 Switching families

Definition 4.1. Let S be a set and $\approx$ an equivalence relation on S. A *switching family compatible with* $\approx$ is a family $I = (\pi_s)_{s \in S}$ of computable total functions $\pi_s : S \times S \to S$ such that the sets $A_I = \{\, s \in S \;:\; \forall x, y. \pi_s(x, y) \approx x \,\}$ and $B_I = \{\, s \in S \;:\; \forall x, y. \pi_s(x, y) \approx y \,\}$ are recursively inseparable.

The definition implies that neither A_I nor B_I is decidable. Also, there may be elements in S neither in A_I nor in B_I such that $\pi_s(x, y)$ is not always equivalent to x and not always equivalent to y. Note that the classical projections $\pi_1(x, y) = x$ and $\pi_2(x, y) = y$ are switching functions but in general switching functions do not need to ignore one of their arguments.

The constructions of $\mathtt{r}'$ in the proof of the Theorem 3.3 constitutes a switching family with $\mathtt{r}'(x) = \pi_{\mathtt{r}}(\mathtt{p}, \mathtt{q})(x) =$ `if r(0) then p(x) else q(x)`. We call this the *standard switching family.*

4.2 Rice's Theorem, second generalisation

Let $\mathfrak{P} = \{\mathtt{P}_1, \mathtt{P}_2, \ldots\}$ be a partition of a set S and $\mathtt{P}$ be a subset of S, it is

- *compatible with* $\mathfrak{P}$ if it is the union of classes of $\mathfrak{P}$;
- *partially compatible* for a block $\mathtt{P}_k$ if it contains all the elements in that block: $\mathtt{P}_k \subset \mathtt{P}$. Note that it does not need to be compatible for other blocks;
- *complete* for a set of blocks $\mathtt{P}_{k_1}, \mathtt{P}_{k_2}, \ldots$ if it contains at least one element of each of these blocks: $\mathtt{P} \bigcap \mathtt{P}_{k_i} \neq \emptyset$;
- *universal* if it is complete for $\mathfrak{P}$ itself: it contains at least one element in each block of the partition.

Theorem 4.2. *Let $\mathfrak{P}$ be a partition of a set S and $I = (\pi_s)_{s \in S}$ be a switching family compatible with it. Any non-empty decidable partially compatible subset of S is universal.*

Sketch of proof. Let x be in a class included in the set and y be in a class that does not intersect it. Consider $s' = \pi_s(x, y)$. Deciding if s' is in the set would recursively separate A_I and B_I, which is impossible by definition of the switching family. □

4.3 Examples : computational complexity

We now come to our most striking example: any decidable set of of programs containing, for instance, the polynomial-time programs, must contain programs of arbitrarily high complexity.

Example 4.3 (Complexity)**.** Let Φ be a complexity measure in the sense of Blum. Let $\equiv_\Phi$ be the equivalence relation $\mathtt{p} \equiv_\Phi \mathtt{q}$ iff $\Phi_\mathtt{p} \in \Theta(\Phi_\mathtt{q})$.

The standard switching family is compatible with the above relation: when `r(0)` terminates it does so with a constant complexity, whence the global complexity is the one of `p` (resp. `q`) up to a constant additive factor.

Hence, any non-empty decidable set of programs partially compatible with $\equiv_\Phi$ is universal and must contain programs of arbitrarily high complexity.

This example can be specialised by making the partially compatible set more precise. Note that we consider sets of programs defined by properties of the programs themselves, and *not* necessarily by properties of the computed function (*i.e.* this is not the set of programs computing PTIME functions as it does not include inefficient programs that compute a "simple" function in a long time).

Example 4.4 (Polynomial time)**.** Let PPTIME be the set of *programs* whose runtime is polynomial. It is partially compatible with $\equiv_\Phi$ when Φ is the usual time complexity. Thus, any decidable set that includes all the *programs* computing in polynomial time must also include programs of arbitrarily high time complexity. Including not only exponential programs but also non primitive recursive ones and even non multiple recursive ones.

Any attempt at finding a decidable over-approximation of PPTIME is doomed to also contain many extremely "bad" programs.

Example 4.5 (Linear space)**.** Let PLINSPACE be the set of programs computing in linear space. It is partially compatible with $\equiv_\Phi$ when Φ is the usual space complexity. Thus, any decidable set that contains PLINSPACE must also contains programs of arbitrarily high space complexity.

Thus, even for sets not closed under composition, over-approximations cannot both be decidable and contain only programs of restricted space complexity.

Example 4.6 (Asperti-Rice, [Asp08])**.** Let $\mathfrak{A}$ be the equivalence relation $\mathtt{p}\mathfrak{A}\mathtt{q}$ iff $\big(\,[\![\mathtt{p}]\!] = [\![\mathtt{q}]\!]$ and $\Phi_\mathtt{p} \in \Theta(\Phi_\mathtt{q})\big)$. The standard switching family is compatible with it (seeing relations as subset of couples, we have $\mathfrak{A} = (\mathfrak{R} \bigcap \equiv_\Phi)$, the standard switching family being compatible with both).

Hence, any non-empty decidable set of programs partially compatible with $\mathfrak{A}$ is universal. This generalises Asperti's result that the only decidable compatibles sets (*i.e.* "Complexity cliques" in his formalism) are the trivial ones.

References

[Asp08] Andrea Asperti. The Intensional Content of Rice's Theorem. In *Proceedings of the 35th Annual ACM SIGPLAN - SIGACT Symposium on Principles of Programming Languages (POPL 2008)*, 2008.

[MS55] John R. Myhill and John Cedric Shepherdson. Effective operations on partial recursive functions. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 1:310–317, 1955.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Ric53] Henry Gordon Rice. Classes of Recursively Enumerable Sets and Their Decision Problems. *Transactions of the American Mathematical Society*, 74:358–366, 1953.

[Sha56] Normann Shapiro. Degrees of computability. *Transactions of the AMS*, 82:281–299, 1956.

[Smu58] Raymond M. Smullyan. Undecidability and recursive inseparability. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 4(7–11):143–147, 1958.