DOI 10.1007/s10469-016-9410-9
Algebra and Logic, Vol. 55, No. 5, November, 2016
(Russian Original Vol. 55, No. 5, September-October, 2016)

A GENERIC RELATION ON RECURSIVELY ENUMERABLE SETS

A. N. Rybalov*

UDC 510.5

Keywords: *generic relation, complete set, recursively enumerable set.*

We introduce the concept of a generic relation for algorithmic problems, which preserves the property of being decidable for a problem for almost all inputs and possesses the transitive property. As distinct from the classical m-reducibility relation, the generic relation under consideration does not possess the reflexive property: we construct an example of a recursively enumerable set that is generically incomparable with itself. We also give an example of a set that is complete with respect to the generic relation in the class of recursively enumerable sets.

A generic approach toward computability and complexity theory was propounded in [1]. In the frames of that approach, an algorithmic problem is considered not on the whole set of inputs but on some subset of almost all inputs. The concept of "almost all" is formalized by introducing a natural measure on a set of input data. From a practical viewpoint, fast decision algorithms for a problem on a generic set are as good as fast algorithms for all inputs. A classical example of such an algorithm is the simlex method—it solves a linear programming problem in polynomial time for most of the data but has exponential worst-case complexity. Moreover, a problem may turn out to be hard or altogether undecidable in the classical sense but easily decidable in the generic sense. In [1, 2], it was proved that many algorithmic problems of algebra exhibit such behavior, and in [3], a generic set was constructed on which the classical halting problem is decidable for Turing machines with a one-way infinite tape.

An important notion in classical computability theory is the notion of reducibility of algorithmic problems. Using it, we can compare problems with respect to computational complexity and develop

*Supported by Russian Science Foundation, project 14-11-00085.

Omsk Branch of Sobolev Institute of Mathematics, Siberian Branch, Russian Academy of Sciences, ul. Pevtsova 13, Omsk, 644099 Russia; Omsk State Technical University, pr. Mira 11, Omsk, 644050 Russia; alexander.rybalov@gmail.com. Translated from *Algebra i Logika*, Vol. 55, No. 5, pp. 587-596, September-October, 2016. Original article submitted April 13, 2016.

a rich theory [4]. In computational complexity theory, the notion of polynomial reducibility makes it possible to distinguish a very important class of NP-complete problems [5]. Attempts to define reducibility in the generic case meet with difficulties. The notion of reducibility (m-reducibility) in the classical case satisfies the following two natural properties:

if problem A reduces to problem B and B is decidable, then A is decidable;

if problem A reduces to problem B and B reduces to problem C, then A reduces to C.

For these properties to be satisfied in the generic case, it is natural to require that generic m-reducibility translate generic sets into generic sets. Unfortunately, on the one hand, this considerably narrows the class of computable functions which might be generic reducibilities. On the other hand, this complicates verification of the very fact of generic reducibility for particular problems: we need to prove that a computable function translates a generic set into a generic set. In [6, Def. 4.3], for instance, we can find the thus defined generic reducibility of sets of natural numbers—the universal quantification figuring in the definition is taken over *any* so-called generic description of a reducible set. In [7, Def. 1.9], another notion of generic reducibility is introduced, namely, one that uses a Turing functional for computing a characteristic function of one set given *any* generic oracle for computing a characteristic function for the other set.

In the present paper, we introduce the concept of a generic relation on sets satisfying the two properties above and not including a universal quantifier over any generic set or oracle. As distinct from the classical m-reducibility relation (and reducibilities in [6, 7]), the relation considered here does not possess the reflexive property: we construct an example of a recursively enumerable set that is generically incomparable with itself. We also give an example of a set that is complete with respect to the generic relation in the class of recursively enumerable sets. Note that an analogous completeness result for reducibilities in [7] is not mentioned explicitly, but follows readily from [7, Prop. 1.14, obs. 2.1]. A similar statement can also be derived from [6, Prop. 4.5, Lemma 4.6].

1. GENERIC ALGORITHMS

Let I be some set. A function $size : I \to \mathbb{N}$ is called the *size function* if, for any $n \in \mathbb{N}$, the set $I_n = \{x \in I : size(x) = n\}$ is finite. For instance, if $I = \Sigma^*$ is a set of words over a finite alphabet Σ, then the size function will be one that is defined for any word w as its length $|w|$. Also, for the set $\mathbb{N}$ of natural numbers, the size function assigns to any natural number the length of its binary representation. As is customary in computability theory, by algorithmic problems we mean recognition problems for subsets of some input set on which the length function is defined. For a subset $S \subseteq I$, we define the sequence

$$\rho_n(S) = \frac{|S_n|}{|I_n|},\ n = 1, 2, 3, \ldots,$$

where $S_n = S \cap I_n$ is a set of inputs of S of size n. Note that $\rho_n(S)$ is the probability of falling into S for a random and equiprobabilistic generation of inputs of I_n. The *asymptotic density* of S

is the limit (if it exists)

$$\rho(S) = \lim_{n\to\infty} \rho_n(S).$$

A set S is *generic* if $\rho(S) = 1$ and is *negligible* if $\rho(S) = 0$. Obviously, S is generic iff its complement $I \setminus S$ is negligible.

An algorithm $\mathcal{A}$ with input set I and output set $J \cup \{?\}$ ($? \notin J$) is said to be *generic* if:

(1) $\mathcal{A}$ halts on all inputs of I;

(2) $\rho(\{x \in I : \mathcal{A}(x) \neq ?\}) = 1$.

A generic algorithm $\mathcal{A}$ computes a function $f : I \to J$ if $\mathcal{A}(x) = y \in J \Rightarrow f(x) = y$ for all $x \in I$. The situation where $\mathcal{A}(x) = ?$ means that $\mathcal{A}$ cannot compute a function f at an argument x, while condition (2) guarantees that $\mathcal{A}$ correctly computes f on almost all inputs (inputs of the generic set). A set $S \subseteq I$ is *generically decidable* if there exists a generic algorithm computing its characteristic function.

2. GENERIC RELATION ON SETS

We introduce the concept of a generic relation $\leq_{Gen}$ on sets and prove its elementary properties.

Let I and J be some sets on which size functions are defined. For sets $A \subseteq I$ and $B \subseteq J$, $A \leq_{Gen} B$ if there exists a total computable function $f : I \times \mathbb{N} \to J \cup \{?\}$ with the following properties:

(1) $\forall x \in I$ if $f(x,0) \neq ?$, then $\forall n \in \mathbb{N}\ f(x,n) \neq ?$;

(2) $\rho(\{x \in I : f(x,0) \neq ?\}) = 1$;

(3) $\forall x \in I$ if $f(x,0) \neq ?$, then $\rho(\{f(x,n) : n \in \mathbb{N}\}) > 0$;

(4) $\forall x \in I$ if $f(x,0) \neq ?$, then

$x \in A \Rightarrow \forall n \in \mathbb{N}\ f(x,n) \in B$, and

$x \notin A \Rightarrow \forall n \in \mathbb{N}\ f(x,n) \notin B$.

Consider each of the above properties in more detail. By (3), the function f (if its value is not equal to ?) assigns to every element x of the set I of inputs in problem A a nonnegligible subset J of inputs in problem B. Property (4) guarantees that this subset is fully inside B, if $x \in A$, and is fully outside B if $x \notin A$. Properties (1) and (2) say that for a negligible set of inputs in I, the function f can yield an indefinite answer.

As distinct from classical m-reducibility which allows only one request about the belonging of an element $f(x)$ to the second set, the availability of a generic relation permits a potentially unbounded number of requests about the belonging of elements $f(x,n)$ to the second set, and these elements form a sufficiently large (nonnegligible) subset.

THEOREM 2.1. If $A \leq_{Gen} B$ and B is generically decidable, then A is generically decidable.

Proof. Let f be a function realizing the generic relation $A \leq_{Gen} B$, and let $\mathcal{B}$ be a generic algorithm computing a characteristic function of B. A generic algorithm $\mathcal{A}$ computing a

characteristic function of A operates on input x as follows. First we compute $f(x,0)$. If $f(x,0) = ?$ then we output ?. Otherwise, we compute the values

$$f(x,0),\ f(x,1),\ f(x,2),\ \ldots$$

until we obtain n such that $\mathcal{B}(f(x,n)) \neq ?$. This will always happen since, on the one hand,

$$\rho(\{y \in J : \mathcal{B}(y) \neq ?\}) = 1,$$

and on the other hand,

$$\rho(\{f(x,n) : n \in \mathbb{N}\}) > 0.$$

Therefore,

$$\{y \in J : \mathcal{B}(y) \neq ?\} \cap \{f(x,n) : n \in \mathbb{N}\} \neq \varnothing.$$

By property (4) in the definition of a generic relation, $\mathcal{B}(f(x,n))$ will be the correct value of the characteristic function of the set A at x. It remains to show that

$$\rho(\{x \in I : \mathcal{A}(x) \neq ?\}) = 1.$$

This follows from property (2) for a generic relation $\leq_{Gen}$. □

We prove that the generic relation is transitive.

THEOREM 2.2. If $A \leq_{Gen} B$ and $B \leq_{Gen} C$, then $A \leq_{Gen} C$.

Proof. Let f be a function realizing the generic relation from A to B and g be a function realizing the generic relation from B to C. We construct a total computable function h realizing the generic relation from A to C. A value for $h(x,n)$ is computed as follows. If $f(x,0) = ?$, then $h(x,n) = ?$ for any $n \in \mathbb{N}$. If $f(x,0) \neq ?$, then again we compute $f(x,0), f(x,1), \ldots$ until we find $k \in \mathbb{N}$ such that $g(f(x,k),0) \neq ?$. That this will happen is proved in the same way as in the proof of the previous theorem. Now we put $h(x,n) = g(f(x,k),n)$. It is easy to verify that the function h constructed realizes the generic relation from the set A to the set C. □

As distinct from the classical m-reducibility relation, the generic relation $\leq_{Gen}$ does not satisfy the reflexive property. This is shown in the following:

THEOREM 2.3. There exists a recursively enumerable set $W \subseteq \mathbb{N}$ such that $W \not\leq_{Gen} W$.

Proof. We show that as W we can take any simple recursively enumerable set of natural numbers, which is not generic. The existence of such (even negligible) sets was shown in [6, proof of Prop. 2.15]. Assume that $W \leq_{Gen} W$ and f is a computable function realizing the generic relation. Since W is not generic, $\mathbb{N} \setminus W$ is not negligible. Therefore, there exists $x \in \mathbb{N} \setminus W$ such that $f(x,0) \neq ?$. Hence $f(x,n) \in \mathbb{N} \setminus W$ for any n, and the immune set $\mathbb{N} \setminus W$ contains an infinite recursively enumerable set $\{f(x,0), f(x,1), f(x,2), \ldots\}$. We have arrived at a contradiction showing that $W \not\leq_{Gen} W$. □

3. GENERICALLY COMPLETE RECURSIVELY ENUMERABLE SETS

As in classical computability theory, a set W is said to be *generically complete* for some class of sets if $S \leq_{Gen} W$ for any set S in this class. We make a start from K_0, a set of Turing machine programs halting on input 0. It is known [4] that K_0 is m-complete in the class or recursively enumerable sets; i.e., all recursively enumerable sets are m-reduced to K_0.

We will consider Turing machines with input alphabet $\{0, 1\}$. A program of a Turing machine with inner states $\{q_0, q_1, \ldots, q_n\}$ over $\{0, 1\}$ consists of rules of the form $(q_i, a) \to (q_j, b, S)$, one rule for all nonzero (nonfinal) states q_i and symbols a of the alphabet. Altogether, there are $2n$ rules. Such a rule prescribes that a machine, being in a nonfinal state q_i and seeing (with its head) on the tape a symbol a, should go into state q_j (which may be the final q_0), write on the tape a symbol b, and move its head by a (left, right) shift $S \in \{R, L\}$. By the size of a machine we mean the number n of nonfinal inner states. Machines will be identified with their programs.

A Turing machine is said to be *normalized* if from any nonfinal state q_i (in its corresponding rules in the program), we can pass to states q_j, where $j \leq 2i + 1$. The meaning of this restriction complies with a sensible program writing process: starting with a first rule, in each next new rule, we can go either into one of the old states, or into a new one. The fact that the class of computable functions does not get narrowed in so doing is confirmed by the following:

LEMMA 3.1. For any Turing Machine M, there exists a normalized Turing machine M' computing the same function.

Proof. If, in the machine, nonfinal states q_i are renamed into q_j and vice versa, and their corresponding rules in the program are interchanged, then the operation of the machine on any input will not change. Therefore, starting with a first state, we can normalize all rules renaming, if necessary, states into which transitions take place. □

In what follows, we consider only normalized Turing machines, omitting the word "normalized" for brevity. Denote by $\mathcal{P}$ the set of all possible programs of such machines.

LEMMA 3.2. The number of Turing machines of size n is equal to

$$|\mathcal{P}_n| = 4^{2n} \prod_{i=1}^{n} (2i + 2)^2.$$

Proof. For each of the two rules corresponding to a state q_i, there are $2i + 2$ options to choose the state for transiting into another state (including the final one), and there are two record symbol variants and two shift variants. Multiplying these quantities for all n states (twice for each), we obtain the desired result. □

Let M be any Turing machine. Denote by $c(M)$ the set of machines obtained from M by adding any number of new states and assigning arbitrary rules for the new states to the program M, so that the machine remains normalized. All the machines obtained compute the same function as M, since new states are inaccessible on any input, and the machines operate in fact in accordance with the program of M.

LEMMA 3.3. For any machine M, the set $c(M)$ is not negligible.

Proof. Let M have k states. Consider a set of machines $c(M)_{k+n}$ of size $k+n$ with n new states. We count the number of such machines. It is equal to the number of fragments of programs for the new states. For a state q_i, $i = k+1, \ldots, k+n$, we can have $(4(2i+2))^2$ options to choose the appropriate rules. Altogether we obtain

$$|c(M)_{k+n}| = 4^{2n} \prod_{i=k+1}^{k+n} (2i+2)^2.$$

Therefore,

$$\rho(c(M)) = \lim_{n\to\infty} \frac{|c(M)_{k+n}|}{|\mathcal{P}_{k+n}|} = \lim_{n\to\infty} \frac{4^{2n} \prod\limits_{i=k+1}^{k+n} (2i+2)^2}{4^{2(k+n)} \prod\limits_{i=1}^{k+n} (2i+2)^2}$$

$$= \lim_{n\to\infty} \frac{1}{4^{2k} \prod\limits_{i=1}^{k} (2i+2)^2} = \lim_{n\to\infty} \text{const} = \text{const} > 0. \ \square$$

THEOREM 3.4. The set $K_0 \subseteq \mathcal{P}$ of programs is generically complete in the class of recursively enumerable sets of natural numbers.

Proof. Since K_0 is m-complete, it follows that for any recursively enumerable set $A \subseteq \mathbb{N}$ there exists a total computable function $h : \mathbb{N} \to \mathcal{P}$ such that $x \in A \Leftrightarrow h(x) \in K_0$ for any $x \in \mathbb{N}$. A function $f : \mathbb{N} \times \mathbb{N} \to \mathcal{P}$ realizing the generic relation from the set A to K_0 is constructed as follows. Fix a number $x \in \mathbb{N}$. Consider a set $c(h(x)) = \{P_0, P_1, P_2, \ldots\}$, where the programs are enumerated in order of increasing size, and programs of equal size are enumerated in lexicographic order. Put $f(x,n) = P_n$. We show that the function f realizes the generic relation. Properties (1) and (2) in the definition of a generic relation are obviously satisfied. Property (3) follows from Lemma 3.3. Property (4) derives from the m-completeness of K_0 and the fact that all programs in $c(h(x))$ on input 0 operate in the same way as the program $h(x)$. $\square$

Acknowledgments. I am grateful to the referee for useful remarks and suggestions.

REFERENCES

1. I. Kapovich, A. Myasnikov, P. Schupp, and V. Shpilrain, "Generic-case complexity, decision problems in group theory, and random walks," *J. Alg.*, **264**, No. 2, 665-694 (2003).
2. I. Kapovich, A. Myasnikov, P. Schupp, and V. Shpilrain, "Average-case complexity and decision problems in group theory," *Adv. Math.*, **190**, No. 2, 343-359 (2005).
3. J. D. Hamkins and A. Miasnikov, "The halting problem is decidable on a set of asymptotic probability one," *Notre Dame J. Form. Log.*, **47**, No. 4, 515-524 (2006).

4. H. Rogers, *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, New York (1967).

5. S. A. Cook, "The complexity of theorem-proving procedures," *ACM, Proc. 3d Ann. ACM Sympos. Theory Computing* (Shaker Heights, Ohio 1971) (1971), pp. 151-158.

6. C. G. jun. Jockusch, and P. E. Schupp, "Generic computability, Turing degrees, and asymptotic density," *J. London Math. Soc., II. Ser.*, **85**, No. 2, 472-490 (2012).

7. G. Igusa, "The generic degrees of density-1 sets, and a characterization of the hyperarithmetic reals," *J. Symb. Log.*, **80**, No. 4, 1290-1314 (2015).