A Remark on Discovery Algorithms for Grammars

E. Shamir*

The Hebrew University, Jerusalem, Israel

I. INTRODUCTION

For the class of all finitely presented groups (i.e., groups defined by a finite list II of generators and relations between them), the problem, whether the group $G_n$ presented by II is cyclic, is recursively unsolvable. However, there is an algorithm which, when applied to II, will lead to a presentation of a cyclic group isomorphic to $G_n$, provided such a cyclic group exists. This algorithm, which is based on Tietze's transformations, recursively enumerates, for a given II, all the representations of groups isomorphic to $G_n$, so that if $G_n$ is cyclic, we shall eventually arrive at a suitable representation (with free generators and no relations). It follows that we may replace cyclicity by many other group-theoretic properties, which are known to be recursively unsolvable (cf. Rabin, 1958).

In this note, we prove a general result which shows that, for various classes of grammars (or automata), an analogous alogorithm does not exist. Such an algorithm should lead from a grammar $G_1$ which belongs to a class $C_1$ to an equivalent grammar $G_2$ of class $C_2 \subseteq C_1$, provided such a $G_2$ exists. This may be called a "discovery algorithm" for $C_2$ relative to $C_1$, since it is able to discover a grammar $G_2$ of class $C_2$, for a language presented by a grammar $G_1$ of class $C_1$, provided such a $G_2$ exists. Since the grammars of $C_2$ will in general be of a simpler kind, we may regard this also as a simplification procedure.

II. DEFINITIONS

Let us first explicate precisely the concepts of grammar and discovery algorithm.

A grammar $G$ is a finite device which classifies every string over a

* This work was supported by the U.S. Office of Naval Research, Information Systems Branch, under Contract N 62558-2214.
finite set of symbols $\Sigma$ (the alphabet) as accepted or rejected. The set of accepted strings is denoted by $L(G)$ and is called the language presented by $G$. For present purposes, we also require $G$ to be a recursive device, in the sense that $L(G)$ is a recursive set. Two grammars $G_1$ and $G_2$ are equivalent if $L(G_1) = L(G_2)$. If we start from a language $L$, then a grammar $G$ such that $L = L(G)$ is a presentation of the language $L$.

Classes of grammars over a common alphabet $\Sigma$ are characterized by constructional specifications on the grammars of the class. We shall consider in particular the class of finite automata (FA) (Rabin and Scott, 1959; Bar-Hillel and Shamir, 1960) or finite state grammars (Chomsky, 1957; Chomsky and Miller, 1958; Chomsky, 1959), the class of context-free phrase-structure grammars (CF) (Chomsky, 1959; Bar-Hillel et al., 1961), and the class of context-sensitive phrase-structure grammars (Chomsky, 1959, 1963).

Let $C_1$ and $C_2$ be two classes of grammars with $C_2 \subseteq C_1$. A discovery algorithm for $C_2$ relative to $C_1$ is a Turing machine which, given a grammar $G_1 \in C_1$, will compute a grammar $G_2 \in C_2$ equivalent to $G_1$, in case such a $G_2$ exists. If we introduce Gödel numbering, we may alternatively define a discovery algorithm as a partial recursive function $f$ which is defined for all numbers $m$ which are Gödel numbers of a $G_1 \in C_1$ so that, for all such $m$, $f(m)$ is the Gödel number of an equivalent grammar $G_2 \in C_2$.

III. THE MAIN THEOREM

Before stating the main theorem, let us note that if $C$ is any class of grammars (by a class we always mean a countable class) and if $G_0 \in C$, then the set of the $G \in C$ which are not equivalent to $G_0$ is recursively enumerable. Indeed, we can enumerate all the strings over $\Sigma$ in a sequence $x_1, x_2, x_3, \ldots$. For a given $G \in C$, we start testing whether $x_1, x_2, \text{etc.}$, are accepted by $G_0$ and $G$. If $L(G) \neq L(G_0)$, we shall ultimately arrive at a string which is accepted by the one and rejected by the other. If we also enumerate the $G \in C$ in a sequence $G_i$ and then arrange the pairs $(x_i, G_i)$ in a single sequence and in this order perform the tests $x_i \in L(G_i)$, then we obtain in fact a recursive enumeration of all grammars not equivalent to $G_0$.

A property $P$ of grammars, which is preserved under equivalence, is called a language-property, since it depends on the language only and not on the specific grammar presenting it.
THEOREM. Let $C_1$ and $C_2$ be two classes of grammars, $C_2 \subset C_1$. Let $P$ be a language-property such that

(a) the set of grammars in $C_1$ for which $P$ does not hold is recursively enumerable but not recursive (which implies that the property $P$ is recursively unsolvable for $C_1$),

(b) if $P$ holds for $G_1$, then $G_1$ is equivalent to a grammar $G_2 \in C_2$,

(c) the property $P$ is recursively solvable for $C_2$, i.e., the set of grammars $G \in C_2$ for which $P$ holds is recursive.

Then there does not exist a discovery algorithm for $C_2$ relative to $C_1$.

PROOF: If there exists a discovery algorithm, then, given $G_1 \in C_1$, the following algorithm will decide the property $P$ for $G_1$:

(i) Enumerate the grammars for which $P$ does not hold and, at the same time,

(ii) apply the discovery algorithm for $G_1$.

If $P$ does not hold for $G_1$, (i) will terminate. If $P$ holds for $G_1$, then by (b), $G_1$ is equivalent to a $G_2 \in C_2$, and then (ii) will terminate, yielding a grammar $G_2 \in C_2$ for which we can settle by (c) that $P$ holds.

But, by (a), $P$ is not recursively solvable for $C_1$, hence a discovery algorithm for $C_2$ relative to $C_1$ cannot exist.

IV. EXAMPLES AND REMARKS

We now turn to some immediate applications of the Theorem.

Example 1. Let $C_1$ be the class CF, $C_2$ the class FA. Every FA can easily be presented by a CF grammar so that $C_2 \subset C_1$. Let $P$ be the property: The complement of $L(G)$ with respect to $\Sigma^*$, the set of all strings over $\Sigma$, is empty. Then (a)-(c) are fulfilled. Indeed, if $\Sigma^* - L(G)$ is empty, then $L(G) = \Sigma^*$, which is easily presented by a FA $\bar{G}$. That $P$ does not hold for $G$ means then that $G$ is not equivalent to $\bar{G}$, and it was noted above that the set of such grammars $G$ is recursively enumerable. Finally, the property $P$, viz., whether $\Sigma^* - L(G)$ is empty, is recursively unsolvable for CF (Bar-Hillel et al., 1961) but solvable for FA (Rabin and Scott, 1959).

Thus we have:

Corollary 1. There does not exist a discovery algorithm for FA relative to CF.

It might be useful to repeat the argument of the proof for this case: If there existed a discovery algorithm, then in order to decide whether $\Sigma^* - L(G)$ is empty, one would have to test whether $\Sigma^* - L(G)$ is not empty (until a string $x \notin L(G)$ appears) and at the same time apply
the algorithm to $G$ which, in case $\Sigma^* - L(G)$ is empty, would yield a FA $G_2$ of which we would know that $\Sigma^* - L(G_2)$ is empty.

Remark 1. In Example 1, we can take $C_2$ as the class of sequential grammars (Ginsburg and Rice, 1961; Shamir, 1961), which is a subclass of CF but has decision (and many other) properties analogous to the superclass. For $C_2$ we may also take other quite restricted subclasses of CF which have the required decidability (or rather undecidability) properties.

Example 2. $C_1 =$ the class of context-sensitive grammars, $C_2 = C$. Let $P$ be: $L(G)$ is empty (in other words, $L(G)$ is equivalent to a FA presenting the empty set). Then $P$ is recursively unsolvable for $C_1$ (Chomsky, 1963), but solvable for $C_2$ (Bar-Hillel et al., 1961). The other requirements in (a)-(c) follow easily and we have:

Corollary 2. There does not exist a discovery algorithm for CF relative to context-sensitive grammars.

Remark 2. In Example 2 we may take $C_1$ as the class of grammars given by intersection of two CF. The emptiness problem for $C_1$ is recursively unsolvable (Bar-Hillel et al., 1961). (This incidentally is one of the methods of proving that the emptiness problem for context-sensitive grammars is unsolvable.)

Remark 3. In all the examples cited above, the problem whether a grammar of $C_1$ has an equivalent grammar in $C_2$ is recursively unsolvable (Bar-Hillel et al., 1961); however, as the example of the finitely presented groups shows, this fact in itself does not imply the nonexistence of a discovery algorithm for $C_2$ relative to $C_1$. Conversely, it may happen that a discovery algorithm does not exist although the problem whether a member of $C_1$ has an equivalent (isomorphic) member in $C_2$ is solvable, but we were unable to find a “natural” example for this.

Remark 4. The property $P$ which we used in the examples is of the form: $G$ is equivalent to a special fixed (“total” or “empty”) FA and we could choose for $C_2$ any class which contains this fixed grammar and for which the problem of the equivalence with this fixed grammar is solvable. Of course, one may take as $C_2$ the class containing as its only member this fixed grammar (or any other fixed grammar $G_2$ for which the problem whether $G$ is equivalent to $G_2$ is unsolvable in the class $C_1$). Then the result simply means that there exists no recursive enumeration of the grammars of $C_1$ equivalent to $G_2$. In other words, for grammars in general there is no chain of simple computable steps (like the steps based on Tietze’s transformation for groups) which links any two equivalent
grammars, a fact which shows that the notion of equivalence between grammars, based on equality of languages alone, is not very useful in the study of grammars and languages. For that purpose, equivalences of stronger kinds such as discussed by Gaifman (1961) and Chomsky (1963) should be introduced.

The above results may also add some weight to the view which has been specially emphasized by Chomsky (1957, 1963) that the quest for a discovery method for grammars of natural languages (as a task of general linguistics) is unrealistic.

Indeed we see (for artificial grammar models) that if a language is presented by a grammar which is (if only "slightly") more complex than necessary (a fact we might know from some general consideration), there does not exist a general method (otherwise it does not belong to general linguistics) for discovering the simpler grammar.

Sometimes, by a discovery method an algorithm is meant which constructs the grammar with the help of a "teacher" or an "oracle" that tells us, for any string, whether it is accepted or not (Solomonoff, 1960). In other words, we do not use the internal structure of the oracle (which could be, for instance, a more complex grammar) but only the characteristic function of the language. But clearly a discovery method in such a strong sense cannot exist for classes of grammars which may present infinite languages, since the discovery method can use in each case a finite set of strings only, and all the languages which differ from the given one outside this finite set would have the same grammar, which is absurd.

Finally, let us note that there are many practical situations in which there arises a demand for simplification algorithms. The well-known problem of simplifying certain kinds of networks, for example, is essentially a problem of this type with respect to FA. In fact, the problem there lies in obtaining efficient, rather than just effective algorithms. The demonstrable nonexistence of a complete simplification algorithm may be an indication that the problem of obtaining partial (heuristic) simplification procedures is a hard one to tackle.

Received: July 30, 1962

References


