STRUCTURED-STORAGE AFA

Armen Gabrielian
Seymour Ginsburg
University of Southern California

Abstract

Among the various types of structures used for storing and representing data there are numerous nonlinear types, e.g., trees (Samuel, 1959), linked structures (Newell and Simon, 1956), and n-dimensional arrays (Hartmanis and Stearns, 1965); nevertheless, most theoretical models of computation adhere to the tape model of storage structure, e.g., Turing acceptors, pushdown acceptors, and stack acceptors. Customarily, the nonlinear types are linearized so that the information stored in them is represented on a tape. However, this representation may be unsuitable for various reasons. For example, in the case of linear coding of organic compounds, difficulties arise with unique representability and decodability. Perhaps the most serious disadvantage of linearization is that it complicates the performance of naturally and structurally dependent operations on the data.

In recent years much of the research on formal languages has been within the framework of AFL theory (Ginsburg and Greibach, 1969), i.e., the study of certain families of formal languages called "AFL". Similarly, much of the research on accepting devices has been made within the framework of AFA theory (Ginsburg and Greibach, 1969), i.e., the study of certain families of one-way, nondeterministic, single storage tape acceptors called "AFA". It is known that, in a natural manner, AFA give rise to AFL and, conversely, AFL give rise to AFA. In AFA theory, several alternatives to the single tape storage structure of acceptors have been considered. For example, nested multitape AFA are used in Greibach and Ginsburg (1972) to give a device-characterization of the substitution of one full AFL into another. A generalization of this is used in Greibach (1970) to give a characterization of nested iterated substitution-closed full AFL. In each of these cases, linearization is employed to show that certain families of acceptors with generalized storage structure are equivalent, from the point of view of languages defined, to ordinary AFA, i.e., the single-tape kind. The purpose of this paper is to introduce a class of families of acceptors called "structured-storage AFA" (abbreviated "SS-AFA") in which the auxiliary storage is of a very general form. Indeed, the auxiliary storage is general enough so that SS-AFA include ordinary AFA and different kinds of multitape AFA, as well as various families of acceptors obtained by imposing geometrical or graphical structures on the auxiliary storage.

The paper is divided into three sections. Section 1 introduces SS-AFA. Section 2 establishes the basic fact that, from the point of view of defining languages, SS-AFA are equivalent to ordinary AFA. (This result eliminates the need to employ the linearization process each time a new SS-AFA is encountered.) Section 2 also examines the effect of imposing certain finiteness conditions on the storage structure of SS-AFA. Section 3 considers a variety of different examples of SS-AFA, each of which either is already in the literature or possesses a storage structure of some prominence. In view of the numerous special instances of SS-AFA, it is reasonable to expect that the placing of appropriate restrictions on SS-AFA will be a useful tool in obtaining new classes of acceptors and corresponding families of languages.

References

Ginsburg, S. and S.A. Greibach (1969), "Abstract Families of Languages", in Studies in Abstract Families of Languages, Memoirs of the American Mathematical Society, No. 87, 1-32.

Greibach, S.A. (1970), "Full AFL's and Nested Iterated Substitution", Information and and Control, 16, 7-35.

Greibach, S.A. and S. Ginsburg (1972),"Multitape AFA", J. A.C.M., to appear.

Hartmanis, J. and R.E. Stearns (1965), "On the Computational Complexity of Algorithms", Trans. Amer. Math. Soc., 117, 285-306.

Newell, A.P. and H.A. Simon (1956), "The Logic Theory Machine: A Complex Information Processing System", IRE Trans. IT, 2, 61-79.

Samuel, A.L. (1959), "Some Studies in Machine Learning Using the Game of Checkers", IBM J. Research and Development, 3, 211-229.