GRAMMATICAL INFERENCE FOR DEFINING EXTENSIONS

S. Crespi Reghizzi
Istituto di Elettrotecnica ed Elettronica - Politecnico
32, P.za Leonardo da Vinci - Milan 20133, Italy.

Extensions to a language need be defined by a grammar, or by any conceivable method which is apt to discriminate valid from unwanted extensions and to provide efficient parsing and translation.

Context-free grammars - a traditional choice - have a major drawback with respect to practical use. Namely, the unsophisticated user of an extensible system would find it hard to formulate an appropriate BNF definition of the constructions he wants.

It is likely that the user would prefer to define by examples the extensions he has in mind. In an interactive environment the system should be able to

a) extract from the examples some principles of well-formedness of the extensions, e.g. in the form of BNF rules to be submitted to the user for approval.
b) Modify the rules derived in a), in order to account for new examples supplied by the user and specific statements of what is, or is not, a valid extension.

The nucleous of such a definition facility could be provided by a grammatical inference algorithm, described in Biermann and Feldmann [1] and Crespi-Reghizzi [2] [3].

In the two latter works, data to the algorithm are bracketed sentences such as

$$\underline{a} + \underline{\underline{a} + \underline{a}} \quad , \quad \underline{\underline{(\underline{a} + \underline{a})} + \underline{a}} \quad , \quad \text{etc.}$$

where underscores display the structure to be assigned to the language. The grammatical inference algorithm [2] construct an operator precedence grammar which 1) generates the sentences of the sample with the specified structure; 2) generates an infinite number of sentences not in the sample, which are a "reasonable" generalization of the given examples.
As new examples are added to the sample, the grammar inferred by the algorithm is incrementally updated. Certain properties of convergence of the algorithm are also considered in the quoted references. The state of the art of grammatical inference is the following.

There exists a satisfactory algorithm for finite-state languages [4] which is actually running.

The program [2] for operator precedence grammars is very fast, by has some restrictions (not discussed here) on the class of grammars.

An improvement over [2] is presented in [3] but has not been tested.

No matter which algorithm is chosen, the inferred grammar should be submitted to the user for approval, either directly, or by generating a representative set of strings which the user examines. The capability of taking other information from the user, as suggested in b), should be added to an inference algorithm for directing the generalization and increasing the speed of convergence to the language intended by the user.

A proposal for the use of inference techniques in the design of programming languages is presented in [5].

REFERENCES

[1] Biermann, A.W. and Feldmann J.A., "A survey of results in grammatical inference", Internat. Conf. on Frontiers in Pattern recognition, Honolulu, 1971.

[2] Crespi-Reghizzi, S., "An effective model for grammar inference", Proc. IFIP Congress, Ljubjana, 1971.

[3] Crespi-Reghizzi, S., "Reduction of enumeration in grammar acquisition", Proc. 2nd Joint Conf. Artificial Intelligence, London, 1971.

[4] Biermann, A.W. and Feldmann, J.A., "On the synthesis of finite-state acceptors", A.I. Memo N. 114, Computer Science Dept., Stanford University, April 197L.

[5] Crespi-Reghizzi, S., Melkanoff, M.A. and Lichten, L., "A proposal for the use of grammar inference as a tool for designing programming languages", Rept. N. 71-1, Istituto di Elettrotecnica ed Elettronica, Politecnico di Milano.