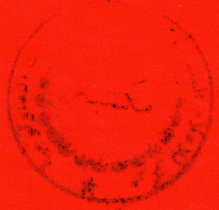
Séminaires IRIA



LANGAGES ET TRADUCTEURS

1977

IRIA

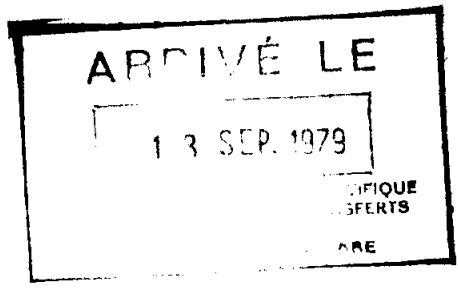
INSTITUT DE RECHERCHE D'INFORMATIQUE ET D'AUTOMATIQUE

DOMAINE DE VOLUCEAU - ROCQUENCOURT - B.P. 105 - 78150 LE CHESNAY - TÉL.: 954 90 20

LANGAGES ET TRADUCTEURS

TEXTES DES EXPOSÉS DU SÉMINAIRE ORGANISÉ PAR
L'INSTITUT DE RECHERCHE D'INFORMATIQUE ET D'AUTOMATIQUE
(IRIA)
ROCQUENCOURT

1977



INSTITUT DE RECHERCHE D'INFORMATIQUE ET D'AUTOMATIQUE
DOMAINE DE VOLUCEAU - ROCQUENCOURT - B.P. 105 - 78150 LE CHESNAY - TÉL.: 954 90 20

Édité par l'**Institut de Recherche d'Informatique et d'Automatique**

Dépôt légal 310378 / 200

I.S.B.N. 2 - 7261 - 0159 - 3

TABLE DES MATIÈRES

AUTOMATIC SYNTACTIC ERROR RECOVERY FOR LR-PARSERS

Pierre BOULLIER

IRIA

ABSTRACT

The method we shall describe is a practical scheme for performing either correction of or recovery from syntactic errors. Though this process is automatically deduced from the formal definition of the programming language, it can be tuned somewhat by the system's user. Its implementation does not require any extra table in addition to the parsing one. Comparisons with various existing methods show that the scheme to be described usually provides an improvement in the quality of the error process as well. This method has been used in the error processing module of a production compiler for SIMULA 67.

RÉSUMÉ

La méthode que nous allons décrire effectue de façon pratique, après chaque erreur de syntaxe, soit une correction, soit une récupération. Bien que le procédé soit déduit automatiquement de la définition formelle du langage, l'utilisateur du système a la possibilité d'ajuster, dans une certaine mesure, les paramètres du module traitant les erreurs. Son implémentation ne nécessite aucune table autre que celle d'analyse. Des comparaisons effectuées avec d'autres méthodes montrent que le schéma décrit améliore en général la qualité du traitement d'erreur. Cette méthode a été utilisée dans un compilateur industriel pour SIMULA 67.

1. Introduction

Error detection and recovery are needed in every translator. In the design of a Translator Writing System it seems natural to consider an automatic error recovery scheme which would relieve the system's user from the burden of dealing with designing, writing and debugging a multitude of adequate recovery routines.

The error processing module of a translator must supply the programmer with maximum information about detected errors gathered during the translation process in order to minimize the effort to achieve a syntactically correct program. The compiler must be able to continue its translation process (or at least the parsing part) after each error as if the program were syntactically valid.

A "good" error recovery mechanism must have certain qualities among which one might mention :

a) It should find the maximum number of syntactic errors in one pass, with each error being detected as soon as possible.
b) The message provided for each error must be clear and guide the programmer towards the correction ; the action taken by the recovery mechanism must be explicit.
c) It should be relatively efficient, both in space and in time ; no overhead should be incurred for correct programs or for correct portions of incorrect programs.
d) It must minimise the number of times it reports an error when there is none (these spurious error detections are usually caused by an inappropriate recovery action).
e) The recovery process must be generated automatically from the formal syntactic definition of the language.

In § 2, the philosophy and architecture of our system and the notions of detection and recovery are presented.
§ 3 presents the LR-parsing technique upon which our system is based and which allow appropriate detection ; this technique provides all the information needed for our recovery process. The error processing mechanism combines two methods, depending on the kind of error : a local correction (§4) of the source text is first attempted ; if that try fails, a global recovery (§5) is then undertaken. Error messages are described in §6. Finally in §7 our method is compared, by means of some examples, with both another automatic error recovery system and the hand-tailored error recovery mechanism of a production compiler.

2. Philosophy of this work

2.1 System architecture

We assume that the erroneous source programs may be divided into two classes :

- the source texts which differ from valid ones in such a way that local corrections around the terminal symbol in error would give a syntactically correct string.
- the erroneous source texts which do not belong to the first category.

After a syntactic error is detected we shall first try a local correction around the error point so as to transform the erroneous source string into a valid one. In order to achieve this goal, it is necessary that the terminal symbol where the error is detected be the first symbol for which the source text no longer belongs to the set of all valid prefixes of the strings of the language generated by the production rules. If that local correction does not work (the source text is "too far" from a valid one or the detection point is after the actual point in error) then we perform a global recovery which tries to detach the shortest part of the source text around the point in error in order to replace it by a valid one.

2.2 Error detection

Detection is the discovery of constructs which are not allowed according to the syntactic definition of the language. Implementation of such a process is