Journal of Automata, Languages and Combinatorics **4** (1999) 2, 77–86

ABSTRACT COMPARISON OF TWO ERROR DETECTION TECHNIQUES FOR PROGRAMMING LANGUAGES

EBERHARD BERTSCH
Ruhr University,
Universitätsstr. 150, D-44780 Bochum, Germany
e-mail: bertsch@lpi.ruhr-uni-bochum.de

ABSTRACT

Two fundamentally different ways of diagnosing grammatical errors in programs are defined and compared. It is shown that for certain languages the sets of error points found by the two techniques are essentially different. Furthermore, one technique will quite generally lead to fewer error positions in given programs than the other technique. It is suggested, but obviously not proved, that the notion of a *spurious* error is related to the discussion in this article.

Keywords: syntax error recovery, parsing.

1. Introduction

Insertion-only error repair as studied in [6, 9, 11] and non-correcting error recovery as introduced by RICHTER [16] are entirely different approaches to the task of detecting syntactic errors in the process of parsing a given program text. The purpose of the present paper is to compare those two concepts on a sufficiently abstract level, using similar notation. Both techniques can be discussed for a given language without reference to the particular grammar being used in the parsing process.

Insertion-only repair can be summarized informally as follows:

An input string of tokens is read from left to right. If some token does not form a valid continuation of the previous text, one or more additional tokens are inserted in front of it. The choice of this inserted string is done in such a way that the previous text, the inserted text and the offending token together are a valid initial part (prefix) of some program.

Variants of the technique use a certain fixed amount of lookahead to decide what should be inserted. Whenever another error is detected further right, another attempt at inserting one or more tokens is made. It must be noted that this procedure will not always find a suitable insertion. In the PASCAL language, for example, no nonempty string can be inserted in front of the *program* keyword. Leaving aside the possibility of deleting symbols in such cases, one suggestion [11] is to complete the current program