A PROGRAM STRUCTURE FOR ERROR
DETECTION AND RECOVERY

J.J. Horning
University of Toronto, Canada

H.C. Lauer, P.M. Melliar-Smith, B. Randell
University Newcastle upon Tyne, England

ABSTRACT

The paper describes a method of structuring programs which aids the
design and validation of facilities for the detection of and recovery from
software errors. Associated with the method is a mechanism for the automatic
preservation of restart information at a level of overhead which is believed
to be tolerable.
1. Introduction

Prior research into reliable computing has concentrated on the reliability of the hardware, on the detection of hardware errors, and on the configuring of systems to allow continuation of service in the presence of hardware errors. But observation of present-day large systems indicates that software faults represent a problem whose significance is at least as great as that of the hardware faults. Whilst conceding the importance of current research on improving the quality of software (e.g. work on program "correctness proofs"), the present paper is based on the view that it is also worth providing error detection and recovery facilities for both hardware and software errors. In what follows we will concentrate on software errors although we believe much of our work is of equal relevance to many types of hardware errors.

This paper describes the recovery block concept, a method of structuring programs which is aimed at aiding the design of error detection and recovery facilities, and the recursive cache, an associated mechanism which provides means for automatic "back-tracking" at a level of overhead which is believed to be tolerable.

2. Error Detection and Recovery

Reliable operation of a computing system depends on both error detection and error recovery. Several classes of error detection techniques are available, some of which operate automatically on an instruction-by-instruction basis, while others take a broader view of correct operation based on programmed checks and assertions. It is characteristic of error detection that several techniques can readily be used together, and the recovery block concept aims to provide error recovery after any kind of detected error.

Recovery and restart of the program after error detection is a difficult problem. Where instruction-by-instruction error detection is provided, the number of possible errors is too great to provide explicit recovery action for each possible case, while any automatic recovery operation which simply aims to repair the state of the program so as to allow its continuation in a valid manner cannot be expected to achieve correctness rather than mere legality. Alternatively, on detection of an error by a programmed check, the number of possible ways in which the program may have erred is very large and obscure side-effects of the error may have spread into the system. The analysis, with certainty, of such an erroneous program state is in general beyond our capabilities, and therefore repair of the program state cannot be recommended.

For many applications, frequently the applications requiring the highest reliability, error recovery by repetition of whole job steps or other major program units must imply a substantial recovery overhead and a degradation of service. Such applications require a recovery mechanism which can achieve local recovery from an error whenever possible.

It is characteristic of error recovery operations that they are very much more error prone than the main programs, and are very difficult to check. It is an objective of the recovery block concept that the error recovery actions should be testable and that they should be checked with the same rigour as the main program. It is of course obvious that repetition of the operation with the same program will not always achieve recovery from a program error.