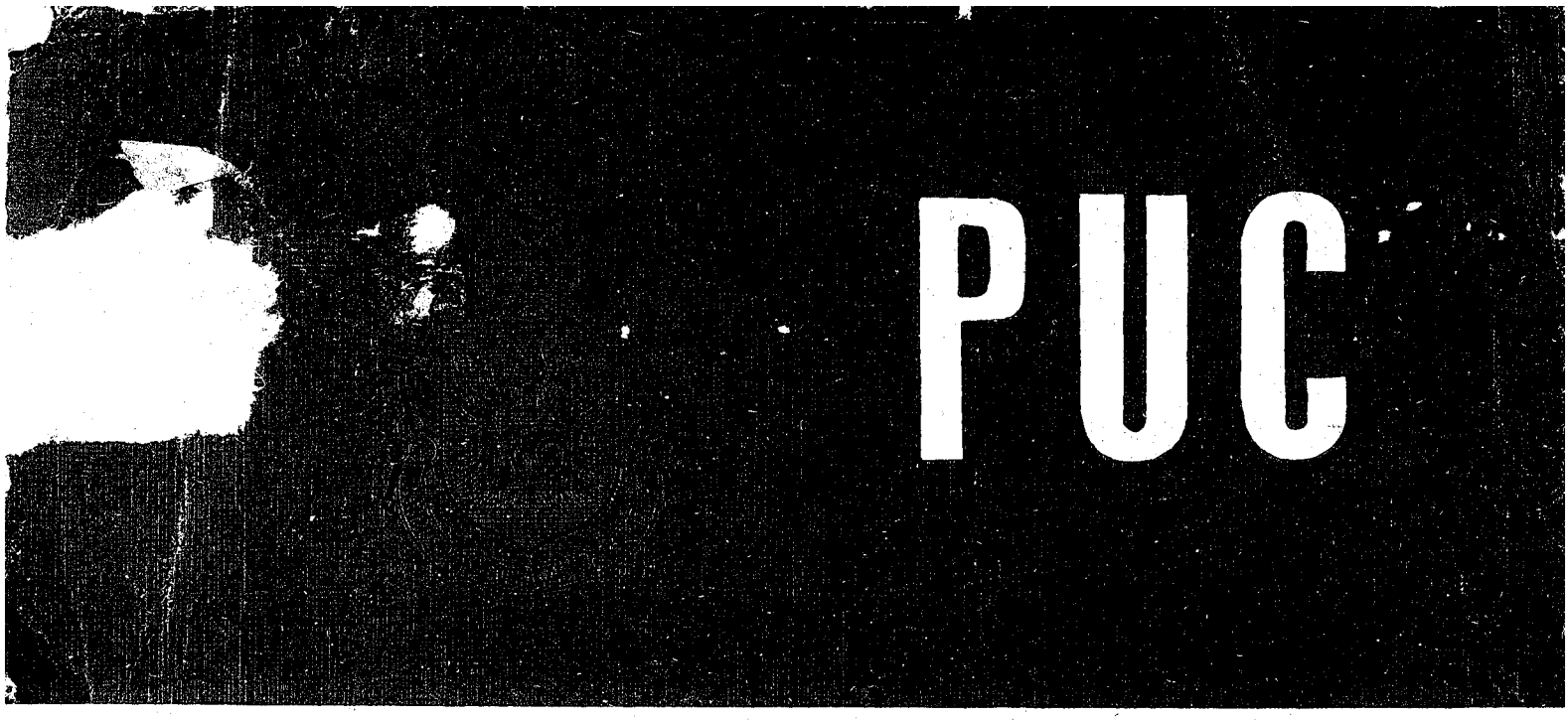

PRE-EDIÇÃO

SIMPOSIO INTERNACIONAL SOBRE METODOLOGIAS PARA O PROJETO E CONSTRUÇÃO DE SISTEMAS DE SOFTWARE E HARDWARE

PATROCINADOR

CONSELHO NACIONAL DE DESENVOLVIMENTO CIENTIFICO E TECNOLOGICO

28 DE JUNHO A 2 DE JULHO DE 1976

RIO DE JANEIRO
BRASIL

Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 209 — ZC-20
Rio de Janeiro — Brasil

PREPRINTS

INTERNATIONAL SYMPOSIUM ON METODOLOGIES FOR THE DESIGN AND CONSTRUCTION OF SOFTWARE AND HARDWARE SYSTEMS

SPONSOR : CONSELHO NACIONAL DE DESENVOLVIMENTO CIENTIFICO E TECNOLOGICO

CO-SPONSORS: CANADIAN INTERNATIONAL DEVELOPMENT AGENCY
GESELLSCHAFT FUR MATHEMATIK UND DATENVERARBEITUNG
NATIONAL SCIENCE FOUNDATION
IBM DO BRASIL

ORGANIZING COMMITTEE:

PROFESSORS

C.J.P. LUCENA
W. BRAUER
N. MACHADO
A.L. FURTADO

DATES

JUNE 28 TO JULY 2, 1976

NOTE

THESE PREPRINTS HAVE BEEN EDITED. THE AUTHORS ARE KINDLY INVITED TO REQUEST ANY CORRECTIONS OR CHANGES FOR THE FINAL EDITION.

EXCEPTION HANDLING*

Arndt von Staa
Departamento de Informática
PUC-RJ

An exception is a detected instance of an exception condition. For example, overflow is an exception condition, whereas the occurrence of overflow is an exception. Exceptions cause the eventual, not necessarily synchronous, activation of an exception handler.

Exceptions may be signalled by hardware, e.g. overflow, end-of-reel, machine-error, or by software, e.g. subscripts out of bounds, lack of convergence.

Though exceptions are quite complex to handle several reasons lead us to their appreciation:

a. programs or sub programs may be interrupt driven, e.g. interactive systems, end of page handlers.

b. some operations fail, this failure determining further action, e.g. detection of end of reel during a write operation.

c. certain problems may be solved by means of several incomplete solution methods, e.g. symbolic integration.

d. machines may fail and adequate actions must be taken to prevent major damages, e.g. I/0 errors, loss of energy supply.

e. programs may fail possibly due to unnoticed incorrectnesses, e.g. subscripts out of hounds, overflow, deadlocks.

An exception descriptor is a module which defines the detection information and the detection condition of an exception. It defines also the exception handler and the parameters, exception information, which this handler requires in order to take adequate actions.

In a modular programming environment exception descriptors are defined within a module. The detection information defined by the exception descriptor must be declared within the module containing this exception descriptor - containment requisite. If not so, semantic context independence cannot be assured - due to unforeseen modules detecting the exception - and/or syntactic

non interference cannot be assured - due to the need of specific linkages between the module containing the descriptor and the module containing the detection point. Furthermore, due to the containment requisite and to the ability of transmitting type descriptors, we may now transmit exception descriptors completely transparent to the user of the module containing it.

The exception information must be made available at the module where the exception was detected. In some cases this information may not be available, e.g. statement number, thus there must be a convention by means of which exception information is consistently initialized to undefined and only defined when the module containing the detection point can make this information available.

Exception handlers may allow the preempted module to recover from the exception. This may be achieved by

a. simply returning back to the detection point;

b. by returning a value to be as result of the operation which caused the exception;

c. by transfering control to some predefined and well known retrial point.

Exception handlers may also pass the exception on to other handlers, or then cause the termination of one or more active modules.

Exception handlers which allow the resumption of the preempted module must be logically correct in the sence that the correctness of this module is not affected by the resumption.

Exceptions may be time dependent, e.g. the exception information may be volatile. There is a critical interval associated with exceptions, which, if exceeded causes an overrun exception to be detected.

Exceptions may require immediate attention, or may be posted for eventual attention or no attention at all.

Exceptions require priority in order to allow exceptions signalled by a handler to preemt this handler whenever

REFERENCES

[Floyd 67] Floyd, R.W. "Assigning Meanings to Programms"; Proceedings of a Symposium in Applied Mathematics, vol. 19; American Mathematical Society 1967; pg 19-32.

[Horning et ali 74] Horning, J.J.; Lauer, H.C.; Melliar-Smith, P.M.; Randell, B. A Program Structure for Error Detection and Recovery; University of New Castle upon Tyne TRS-59, Apr 1974.

[Rosenberg 75] Rosenberg, P.A. On Demons, mousetraps and breakpoints, UCLA, Internal Report, Computer Science Department, 1975.

[Staa 74] v. Staa, A. Data transmission and Modularity Aspects of Programming Languages; University of Waterloo, Computer Science Department CS-74-17; 1974.