ALEPH 90-125
SOFTWR 90-13
Ll.Garrido, G. Musolino
August 90

A new analysis tool for JULIA/ALEPHLIB

LLuís Garrido[(a)]*, **Giovanni Musolino**[(b)]
(a) Laboratori de Física d'Altes Energies
Universitat Autònoma de Barcelona
E-08193 Bellaterra (Barcelona), Spain
(b) Istituto di Fisica
Università di Trieste
Via A. Valerio 2
I-63177 Trieste, Italy

August 1990

Abstract

We have developed a new tool to give interactively the tree structure, the input and output banks for a given subroutine of JULIA and ALEPHLIB, and the program modules of JULIA. The user can make new requests using the provided PROLOG interface.

1 Introduction

We have developed a program written in the artificial intelligence language PROLOG that permits in a natural way to find the input and output banks of all the subroutines present in JULIA and ALEPHLIB, and their tree call structure. We define an input module given by the bank 'bank-name' as the subroutine or set of subroutines that use 'bank-name' as input bank and none of the subroutines above in the tree structure use the same bank as input. An analog definition is used for an output module. The package we present is able to find these kind of modules.

This paper is organized in two sections. In the first section we describe the set of programs to create the input files needed in the interactive session. In the second part we describe how to run the interactive part and the commands available.

* Now at CERN

Normal users of this package can skip the next section. This section is intended only for users that want to create new input files for the interactive session because the changes on JULIA and ALEPHLIB are so important that the input files will be seriously affected.

We want to mention that this package can be used to analyze any fortran file where the BOS banks are used in the same way as in JULIA.

2 Generation of the input files for the interactive session

In this section we discuss how to generate the input files for the interactive session. All the files needed for this purpose can be found on GARRIDO's 191 disk (CERNVM). These files are: protree.exec, promod.exec, mod.for, mod.pro, mod.psv, tree.psv and tree.pro. The input files, the code source for JULIA and ALEPHLIB, can be found normally on the PUBXU 409 disk (CERNVM).

Two input files are needed for the interactive session: tree.out and mod.out. The exec file "protree.exec" is used to generate the first file while "promod.exec" is used for the second one.

Due to the big amount of memory needed, our program can not run with an input file that is longer than 60000 lines. Then it is necessary to split the files containing the fortran code of JULIA and ALEPHLIB in two or more parts, taking care of not cutting a subroutine in the middle.

Protree.exec has to run over these files containing the JULIA and ALEPHLIB fortran code. All the outputs obtained have to be collected in a single file: tree.out. This file will contain the following PROLOG facts:

- list-tree("calling-subroutine","called-subroutine") → ;

  that gives information about the calling struture of the programs

- list-in-banks("subroutine-name","bank-name") → ;

  that gives information about the banks that are used as input for a given subroutine.

- list-out-banks("subroutine-name","bank-name") → ;

  that gives informations about the banks that are used as output for a given subroutine.

Promod.exec uses as input file tree.out and creates the file mod.out. This file will contain the following PROLOG facts:

- list-in-module("subroutine-name","bank-name") → ;

  that gives the subroutine names that can be considered as input modules for a given bank, following the definition given in the introduction.

- list-out-module("subroutine-name","bank-name") → ;

  that gives the subroutine names that can be considered as output modules for a given bank.

3 User manual for the interactive session

The starting of the interactive session is done by executing the file julmod.exec that can be found on GARRIDO's 191 disk (CERNVM). This exec file executes a prolog program (julia.pro) that permits to the user the execution of the following commands (notice that each command has to be written in lower case and followed by the special character " ; " , and all the subroutine and bank names have to be between double quotes) :

- **help;**

  gives the list of command available.

- **julia-version;**

  gives the number of the Julia version from which the information have been obtained.

- **calls("calling-subroutine");**

  gives the list of the subroutines that are called by "calling-subroutine".

- **called-by("called-subroutine");**

  gives the list of the subroutines that call "called-subroutine".

- **in-banks("subroutine-name");**

  gives the names of the BOS-banks used as input in "subroutine-name".

- **out-banks("subroutine-name");**

  gives the names of the BOS-banks used as output in "subroutine-name".

- **used-banks("subroutine-name");**

  gives the names of all the BOS-banks used in "subroutine-name" and their usage (input, output).

- **in-sub("bank-name");**

  gives the names of the subroutines that use the BOS-bank "bank-name" as an input bank.

- **out-sub("bank-name");**

  gives the names of the subroutines that use the BOS-bank "bank-name" as an output bank.

- **used-sub("bank-name");**

  gives the names of all the subroutines that use the BOS-bank "bank-name" and the usage (input, output). and which as output

- **in-mod("bank-name");**

  gives the names of the modules that use the BOS-bank "bank-name" as input.

- **out-mod("bank-name");**

  gives the names of the modules that use the BOS-bank "bank-name" as output.

- **used-mod("bank-name");**

  gives the names of the modules that use the BOS-bank "bank-name" and the usage (input, output).

- **quit;**

  to end the session.

New commands can be created by the user using PROLOG in the same way as it is done in the file julia.pro .