UNIVERSIDADE DE SÃO PAULO

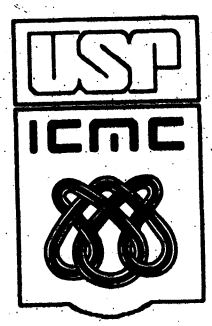
Instituto de Ciências Matemáticas e de Computação

DOMUMENTAÇÃO TÉCNICA DO EDITOR GENÉRICO
SENSÍVEL À SINTAXE

CAETANO TRAINA JUNIOR
CARLOS ROBERTO VALÊNCIO

Nº 11

RELATÓRIOS TÉCNICOS



São Carlos - SP

Instituto de Ciências Matemáticas de São Carlos

ISSN - 0103-2569

Documentação Técnica do Editor Genérico Sensível à Sintaxe

CAETANO TRAINA JUNIOR

CARLOS ROBERTO VALÊNCIO

Nº 11

RELATÓRIOS TÉCNICOS DO ICMSC

São Carlos

Nov. / 93

Universidade de São Paulo
Instituto de Ciências Matemáticas de São Carlos
Departamento de Ciências de Computação e Estatística

DOCUMENTAÇÃO TÉCNICA DO EDITOR GENÉRICO SENSÍVEL À SINTAXE

Caetano Traina Júnior

Carlos Roberto Valêncio

Endereço: *Instituto de Ciências Matemáticas de São Carlos - ICMSC*
Av. Dr. Carlos Botelho, 1465
CEP. 13560.970 - Centro
FAX: (0162)749150 - TEL:(0162)726222 Ramal 3851

DOCUMENTAÇÃO DO
EDITOR GENÉRICO SENSÍVEL À SINTAXE

Resumo

Este trabalho descreve uma parte da documentação do projeto de implementação do Protótipo de Editor Sensível à Sintaxe (PEGSS) [Val_93], composta pelos seguintes elementos:

. Composição Hierárquica dos Gabaritos das Linguagens Pascal, C e FORTRAN;
. Definição da Linguagem Abstrata;
. Esquemas de Recuperação de Programas para as Linguagens Pascal, C e FORTRAN.

Os elementos que compõem esta documentação permitem ao PEGSS exibir um programa através das composições hierárquicas da linguagem corrente, armazenar genericamente um programa e recuperá-lo sob o contexto de qualquer das linguagens suportadas pelo Editor.

1 - Introdução

Na atual versão do Editor é oferecida a edição sensível à sintaxe para um subconjunto das linguagens Pascal, C e FORTRAN. Programas sintaticamente corretos são construídos através da requisição pelo usuário de gabaritos que correspondem aos comandos e pela descrição das expressões. Os comandos de atribuição e as expressões são introduzidas digitando-se caractere por caractere, enquanto as demais estruturas da linguagem são oferecidas como gabaritos, através dos menus de opções.

Um gabarito representa uma estrutura sintática da linguagem composto por palavras-chave e lugares para as possíveis introduções de outros gabaritos ou expressões. Os lugares para as novas introduções definem a classe das possíveis inserções, evitando-se erros sintáticos na introdução de novas estruturas. Erros léxicos também são evitados, pois não é permitido ao programador alterar as palavras-chave que compõem o gabarito, além da verificação léxica das expressões inseridas. Cada gabarito é representadado em uma composição hierárquica cujo processamento permite exibir e movimentar-se sobre o comando descrito por ele.

Por outro lado, cada programa escrito sob o contexto desse Editor é representado internamente através de um modo único, independentemente da linguagem utilizada para a sua construção. Assim, um conjunto de ferramentas pode beneficiar-se dessa forma de representação. Por exemplo, é permitido que programas escritos em um linguagem sejam obtidos em qualquer das outras linguagens contempladas pelo Editor. Além disso, pode ser efetuada a adição de novas linguagens pertencentes a um mesmo paradigma, pois a construção dessa representação interna prevê esta extensão.

O PEGSS utiliza-se de uma base de dados GEO (GErenciador de Objetos) [Tra_91] para armazenar seus programas e recuperá-los sob o contexto da linguagem desejada. O armazenamento dos programas é possível através da Descrição da Linguagem Abstrata (DLA) que define o Esquema da base de dados utilizada pelo Editor. A recuperação é proporcionada pelos Esquemas de Recuperação de Programas, sendo um Esquema para cada uma das linguagens suportadas pelo Editor. Estes Esquemas tambés são armazenados na base de dados.

Nas subseções seguintes deste trabalho são apresentados os elementos que compõem a documentação necessária para descrever as partes principais do Editor.

2 - Composição Hierárquica dos Gabaritos

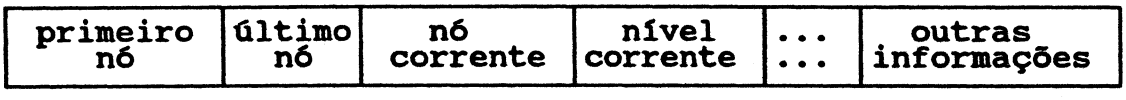
A maioria dos ambientes orientados para estruturas representam seus programas em árvores de sintaxe, de onde são obtidas informações não apenas para a exibição do programa, mas também para a verificação semântica, outros recursos de suporte à compilação incremental e auxílio à depuração.

O PEGSS utiliza-se de uma estrutura hierárquica para a representação intermediária de seus programas, através da qual são obtidas as informações para a exibição das estruturas sintáticas nela representadas e as correspondentes informações na base de dados do programa corrente.

A hierarquia de um programa é composta de dois tipos de elementos: uma estrutura de controle e seus nós. A estrutura de controle contém informações gerais de uma hierarquia, como por exemplo, seus nós primeiro e último, o nó corrente e o seu nível, durante o processamento da hierarquia, etc. Um nó contém informações como o item sintático nele armazenado, código do objeto da base de dados que representa a estrutura sintática especificada neste nó, o nível em que está localizado na hierarquia, etc. Como um nó não está isolado em uma hierarquia, também estão armazenadas nele as informações do seu nó pai, seu nó irmão de cima (prévio), seu nó irmão de baixo (próximo) e seu primeiro nó filho. No apêndice A são omitidas algumas destas ligações, apenas para tornar mais legíveis as representações das subhierarquias ali descritas [Tra_93].

A ilustração abaixo mostra os dois tipos de elementos de uma hierarquia.

Representação de uma estrutura de controle de hierarquia:

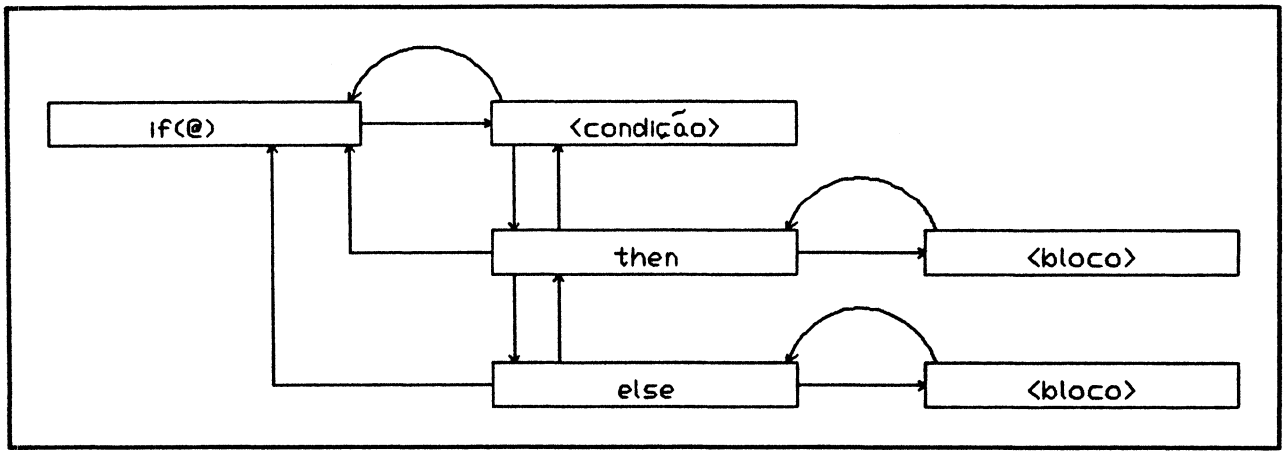


Representação de um nó da hierarquia:

nó pai	nó prévio	nó próximo	item sintático	...

...	código do objeto	nível	...	nó filho

A endentação apresentada pelas estruturas sintáticas, em um programa, é proporcionada pela disposição das palavras-chave e dos lugares de inserção nos nós da hierarquia, pois esta estrutura de dados permite que um gabarito tenha os seus itens hierarquicamente dependente de um outro, no nível imediatamente acima. Por exemplo, a estrutura "ifte" tem as palavras-chave "if(@)", "then", "else", além dos lugares de inserção <condição> e <bloco>, dispostos de modo que o item "if(@)" seja apresentado algumas colunas à esquerda do "then" e "else", indicando que esta estrutura tem início no item "if(@)" e que compreende os demais, inclusive seus respectivos lugares de inserção. Isto ocorre porque o item "if(@)" é alocado em um nó de nível superior aos outros itens "then" e "else", estando estes dois últimos, um nível acima dos itens <condição> e <bloco>. A composição hierárquica desta estrutura é mostrada a seguir:



Na representação da estrutura "ifte" , o item "if(@)" possui três itens subordinados e os dois lugares de inserção subordinados a dois destes itens. Mas em geral, um ítem pode possuir um número qualquer de subordinados, e cada subordinado também, em um número qualquer de níveis.

Para que as estruturas de um programa representadas em uma hierarquia sejam mostradas ao usuário, esta terá de ser preparada anteriormente e, depois, processada. Preparar uma hierarquia significa determinar, após uma atualização qualquer, as linhas e colunas iniciais nas quais deve ser exibido cada um dos itens, nos respectivos nós, na janela de apresentação dos programas do PEGSS. Depois disso, a hierarquia poderá ser processada, isto é, o programa,

representado em uma hierarquia preparada, é mostrado ao usuário.

Exemplificando, a estrutura "ifte" representada na hierarquia acima terá, através do seu processamento, a seguinte apresentação:

```
if( <condição> )          if( <condição> )          if( <condição> )
  then                      then                      then
    <bloco>                   <bloco>                   <bloco>
  else                      else                      else
    <bloco>                   <bloco>                   <bloco>

    (1)                       (2)                       (3)
```

Na representação (1) da estrutura "ifte" aparecem em destaque todos os itens e lugares de inserção que compõem o seu gabarito, pois o nó corrente durante o processamento é o que contém o item "if(@)". Pressionando-se a "seta para a direita", o nó corrente torna-se o que contém o lugar de inserção <condição>, deixando apenas este em destaque na sua representação, como mostrado em (2), pois este nó não possui qualquer nó filho. Através da "seta para baixo", a "navegação" na hierarquia segue, posicionando-se no nó abaixo do corrente do mesmo nível, deixando em destaque o item do nó corrente e seus filhos, como se observa em (3).

Outros detalhes sobre inclusão, substituição e eliminação em composições hierárquicas podem ser encontradas em [Val_93]. A respeito dos detalhes de implementação destas estruturas pode-se encontrar em [Tra_93] e [Val_93].

São apresentadas no **apêndice A** as subhierarquias que compõem cada um dos gabaritos utilizados para a representação intermediária das estruturas sintáticas da linguagem Pascal, C e FORTRAN. Para facilitar esta apresentação são omitidas algumas ligações dos nós. Na maioria das vezes é mostrado o nó isolado contendo um lugar de inserção e, logo abaixo, a subhierarquia que o substituirá durante uma seção de edição.

3 - Descrição da Linguagem Abstrata

O PEGSS utiliza uma base de dados orientada a objetos denominada de GEO (GErenciador de Objetos) [Tra_91] para o armazenamento da maioria das informações manipuladas. O GEO é um Sistema de Gerenciamento de Dados para Aplicações de Projeto, baseado no Modelo de Representação de Objetos (MRO) [Tra_86] [Tra_88] [Bia_92]

Um programa escrito, utilizando o PEGSS, poderá ter a sua representação em qualquer das linguagens por ele suportadas, dentro dos limites da construção sintática da linguagem em que se deseja tê-lo representado. Por exemplo, a definição de novos tipos de dados em um programa editado sob o contexto da linguagem Pascal ou C não poderá ter uma representação equivalente em uma tradução deste programa para a linguagem FORTRAN, pois tal especificação sintática não é possível nesta última linguagem. Deste modo, outras estruturas sintáticas definidas para uma linguagem podem não ter uma representação equivalente em outra linguagem. Como as linguagens suportadas pelo Editor pertencem a um mesmo paradigma, a maioria de suas estruturas sintáticas possui uma representação equivalente entre elas.

A representação de um programa, nas várias linguagens, é possível para o PEGSS graças ao modo único segundo o qual são armazenadas as informações sobre as estruturas sintáticas que o descrevem. Este modo único, ou genérico, de armazenamento de um programa está definido através da Descrição da Linguagem Abstrata (DLA).

A DLA foi concebida de modo que apenas as informações comuns às linguagens de uma estrutura sintática fossem armazenadas na base de dados e as informações particulares para cada linguagem da estrutura fossem obtidas, quando sob o seu contexto. Por outro lado, algumas informações representadas na DLA só têm sentido para algumas linguagens e não para as outras, cabendo ao Editor selecionar as informações da DLA referentes a uma linguagem, quando sob o seu contexto.

Para a representação da DLA, foi utilizado o MRO, de acordo com o qual cada estrutura sintática é definida através de um tipo de objeto, e as estruturas, que podem estar a ela associadas são definidas por meio das suas modalidades. Em um tipo de objeto pode estar representada toda uma classe sintática, sendo que as estruturas possíveis de serem descritas neste tipo de objeto definem esta classe.

Para a representação da DLA foi utilizado o Diagrama de Representação de Objetos (DRO) e as figuras que a descrevem são apresentadas no **apêndice B**. As figuras estão dispostas em conjuntos que representam as estruturas sintáticas da DLA, relevantes para o Editor, quando sob o domínio de uma dada linguagem. Deste modo, nas Figuras Pascal-1 até a Pascal-13 estão representadas as estruturas da DLA manipuladas pelo Editor, quando sob o contexto da

linguagem Pascal; as Figuras C-1 até a C-13 contém as estruturas da DLA consideradas pelo Editor, quando sob o contexto da linguagem C; e as Figuras FORTRAN-1 até a FORTRAN-11 possuem representadas as estruturas da DLA tratadas pelo Editor, quando sob o contexto da linguagem FORTRAN.

4 - Esquemas de Recuperação de Programas

Para cada linguagem existe um Esquema de Recuperação de Programas (ERP). Como a versão atual do PEGSS suporta subconjuntos das linguagens Pascal, C e FORTRAN, então os Esquemas definidos são: o Esquema de Recuperação de Programas para a linguagem Pascal (ERPPas), o Esquema de Recuperação de Programas para a linguagem C (ERPC) e o Esquema de Recuperação de Programas para a linguagem FORTRAN (ERPFOR). Estes Esquemas compreendem as informações para a leitura de um programa armazenado na base de dados e de sua apresentação na linguagem corrente.

As informações de cada um dos ERP estão armazenadas na base de dados, através de colônias de objetos. Cada objeto possui informações sobre a recuperação e apresentação de uma estrutura armazenada na base de dados, através de um objeto de mesmo tipo. Por exemplo, na colônia que contém o ERPPas existe um objeto do tipo "def_const" dotado das informações para a recuperação e apresentação de uma definição de constante qualquer, em um programa para a linguagem Pascal. O modo de obter das informações de um objeto do tipo "def_const", e de sua representação, segundo a linguagem Pascal, se obtém através do objeto de tipo "def_const" em ERPPas. Do mesmo modo existem nas colônias que contém o ERPC e o ERPFOR um objeto do tipo "def_const" que permite a recuperação e apresentação de uma estrutura de definição de constante armazenada na base de dados para as respectivas linguagens C e FORTRAN.

Todos os objetos das três colônias que compreendem os ERP possuem o tipo de atributo "lebase", pois a maioria das informações de obtenção e apresentação de um programa na base de dados estão na forma de atributos para este tipo de atributo. Estas informações também se encontram nos objetos cujos tipos possuem subtipos, como atributos de outros tipos diferente do "lebase".

O tipo de atributo "ifte" contém as informações de recuperação e apresentação de objetos de subtipo "ifte" da colônia que contém o programa que está sendo lido. Como o tipo de objeto "comando", na DLA, possui outros subtipos ("while", "atrib", "ift" e etc), existem no objeto

de tipo "comando", no ERPPas, os tipos de atributos "while", "atrib", "ift" e etc. com as informações de leitura e apresentação correspondente a estes subtipos.

As informações de leitura do programa na base de dados se constituem em instruções para a rotina de leitura, determinamtes da ação a ser tomada em relação ao objeto corrente na colônia que contém o programa. Algumas destas instruções completam-se com outras informações que compõem o atributo. Estas instruções têm a seguinte forma:

Função	Modalidade ou Tipo de Atributo

ou

Função

Uma Função pode ser:

TIPO: obter o atributo do tipo de atributo "tipo" do objeto corrente na colônia que contém o programa que está sendo lido.

RELA: a partir do objeto corrente, na colônia que contém o programa, posicionar no objeto destino através da modalidade especificada neste atributo corrente. Se existir o relacionamento com origrm nesse objeto, o objeto destino torna-se o objeto corrente, e lêem-se os atributos de seu tipo de atributo "lebase".

ATRI: ler o atributo do objeto corrente na colônia que contém o programa cujo tipo é especificado neste atributo corrente.

FIM: determina o final de leitura dos atributos do tipo de atributo "lebase" do objeto corrente.

Uma função pode vir ou não seguida de uma modalidade ou um tipo de atributo, determinando a peça sobre a qual a função deverá ser aplicada. O elemento que acompanha a função especifica a peça[1] do objeto corrente na colônia que contém o programa que deverá ser

[1] *peça: pode ser tipo de objeto, tipo de atributo, modalidade, etc*

aplicada a função.

Podem ser exemplos de atributos que compõem as instruções descritas acima:

```
ATRI "descrição"
RELA   "segue"
FIM
```

onde, ATRI, RELA e FIM são funções já descritas acima, "descrição" é um tipo de atributo e "segue" é uma modalidade.

As informações segundo as quais a estrutura representada no objeto corrente deve ser exibida sob forma de gabaritos também se constituem em instruções nos atributos do tipo "lebase". Estas informações são utilizadas pelas rotinas que tratam da construção da hierarquia de um programa. As instruções que descrevem estas informações possuem o seguinte formato:

Função	Texto	Incremento de nível

ou

Função

Uma função pode ser:

INSE: inserir um novo nó na hierarquia com o texto e incremento de nível especificado no restante desta instrução.

SUBS: substituir o texto do nó corrente da hierarquia pelo texto especificado nesta instrução.

AVAN: posicionar no nó filho do nó corrente da hierarquia.

ELIM: extrair o nó corrente da hierarquia.

CONC: concatenar ao texto do nó corrente da hierarquia o texto especificado nesta instrução.

ULTI: posiciona no último nó do mesmo nível, tornando-o o nó corrente.

As funções descritas acima podem ser seguidas de um texto e um incremento de nível. Um texto é uma cadeia de caracteres que descreve um ítem sintático de um gabarito ou uma expressão. O incremento de nível é o incremento positivo ou negativo de nível que deverá sofrer o nó corrente da hierarquia.

São exemplos de atributos que compõem algumas das instruções descritas acima:

```
SUBS    if(@)          0
INSE    <condição>     1
ELIM
```

onde,

SUBS, INSE e ELIM são funções já descritas acima, if(@) é um texto que descreve um ítem sintático, <condição> é um texto que descreve um lugar de inserção de um gabarito, 0 e 1 são incrementos de nível.

Através do modo segundo a qual as informações de recuperação e apresentação foram definidas, é possível obter as várias representações de uma estrutura genérica. Para isto, basta estar no contexto da linguagem alvo.

Algumas informações não são apresentadas, quando sob o contexto de certas linguagens, satisfazendo suas respectivas construções sintáticas. Por exemplo, a declaração de rótulos não aparece na representação do programa principal, quando sob o contexto da linguagem C e FORTRAN. A definição de tipos não aparece na representação do programa em FORTRAN. Outras informações são mostradas de modo diferente em cada uma das representações, como por exemplo, a declaração de variáveis. Isto é proporcianado através das informações descritas nos atributos apresentados nas figuras que seguem para o tipo de objeto "subrotina", nos ERP. Da representação na base de dados de uma subrotina principal é obtido apenas o que é relevante para a sua representação na linguagem corrente, além de apresentar as estruturas no lugar e formato especificado pela construção sintática desta linguagem.

Através das informações nos ERP, é possível que determinadas estruturas sintáticas inexistentes para uma linguagem sejam apresentadas por combinações de outras estruturas desta linguagem.

As figuras no **apêndice C** mostram os respectivos Esquemas de Recuperação de Programas para as linguagens Pascal, C e FORTRAN.

Referências Bibliográficas

[Bia_92] Mauro Biajiz, **"MRO* - Uma Atualização do Modelo de Representação de Objetos e uma Abordagem Funcional"**, Dissertação de Mestrado, ICMSC-USP, 1992.

[GEO_91] **"Manual de Referência do GErenciador de Objetos"**, Documento de Trabalho do Grupo de Base de Dados e Engenharia de Software do ICMSC, Outubro 1991.

[Tra-86] Caetano Traina Jr, **"Máquina e Modelo de Dados Dedicados para Aplicações de Engenharia"**, Tese de Doutorado, IFQSC-USP, Dezembro 1986.

[Tra_88] Caetano Traina Jr., e Jan Frans Willem Slaets, **"Um Modelo de Representação de Objetos"**, in Anais do 3o. Simpósio Brasileiro de Banco de Dados, Recife, pp. 227-242, Março de 1988.

[Tra_91] Caetano Traina Jr., **"GEO: Um Sistema de Gerenciamento de Base de Dados Orientado a Objetos - Estado Atual de Desenvolvimento e Implementaçao"**, in Anais do 6o. Simpósio Brasileiro de Banco de Dados, Manaus, pp. 193-207, Maio 1991.

[Tra_93] Caetano Traina Jr. e Carlos Roberto Valêncio, **"Primitivas para o Tratamento de Hierarquia de Textos"**, Notas Internas do ICMSC-USP, Outubro de 1993.

[Val_93] Carlos R. Valêncio, **"Um Editor Genérico Sensível à Sintaxe Armazenada numa Base de Dados"**, Dissertação de Mestrado - ICMSC - USP - São Carlos, Fevereiro de 1993.

APÊNDICE A

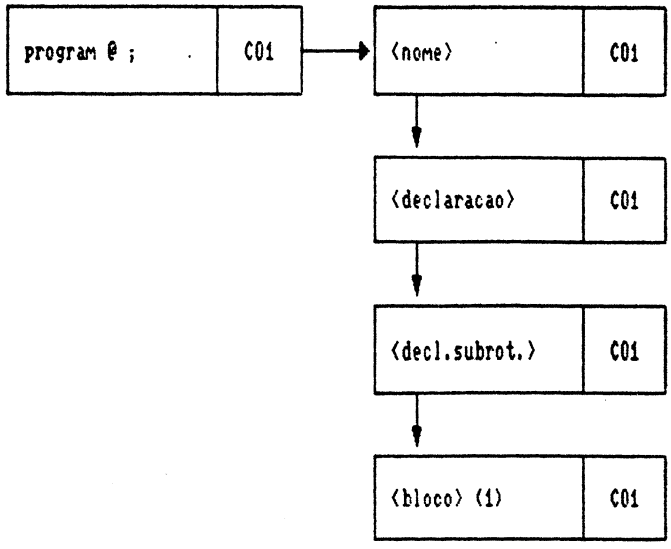
Composições Hierárquicas dos Gabaritos das Linguagens Pascal, C e FORTRAN

Considerações:

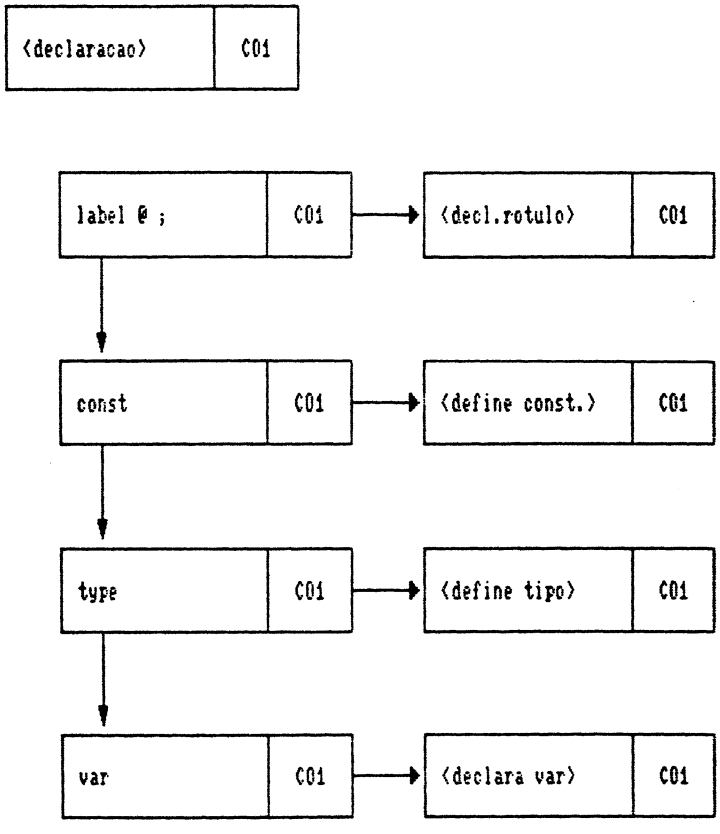
*São apresentadas as subhierarquias que compõem cada um dos gabaritos utilizados para a representação intermediária das estruturas sintáticas das linguagens **Pascal, C e FORTRAN**. Para facilitar esta apresentação são omitidos algumas ligações dos nós.*

APÊNDICE A

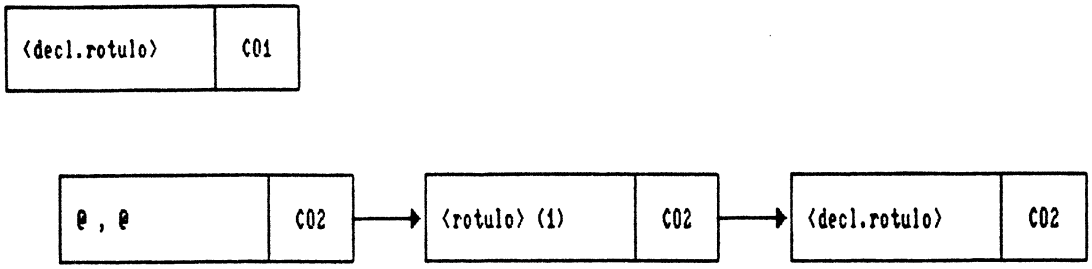
Composições Hierárquicas dos Gabaritos da Linguagem Pascal



C01 : subrotina (principal)

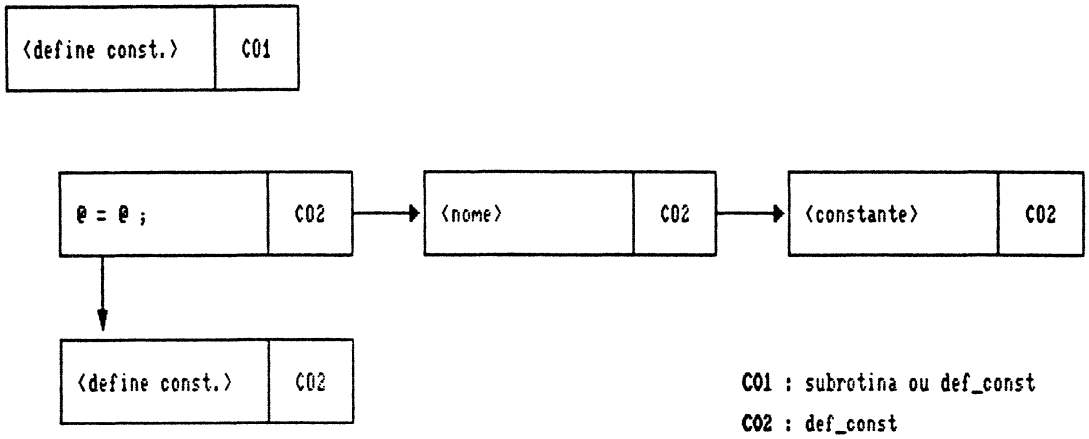


C01 : subrotina (principal ou procedimento ou funcao)



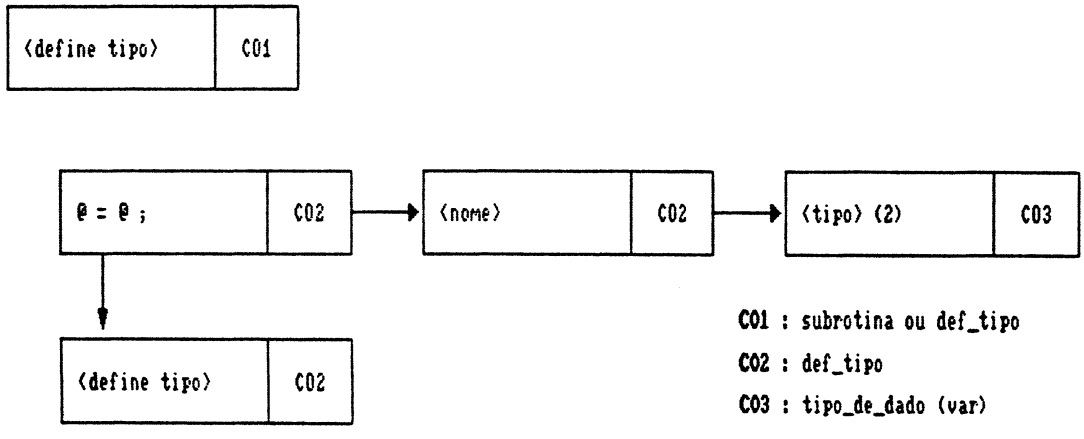
C01 : subrotina ou decl_rotulo

C02 : decl_rotulo



C01 : subrotina ou def_const

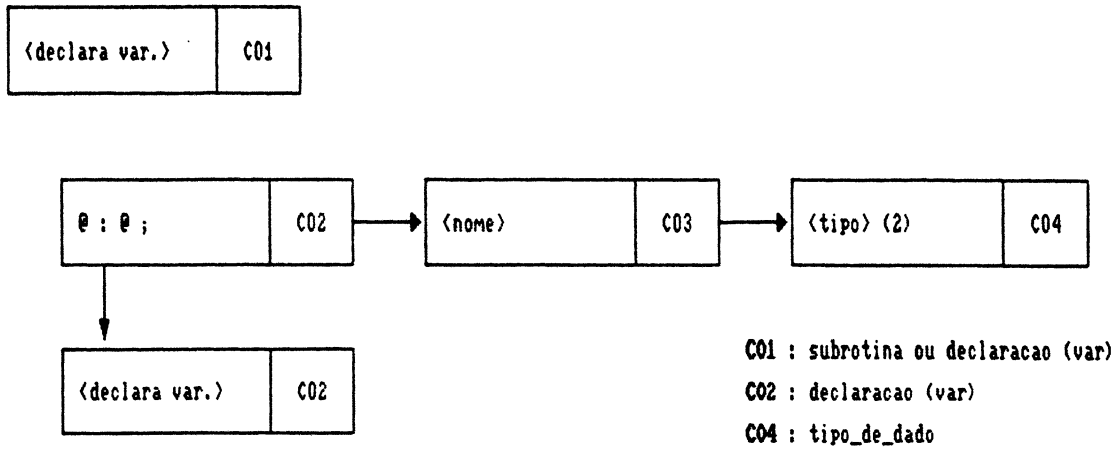
C02 : def_const



C01 : subrotina ou def_tipo

C02 : def_tipo

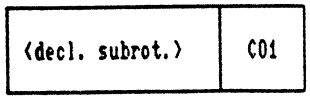
C03 : tipo_de_dado (var)

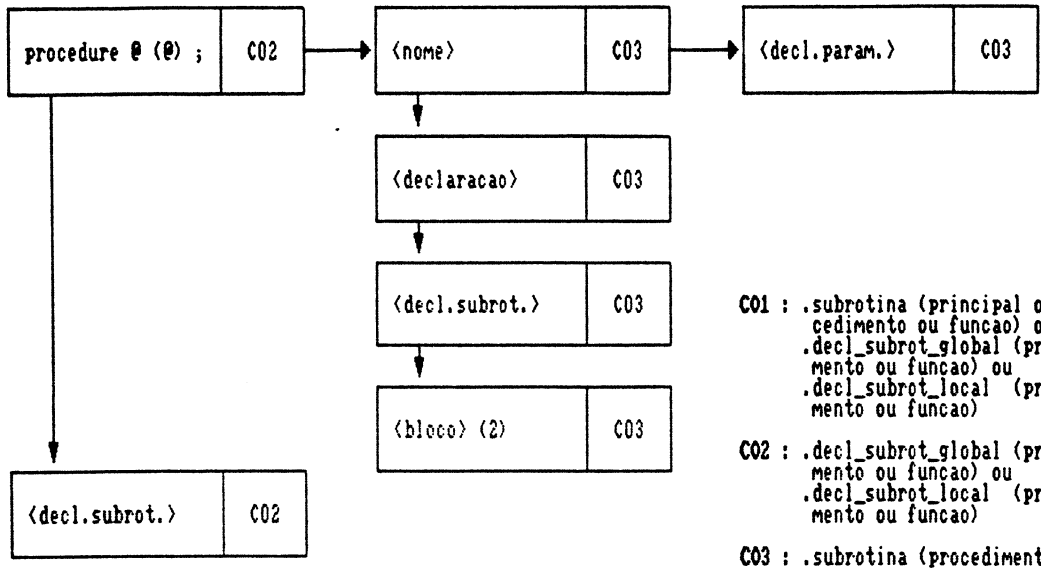


C01 : subrotina ou declaracao (var)

C02 : declaracao (var)

C04 : tipo_de_dado

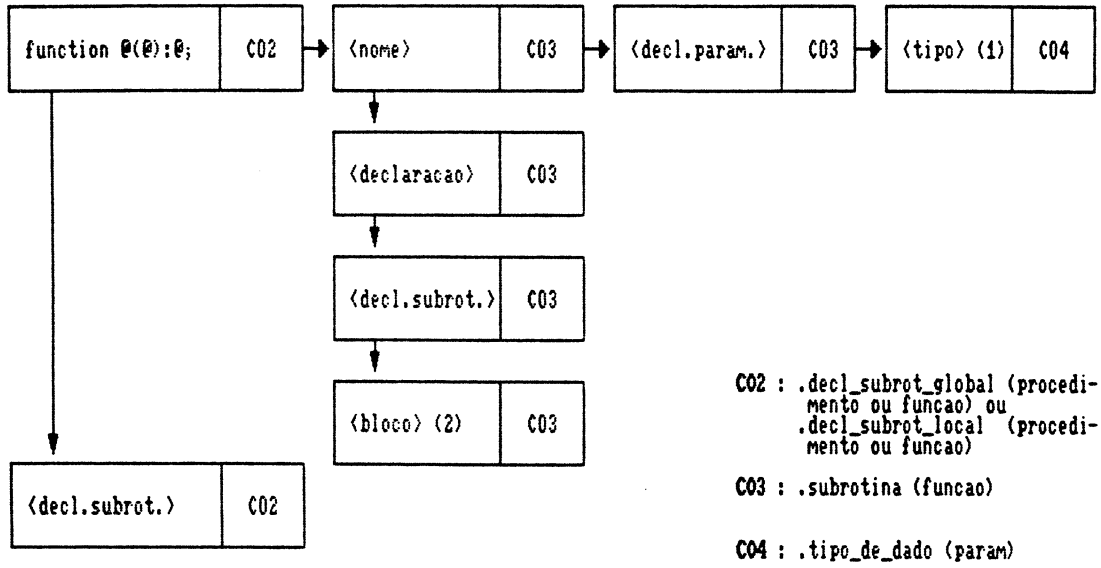




C01 : .subrotina (principal ou procedimento ou funcao) ou
.decl_subrot_global (procedimento ou funcao) ou
.decl_subrot_local (procedimento ou funcao)

C02 : .decl_subrot_global (procedimento ou funcao) ou
.decl_subrot_local (procedimento ou funcao)

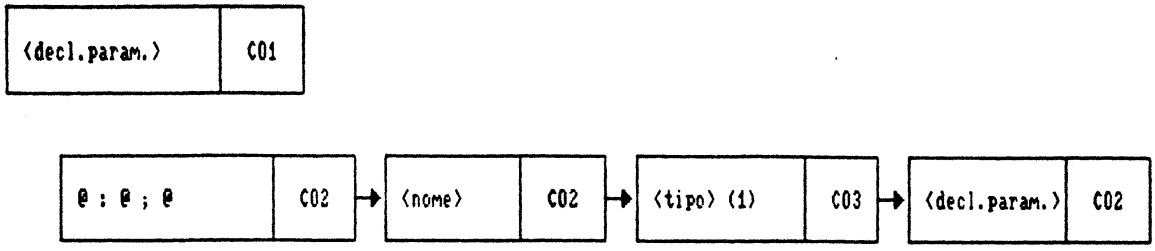
C03 : .subrotina (procedimento)



C02 : .decl_subrot_global (procedimento ou funcao) ou
.decl_subrot_local (procedimento ou funcao)

C03 : .subrotina (funcao)

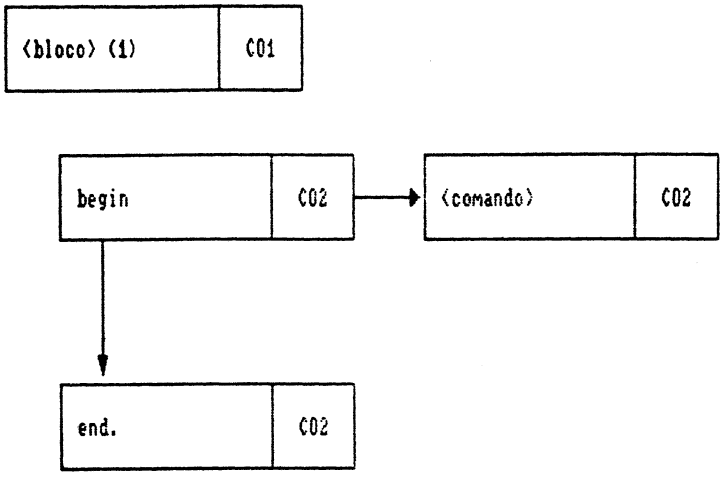
C04 : .tipo_de_dado (param)



C01 : subrotina (procedimento ou funcao

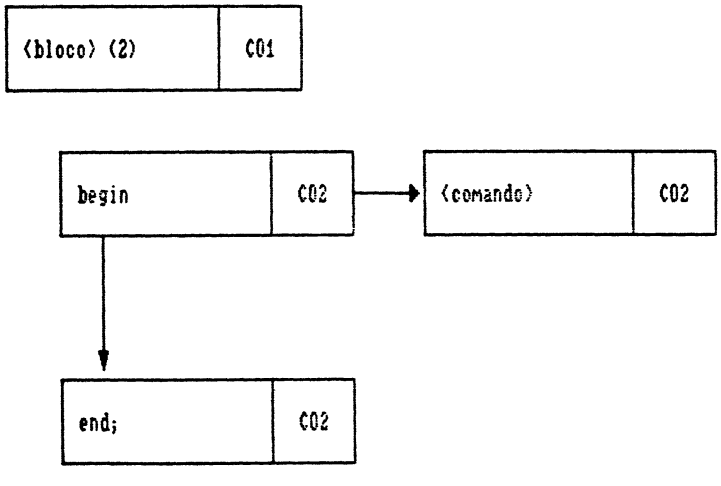
C02 : declaracao (param)

C03 : tipo_de_dado (param)



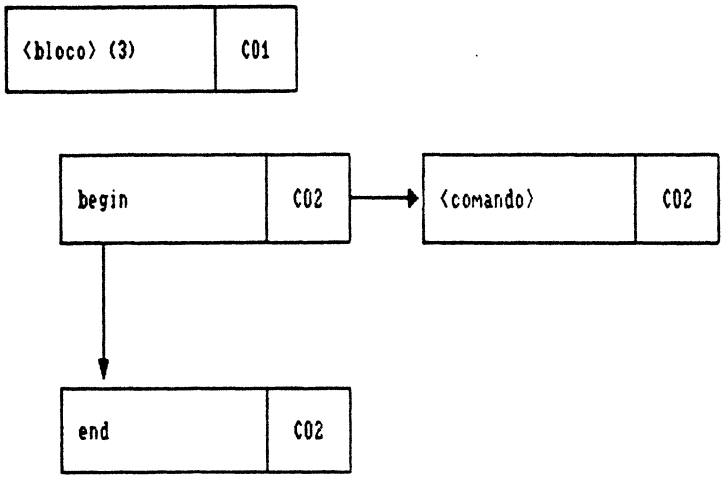
C01 : subrotina (principal)

C02 : bloco (inicio)



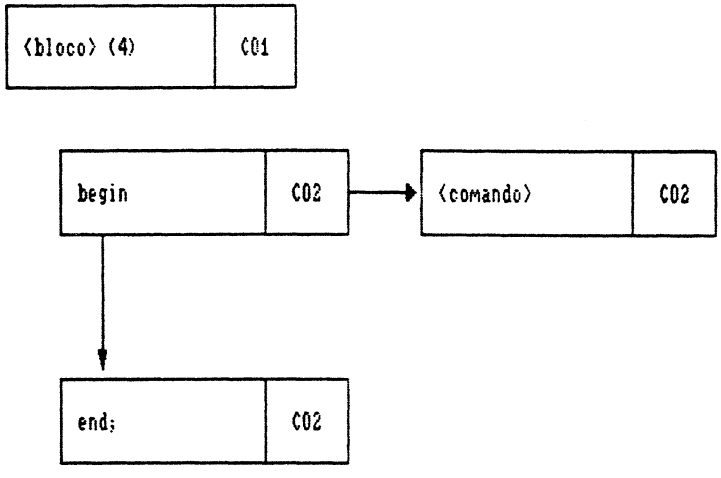
C01 : subrotina (proc. ou funcao)

C02 : bloco (subrot)



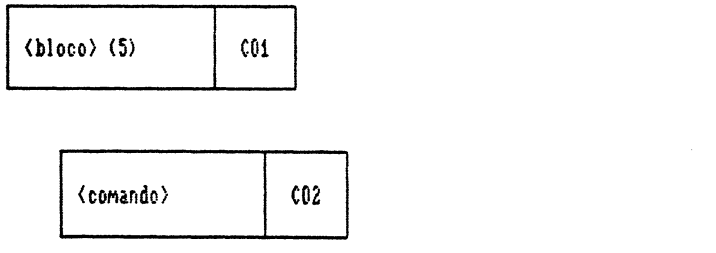
C01 : comando (ifte ou ifte_r)

C02 : bloco (composto (1))



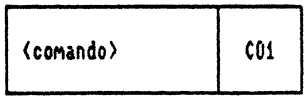
C01 : comando (ift, ift_r, ifte, ifte_r, while, while_r, do, do_r, for ou for_r)

C02 : bloco (composto (2))



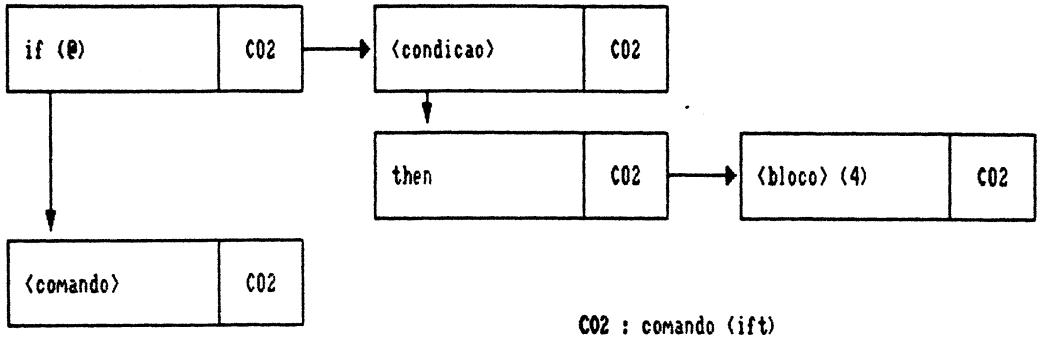
C01 : comando (repeat ou repeat_r)

C02 : bloco (composto (3))



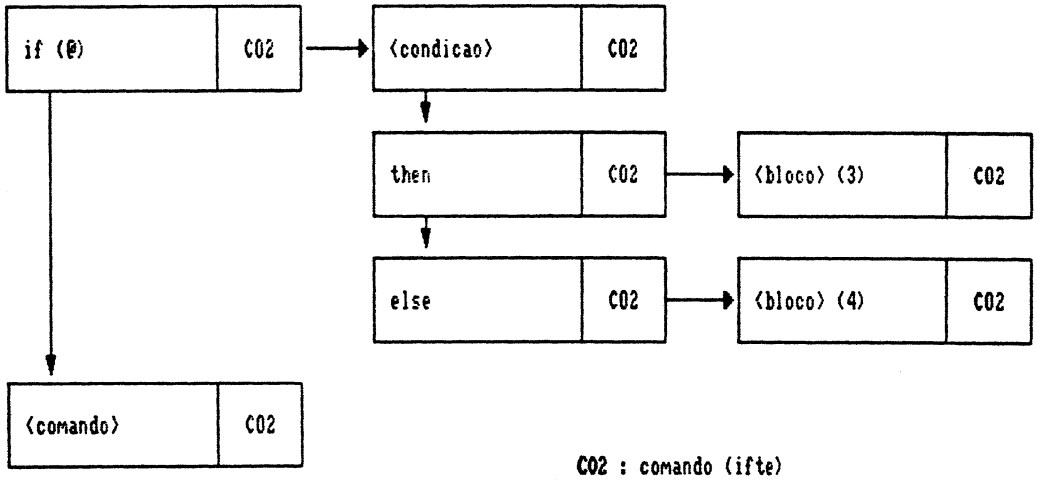
C01 : comando ou bloco (composto)

ift:



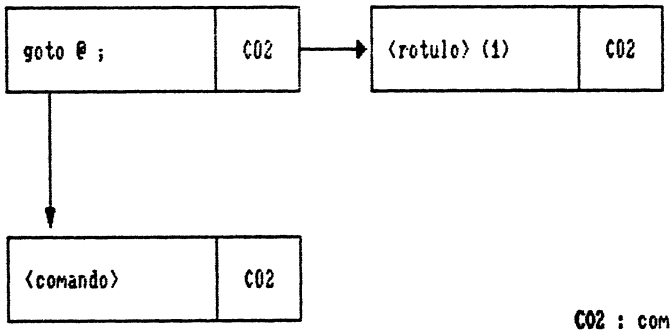
C02 : comando (ift)

ifte:



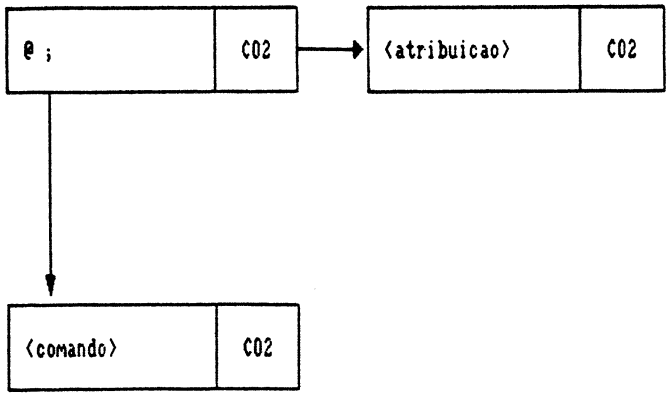
C02 : comando (ifte)

goto:



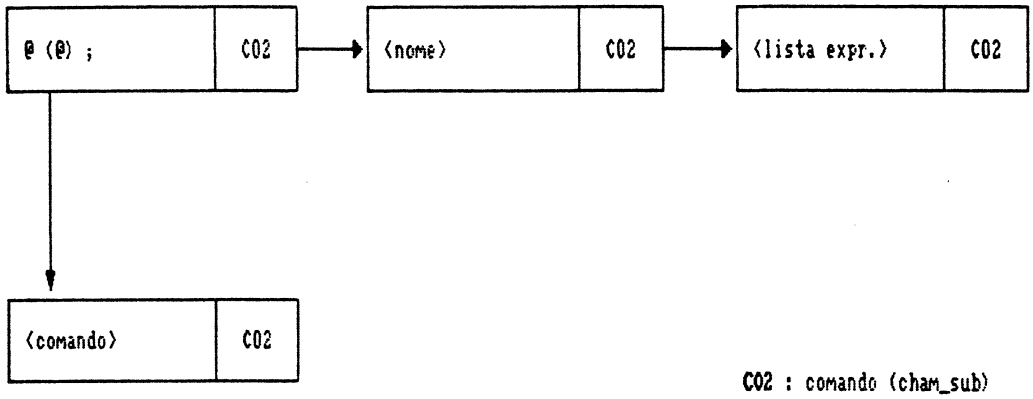
C02 : comando (goto)

atrib:



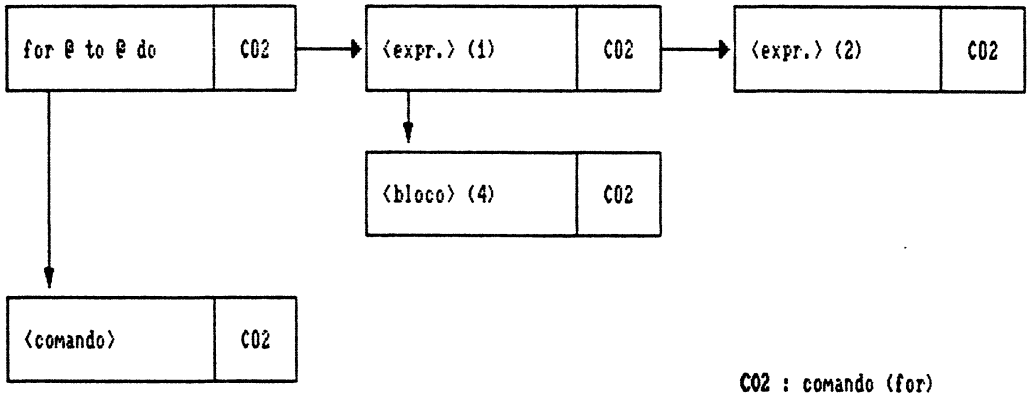
C02 : comando (atrib)

cham_sub:



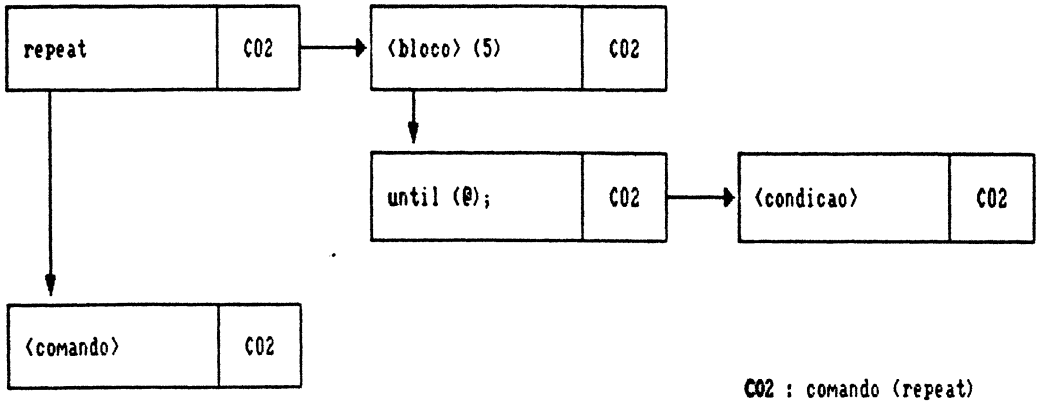
C02 : comando (cham_sub)

for:



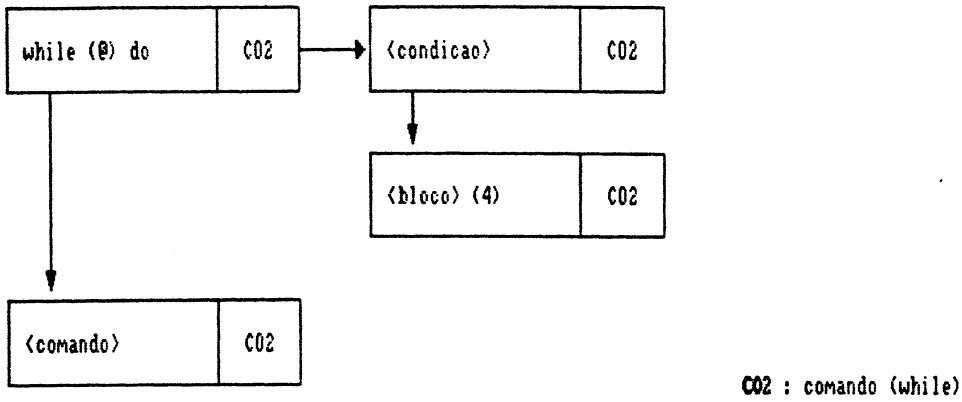
C02 : comando (for)

repeat:



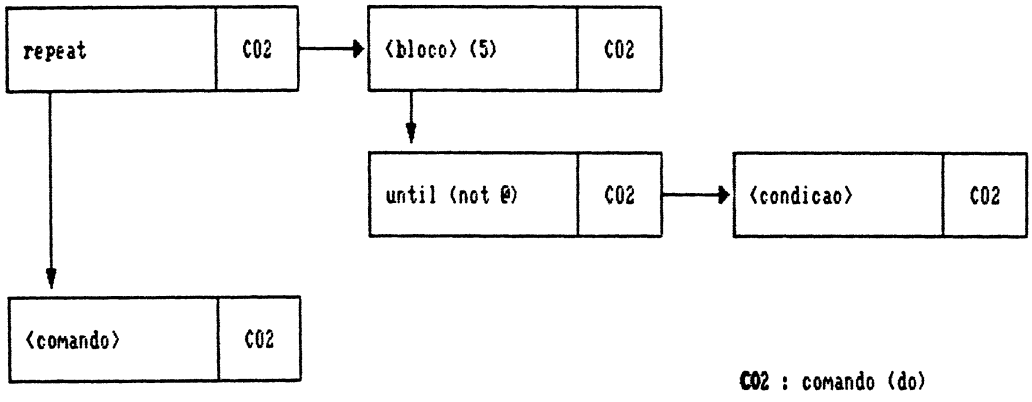
C02 : comando (repeat)

while:



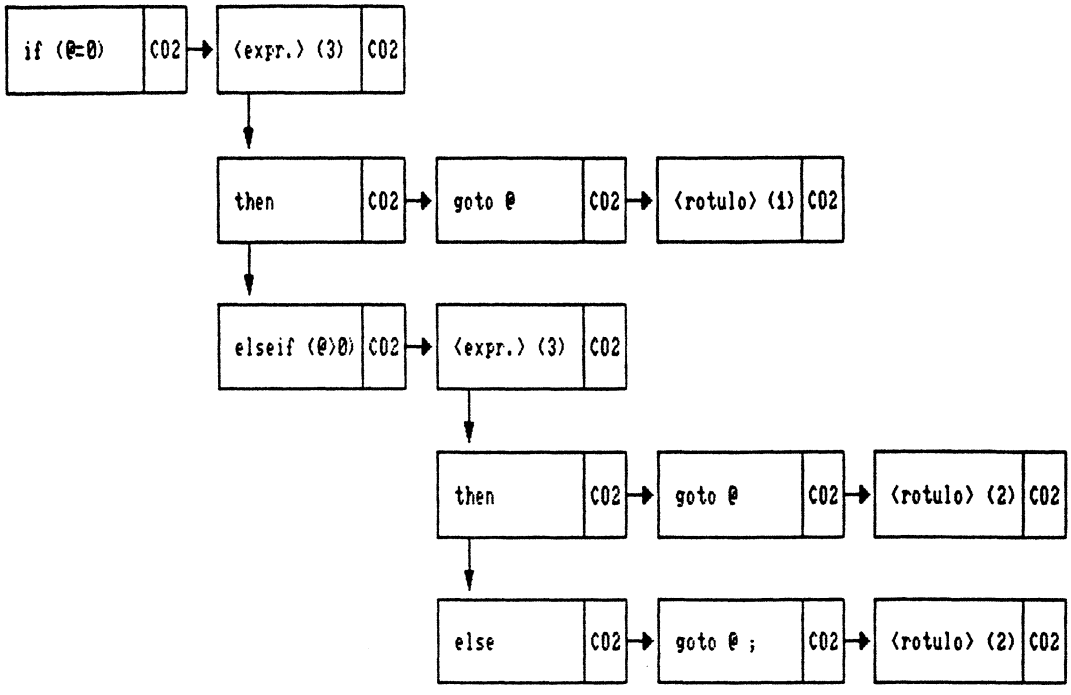
C02 : comando (while)

do:



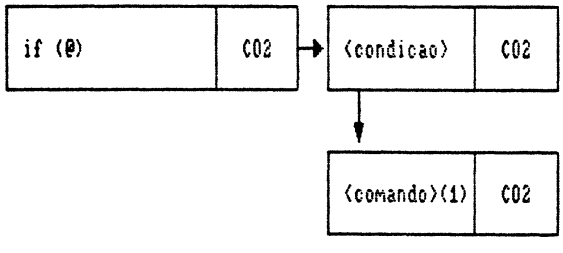
C02 : comando (do)

ifa:

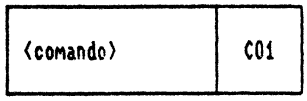


C02 : comando (ifa)

ifl:

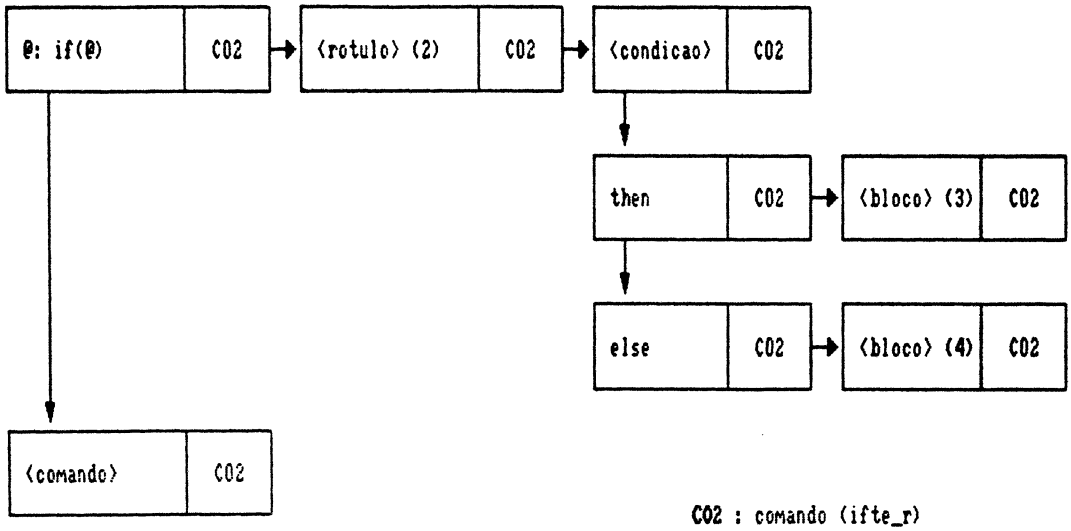


C02 : comando (ifl)



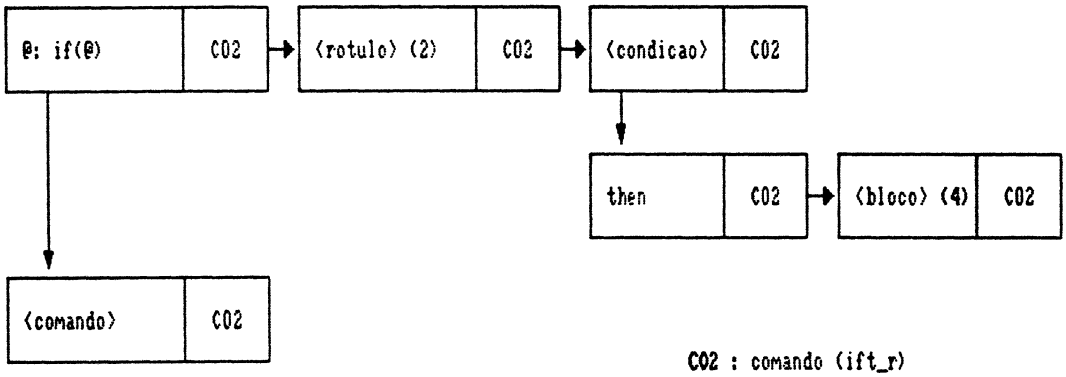
C01 : comando ou bloco (composto)

ifte_r:



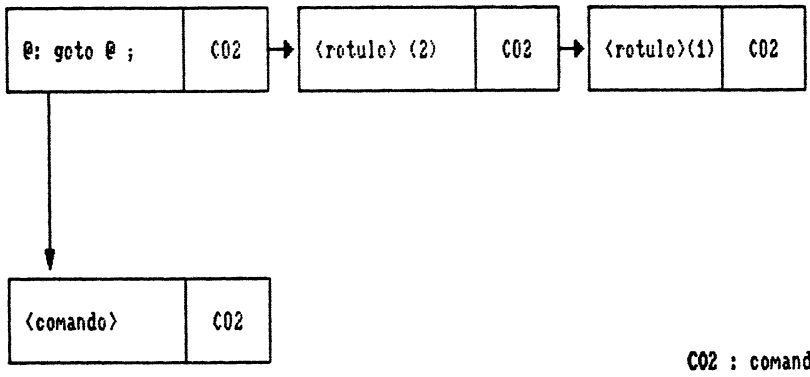
C02 : comando (ifte_r)

ift_r:



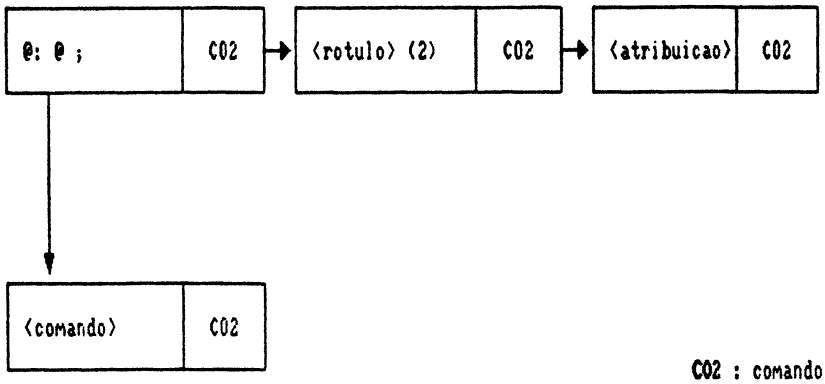
C02 : comando (ift_r)

goto_r:



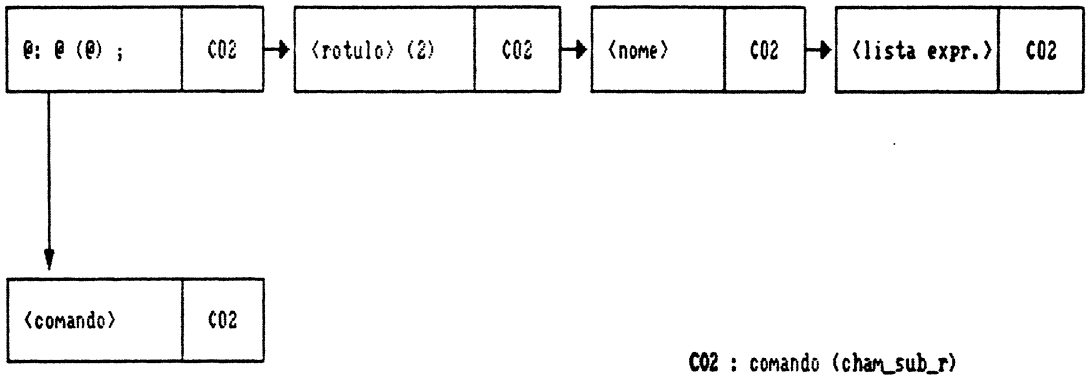
C02 : comando (goto_r)

atrib_r:



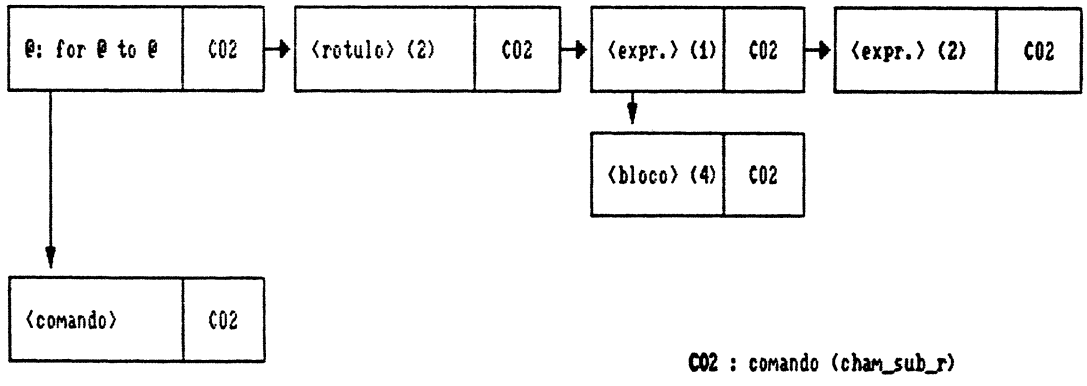
C02 : comando (atrib_r)

cham_sub_r:



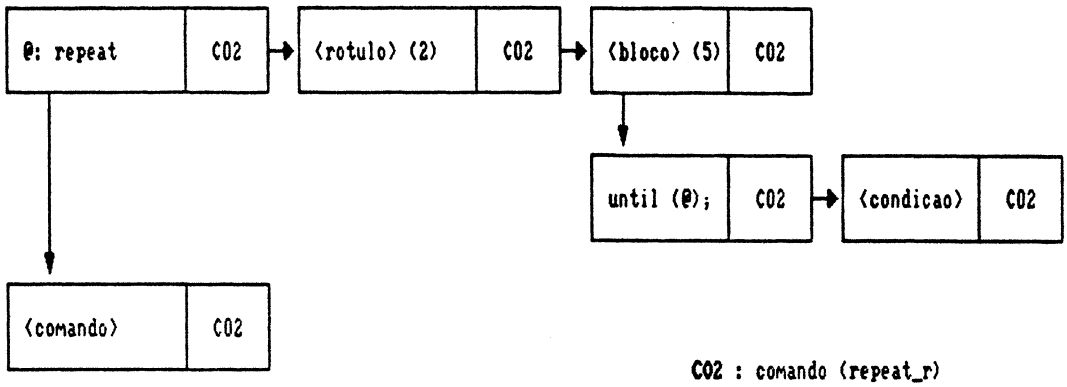
C02 : comando (cham_sub_r)

for_r:



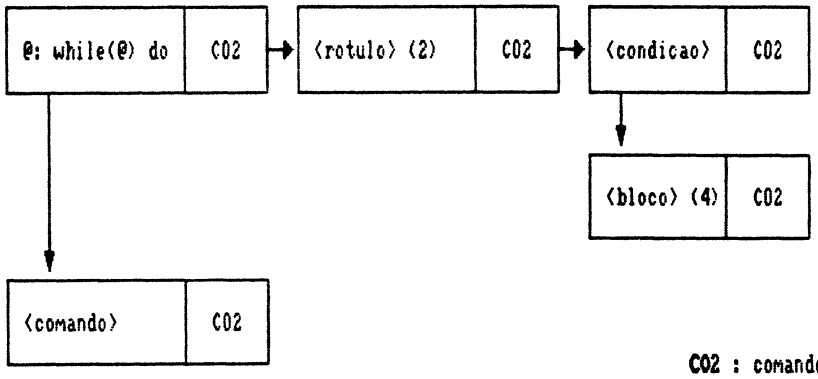
C02 : comando (cham_sub_r)

repeat_r:



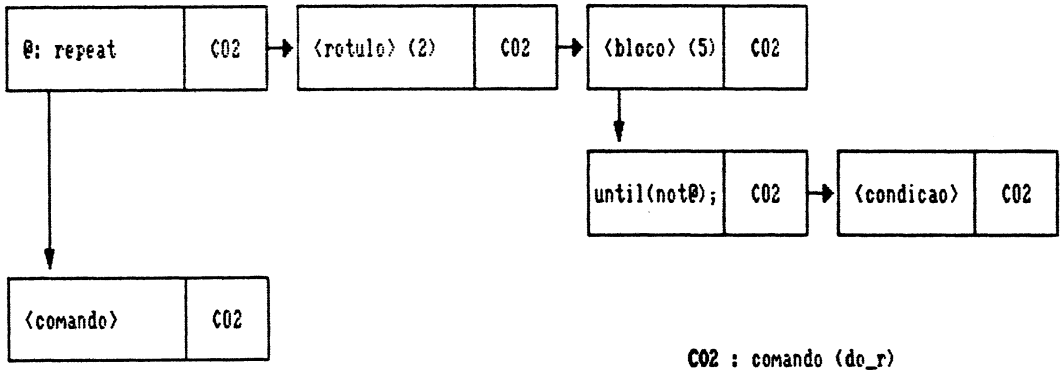
C02 : comando (repeat_r)

while_r:



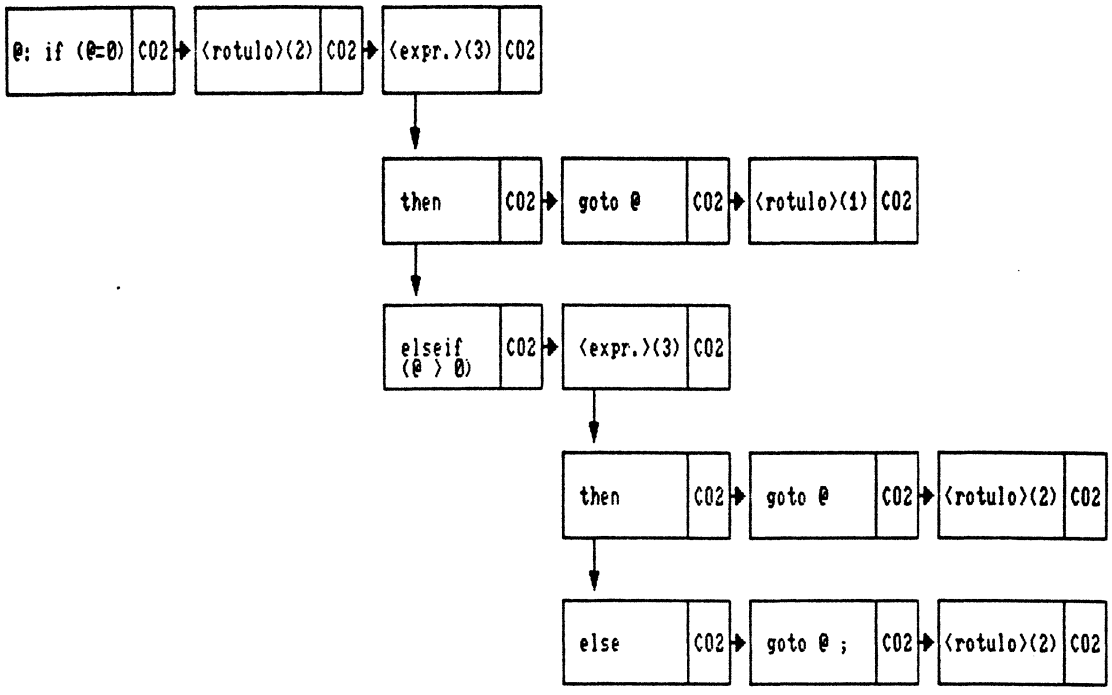
C02 : comando (while_r)

do_r:



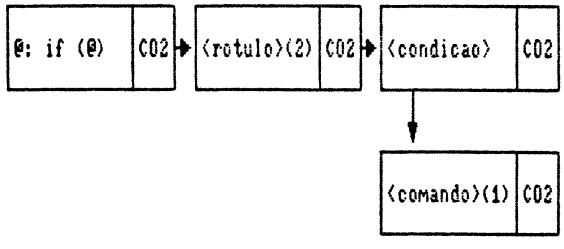
C02 : comando (do_r)

ifa_r:

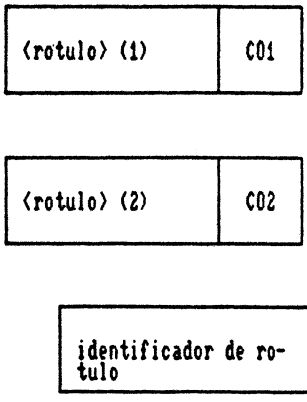


C02 : comando (ifa_r)

ifl_r:



C02 : comando (ifl_r)



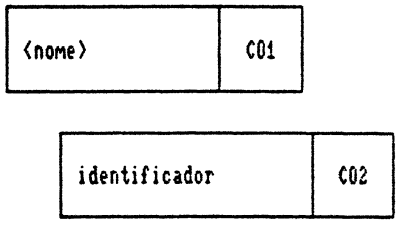
C01 : comando (goto ou goto_r)

C02 : comando com rotulo

C03 : rotulo

(1) : rotulo de desvio

(2) : rotulo de comando

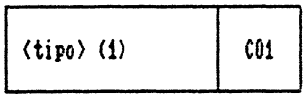


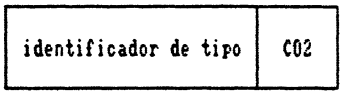
C01 :

- subrotina (principal ou funcao ou procedimento),
- def_const,
- def_tipo,
- declaracao (var),
- declaracao (param),
- comando (cham_sub ou (cham_sub_r).

C02 :

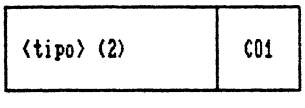
- nome_subrot,
- nome_const,
- ident_tipo_novo,
- nome_var,
- nome_parametro,
- nome_subrot.

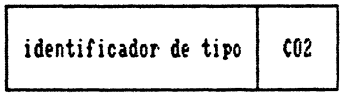




C01 : tipo_de_dado (param)

C02 : tipo_basico ou ident_tipo_novo

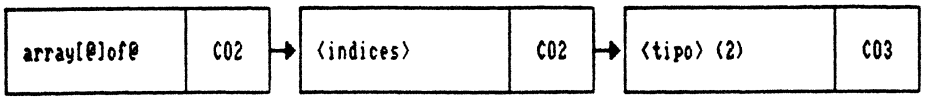




C01 : tipo_de_dado (var)

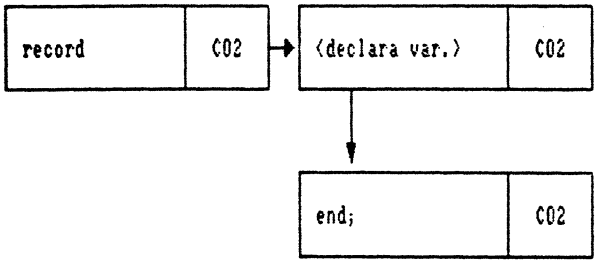
C02 : tipo_basico ou ident_tipo_novo

Vetor:

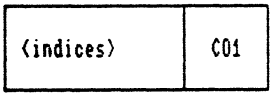


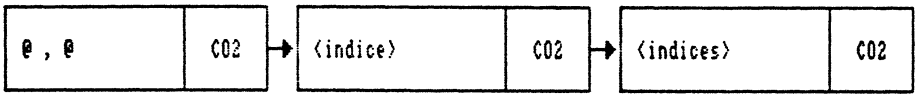
C02 : array
C03 : tipo_de_dado (var)

Registro:

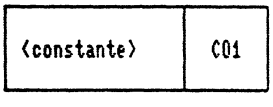


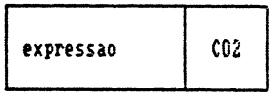
C02 : registro



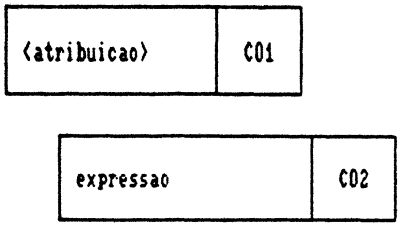


C02 : array ou indice
C03 : indice



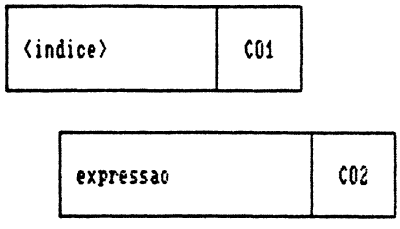


C01 : def_const
C02 : expressao (constante)



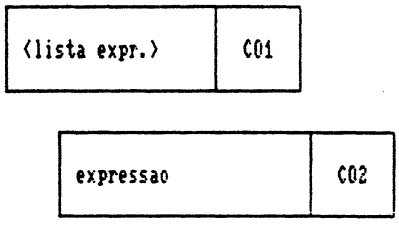
C01 : comando (atrib. ou atrib_r)

C02 : expressao (atribuicao ou retorno)



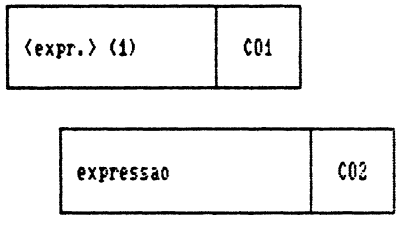
C01 : indice ou array

C02 : expressao (indice (indice_g ou indice_p ou indice_pc ou indice_pf))



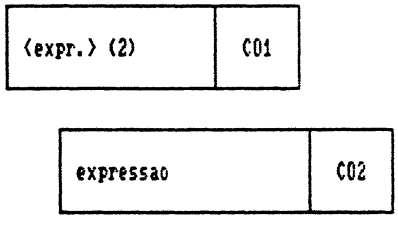
C01 : comando (cham_sub ou cham_sub_r)

C02 : expressao (lista_expr)



C01 : comando (for ou for_r)

C02 : expressao (inicio)



C01 : comando (for ou for_r)

C02 : expressao (parada)

⟨expr.⟩ (3)	C01

expressao	C02

C01 : comando (ifa ou ifa_r)

C02 : expressao (expr_aritm)

⟨condicao⟩	C01

expressao	C02

C01 : comando (ift, ifte, repeat, while, do, ifl,
ift_r, ifte_r, repeat_r, while_r,
do_r, ifl_r)

C02 : expressao (condicao)

identificador de tipo	C01

integer	C02

real	C02

char	C02

boolean	C02

tipo definido pelo usuario	C03

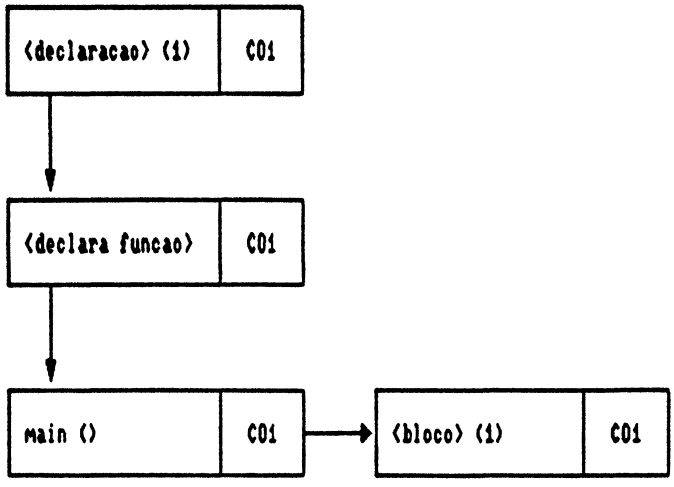
C01 : tipo_de_dado (var ou param)

C02 : tipo_basico

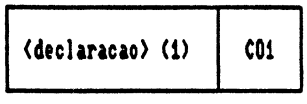
C03 : ident_tipo_novo

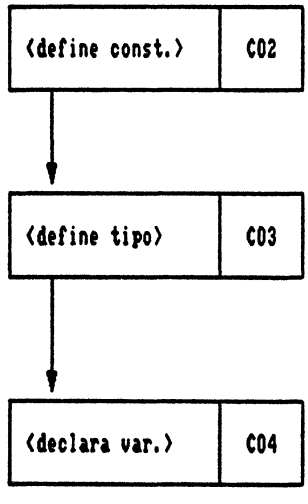
APÊNDICE A

Composições Hierárquicas dos Gabaritos da Linguagem C

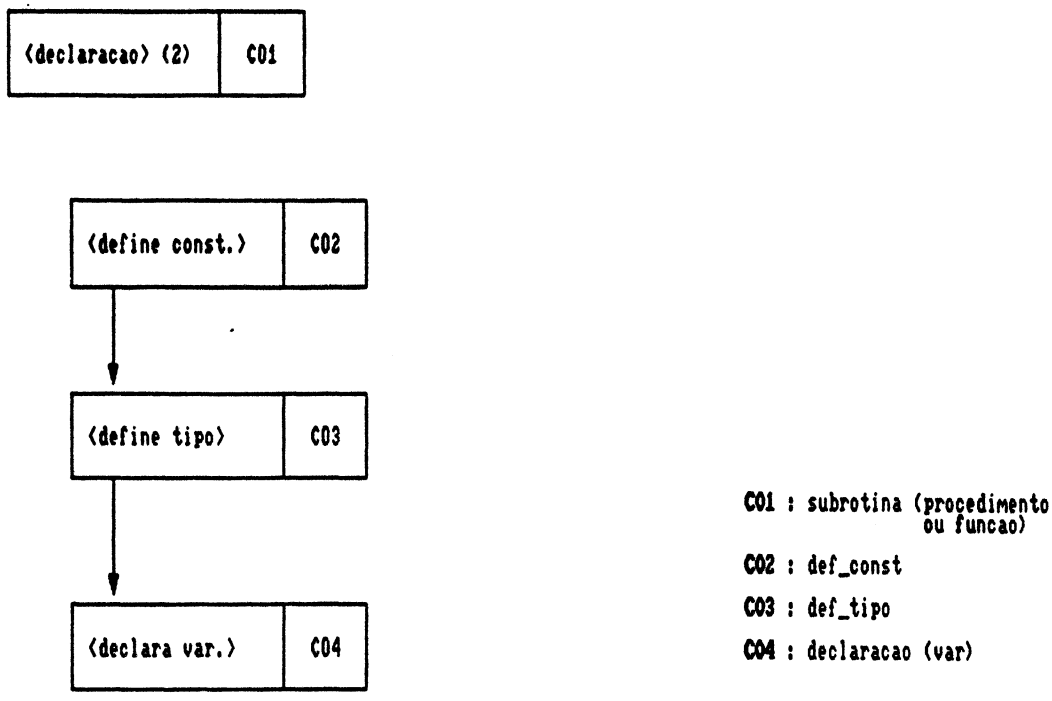


C01 : subrotina (principal)





C01 : subrotina (principal)
C02 : def_const
C03 : def_tipo
C01 : declaracao (var)

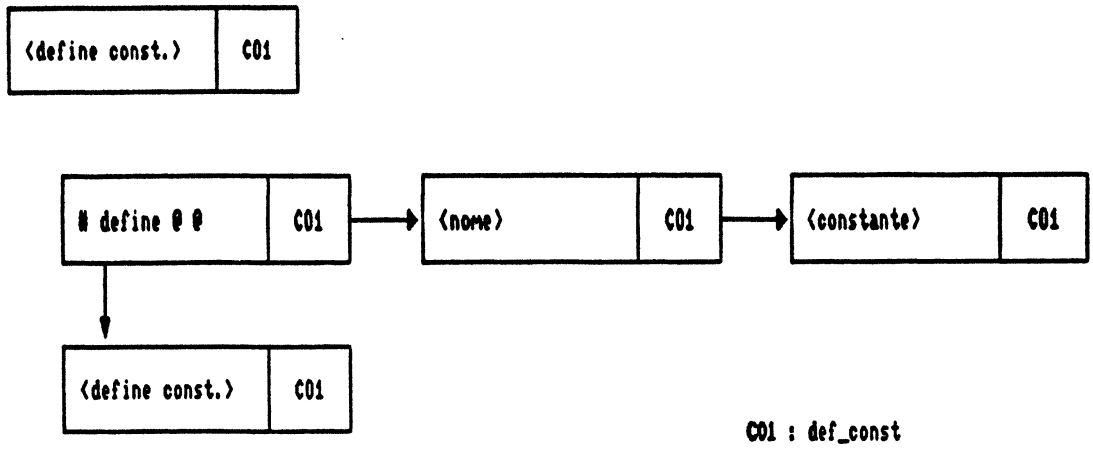


C01 : subrotina (procedimento ou funcao)

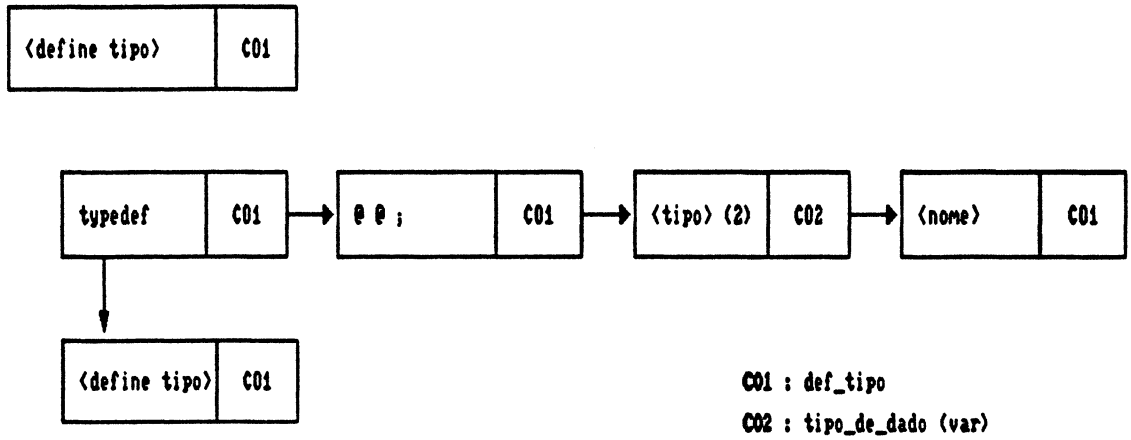
C02 : def_const

C03 : def_tipo

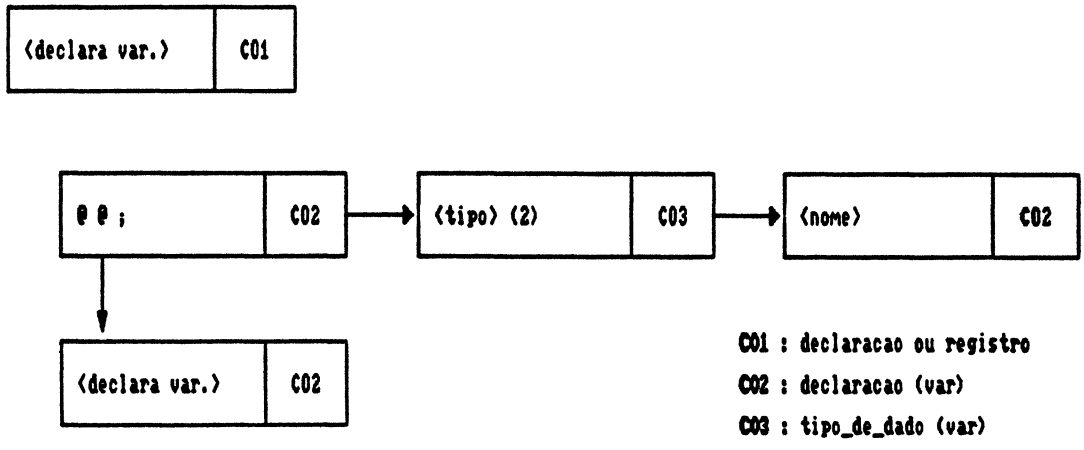
C04 : declaracao (var)



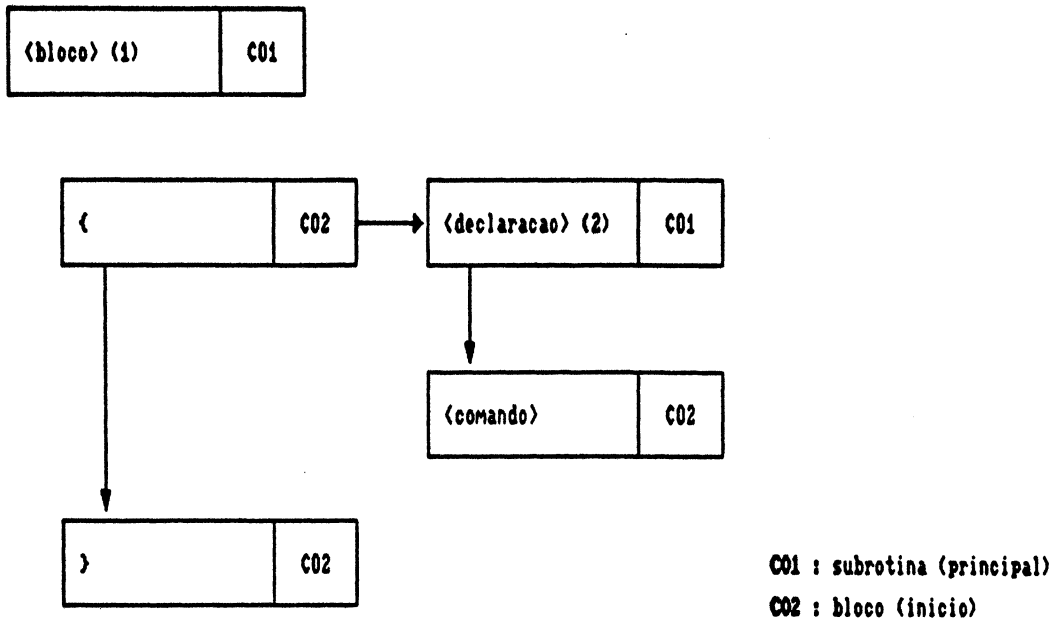
C01 : def_const



C01 : def_tipo
C02 : tipo_de_dado (var)

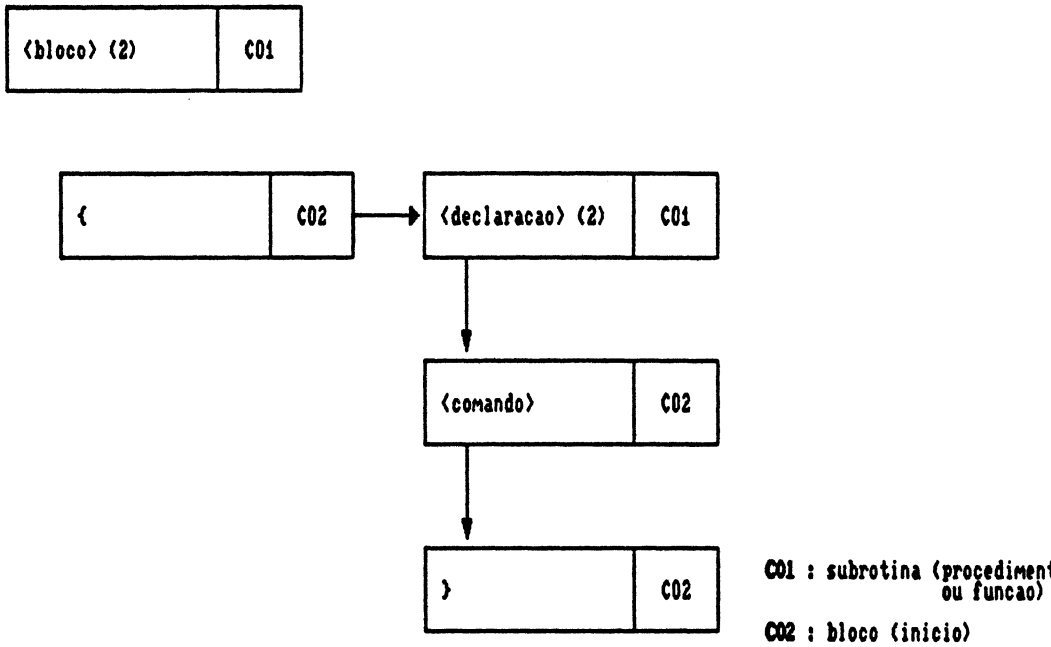


C01 : declaracao ou registro
C02 : declaracao (var)
C03 : tipo_de_dado (var)



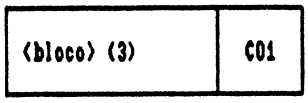
C01 : subrotina (principal)

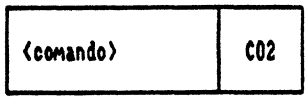
C02 : bloco (inicio)



C01 : subrotina (procedimento ou funcao)

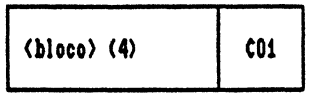
C02 : bloco (inicio)

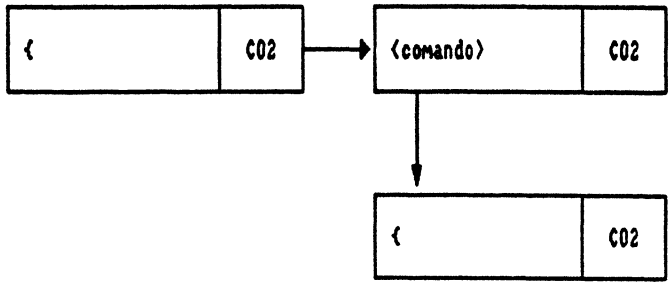




C01 : comando (ifte ou ifte_r)

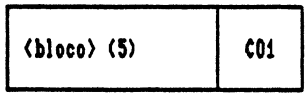
C02 : bloco (composto (1))

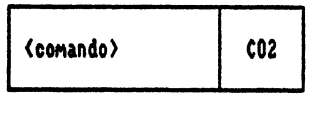




C01 : comando (ift, ift_r, ifte, ifte_r, while, while_r, do, do_r, for ou for_r)

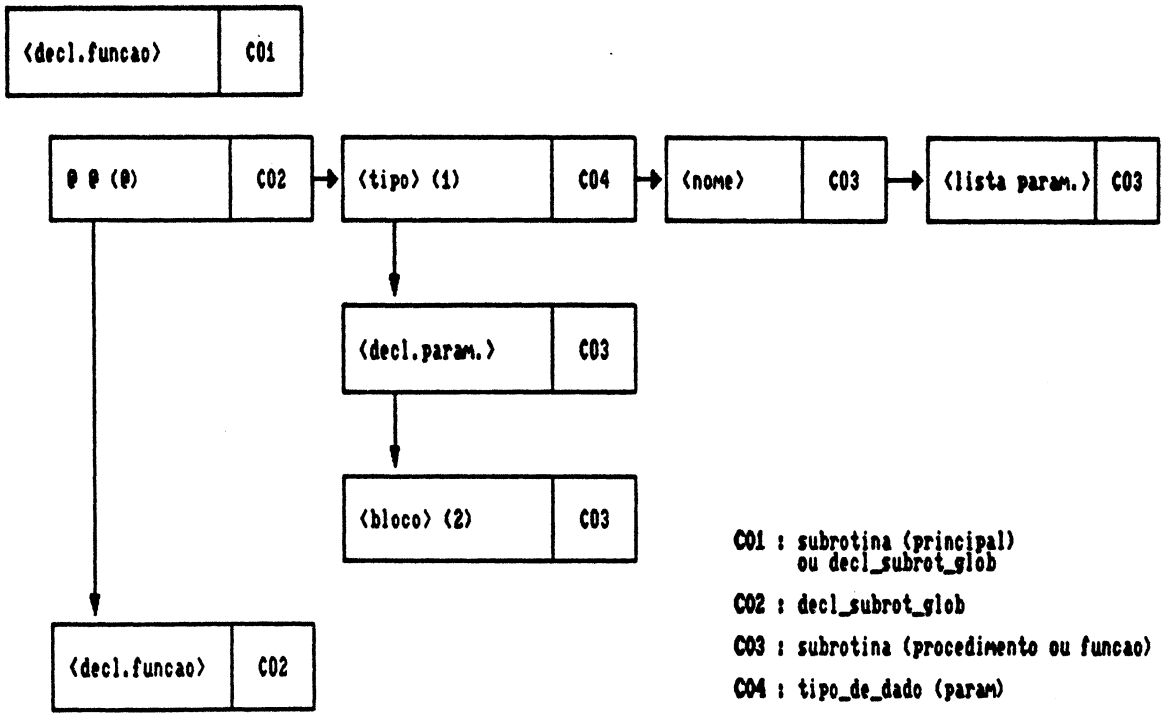
C02 : bloco (composto (2))

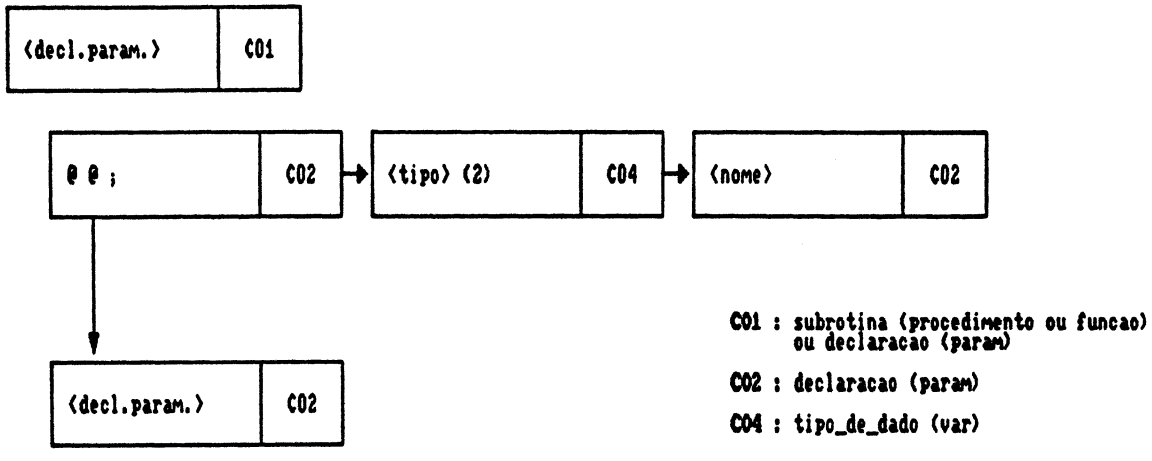


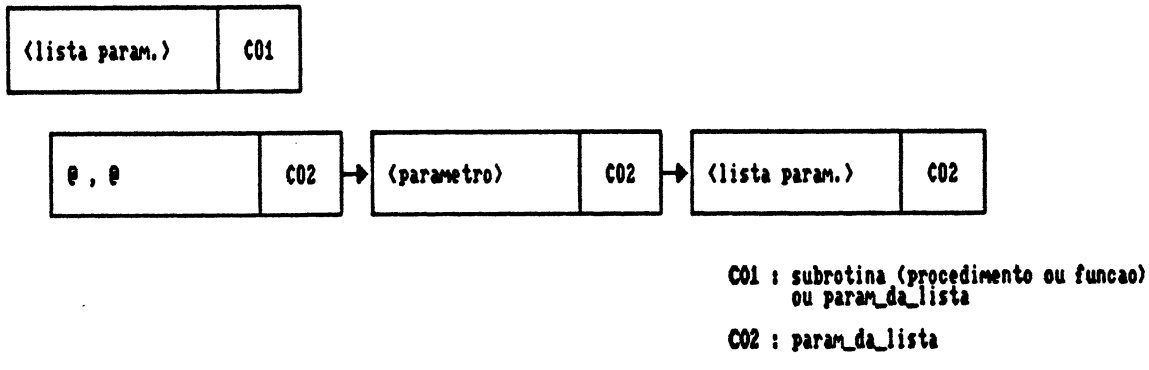


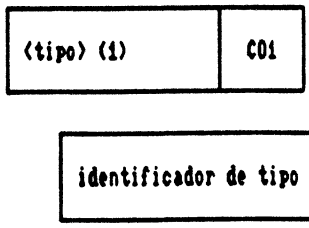
C01 : comando (repeat, repeat_r, case ou case_r)

C02 : bloco (composto (3))

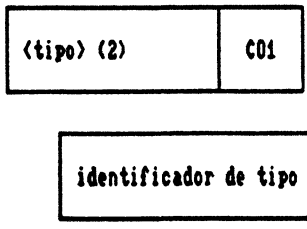

⟨decl.funcao⟩ C01
@ @ (@) C02
⟨tipo⟩ (1) C04
⟨nome⟩ C03
⟨lista param.⟩ C03
⟨decl.param.⟩ C03
⟨bloco⟩ (2) C03
⟨decl.funcao⟩ C02
C01 : subrotina (principal) ou decl_subrot_glob
C02 : decl_subrot_glob
C03 : subrotina (procedimento ou funcao)
C04 : tipo_de_dado (param)


⟨decl.param.⟩ C01
@ @ ; C02
⟨tipo⟩ (2) C04
⟨nome⟩ C02
⟨decl.param.⟩ C02
C01 : subrotina (procedimento ou funcao) ou declaracao (param)
C02 : declaracao (param)
C04 : tipo_de_dado (var)


⟨lista param.⟩ C01
@ , @ C02
⟨parametro⟩ C02
⟨lista param.⟩ C02
C01 : subrotina (procedimento ou funcao) ou param_da_lista
C02 : param_da_lista

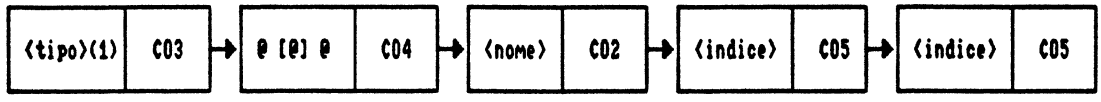


C01 : tipo_de_dado (param)
C02 : tipo_basico ou ident_tipo_novo



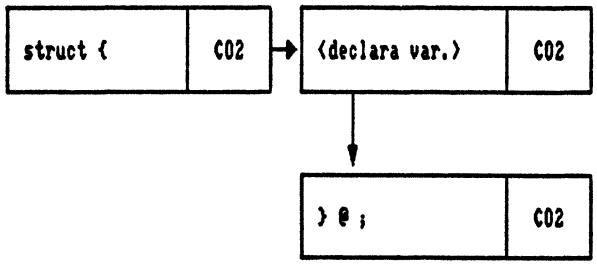
C01 : tipo_de_dado (var)
C02 : tipo_basico ou ident_tipo_novo

Vetor:

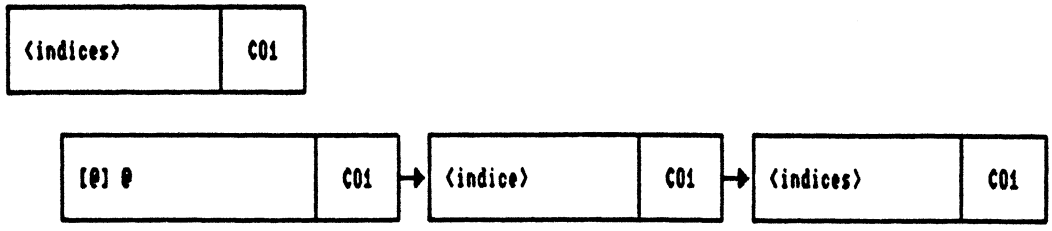


C02 : declaracao (var) ou def_tipo
C03 : tipo_de_dado (param)
C04 : array
C05 : indice

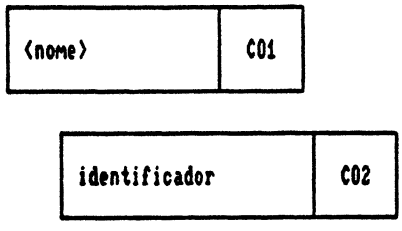
Registro:



C02 : registro



C01 : indice

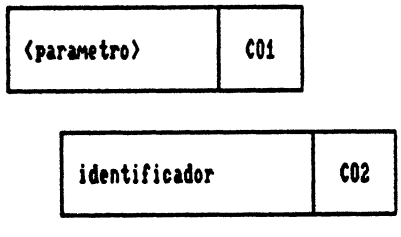


C01 :

. def_const,
. def_tipo,
. declaracao (var),
. subrotina (proc. ou funcao),
. declaracao (param),
. comando (cham_sub),
. comando (cham_sub_r).

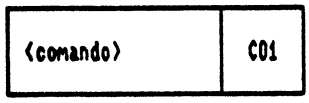
C02 :

. nome_const,
. ident_tipo_novo,
. nome_var,
. nome_subrot,
. nome_parametro,
. nome_subrot,
. nome_subrot.



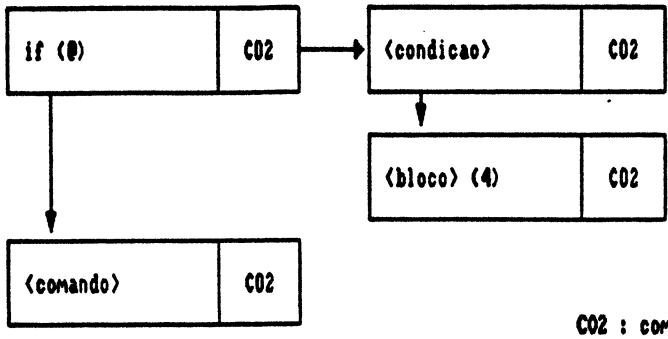
C01 : param_da_lista

C02 : nome_parametro



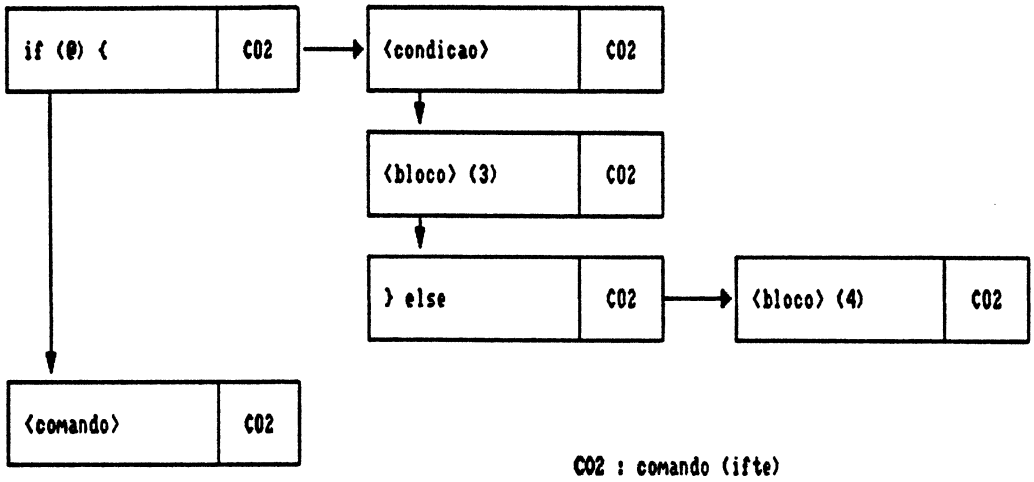
C01 : comando ou bloco (composto)

ift:



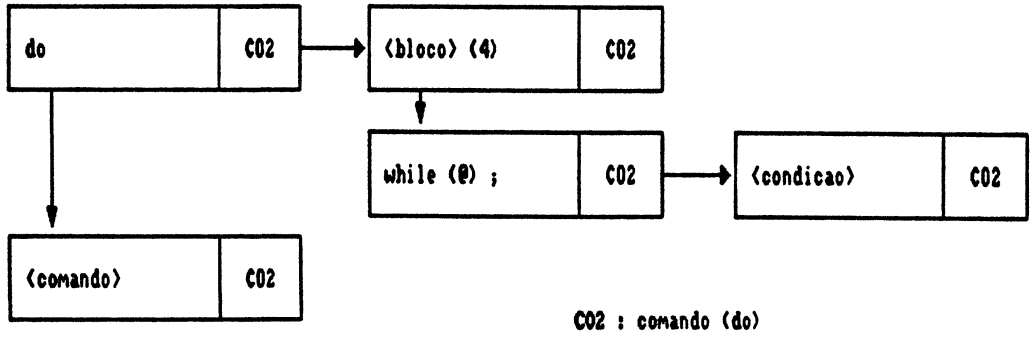
C02 : comando (ift)

ifte:



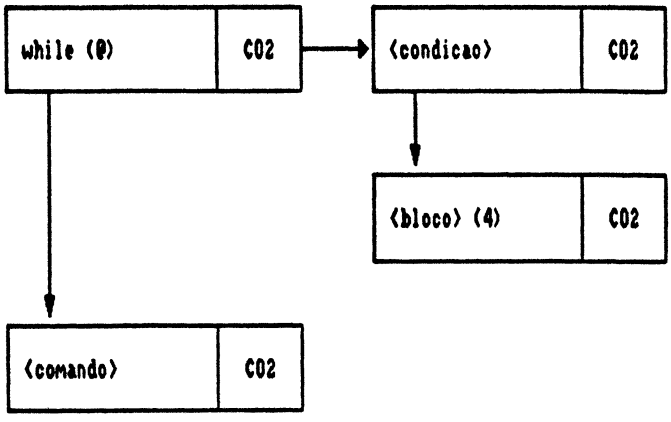
C02 : comando (ifte)

do:



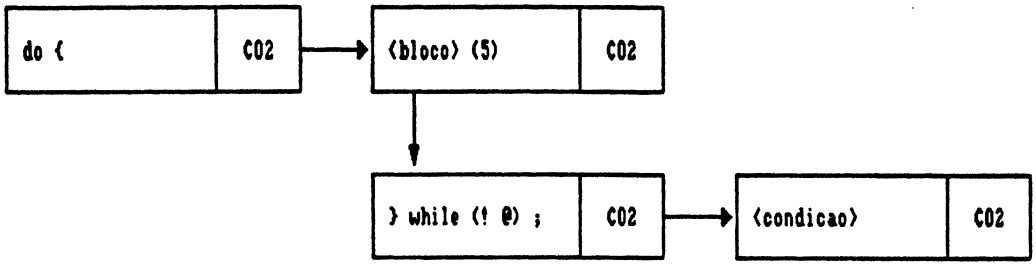
C02 : comando (do)

while:



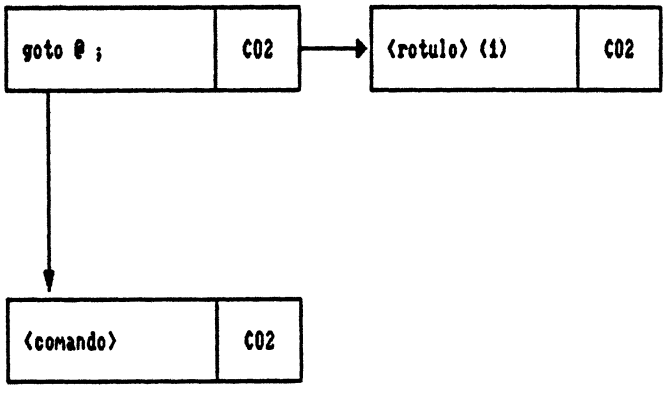
C02 : comando (while)

repeat:



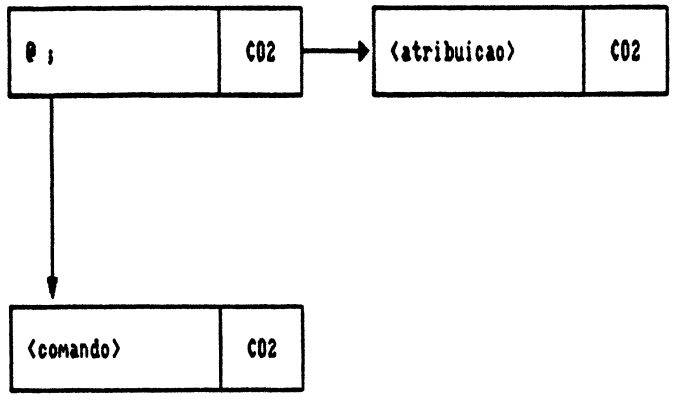
C02 : comando (repeat)

goto:



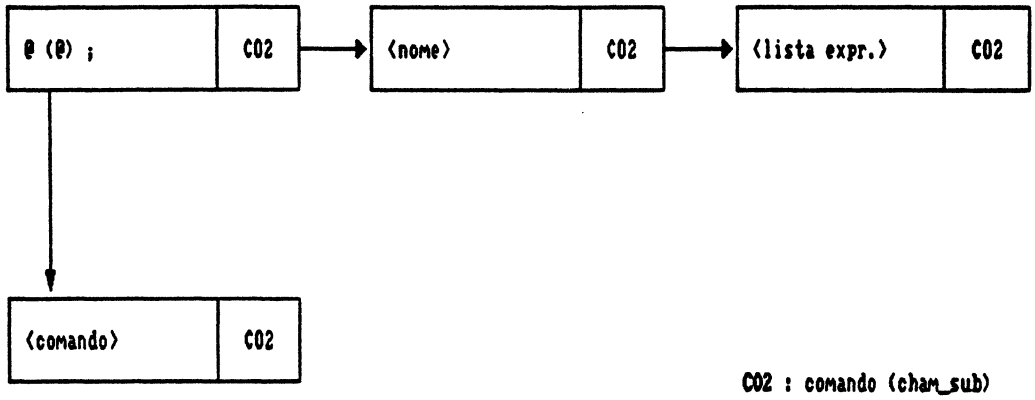
C02 : comando (goto)

atrib:



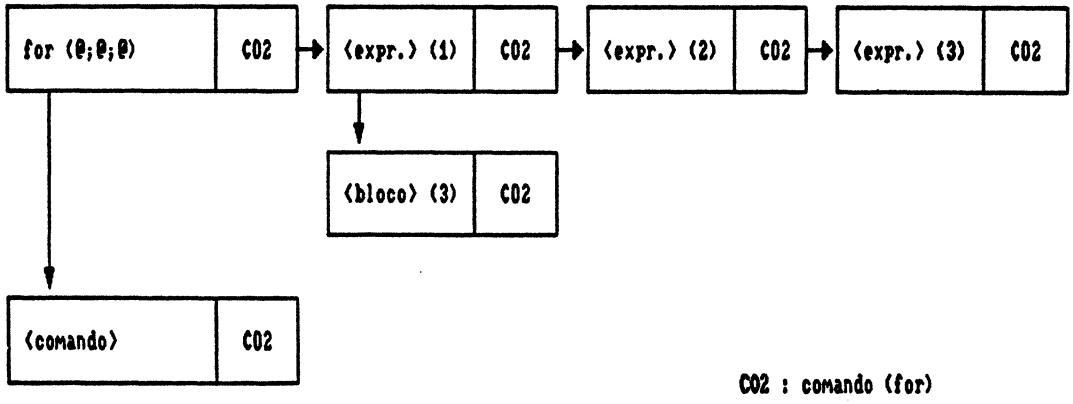
C02 : comando (atrib)

cham_sub:



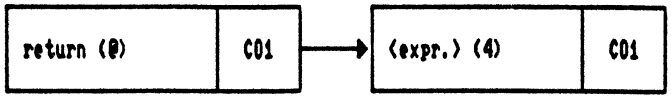
C02 : comando (cham_sub)

for:



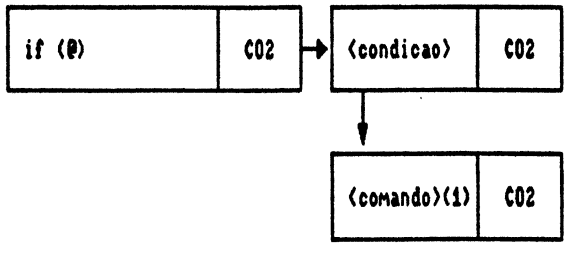
C02 : comando (for)

return:



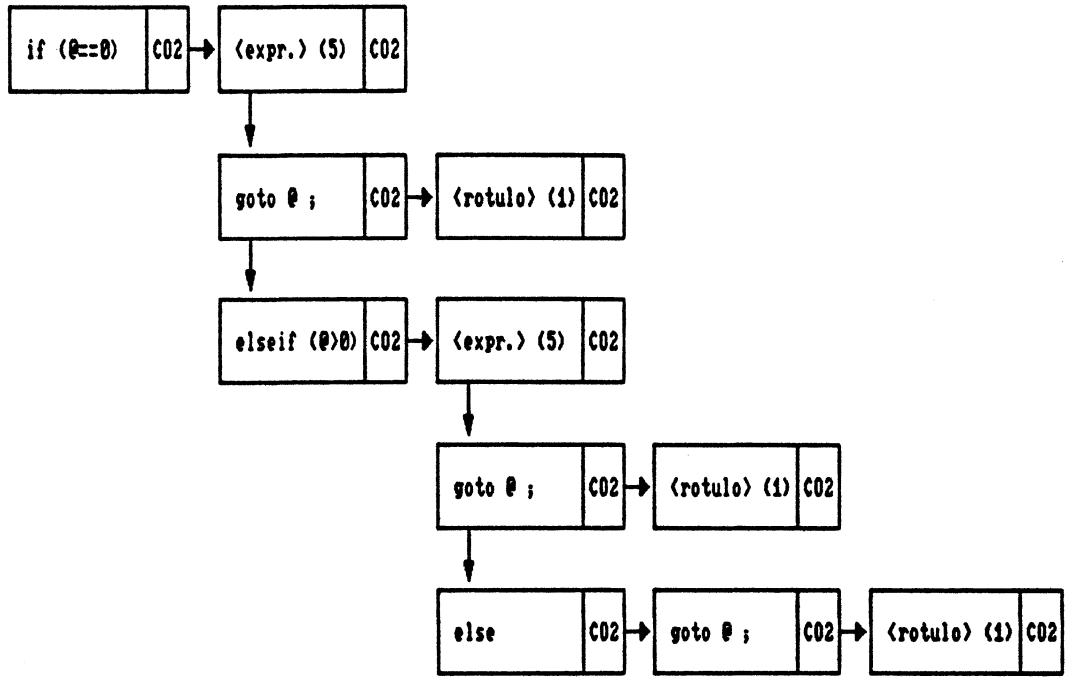
C01 : comando (return)

if1:

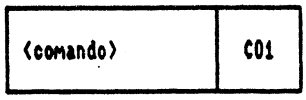


C02 : comando (if1)

ifa:

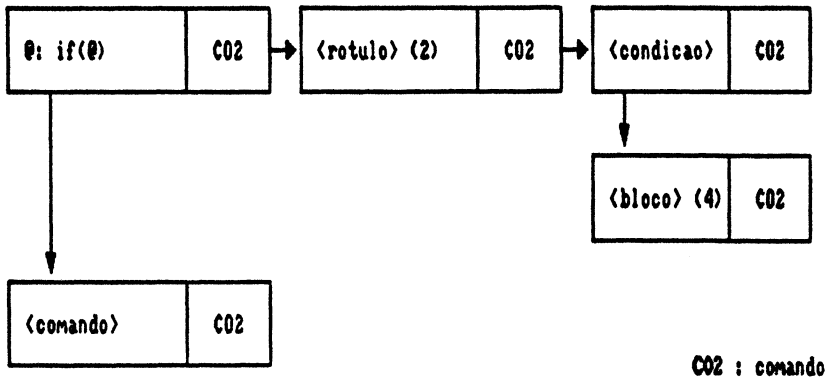


C02 : comando (ifa)



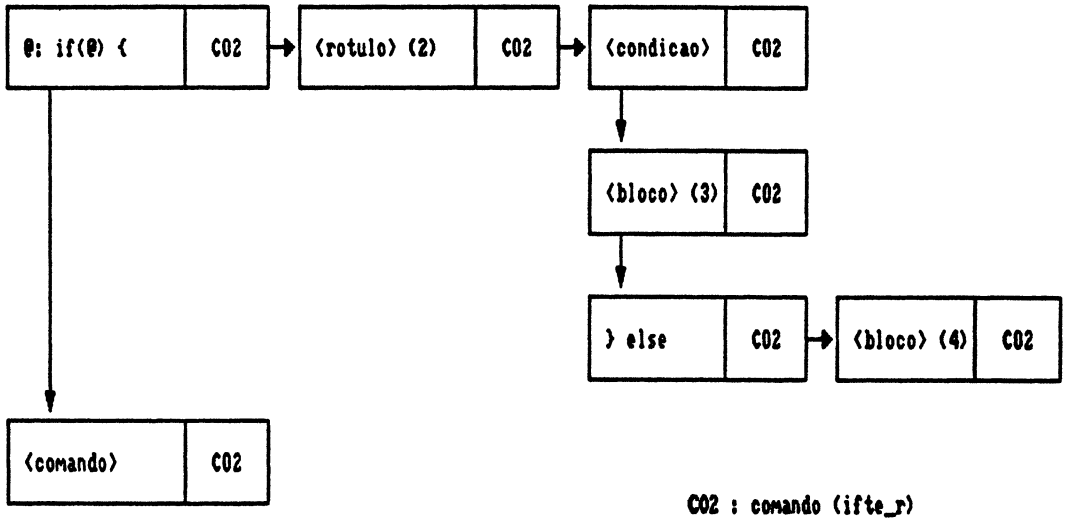
C01 : comando ou bloco

ift_r:



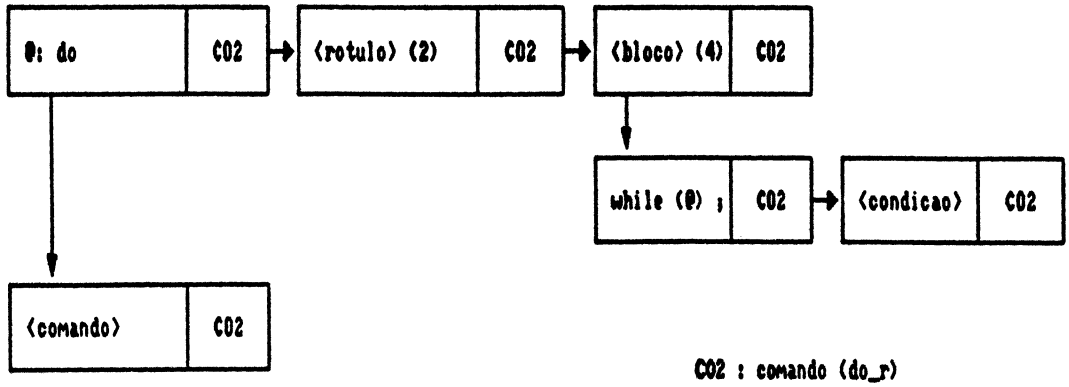
C02 : comando (ift_r)

ifte_r:



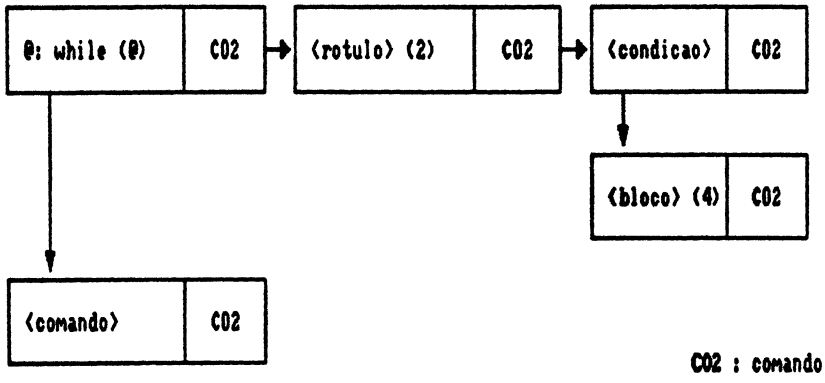
C02 : comando (ifte_r)

do_r:



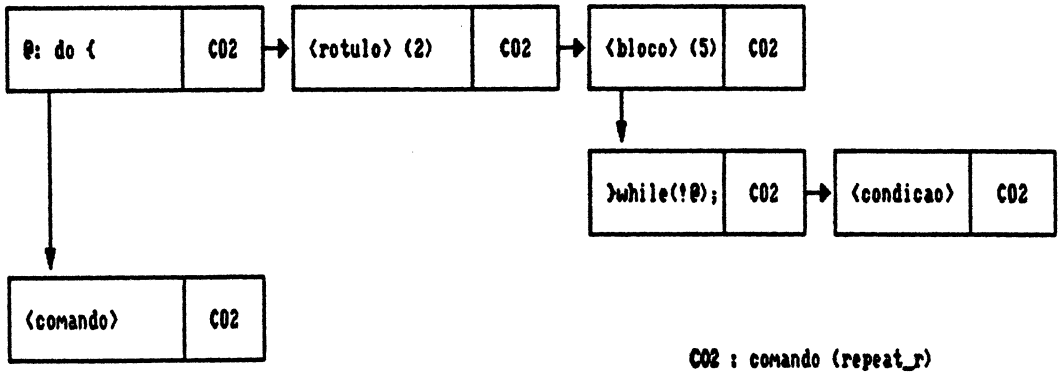
C02 : comando (do_r)

while_r:



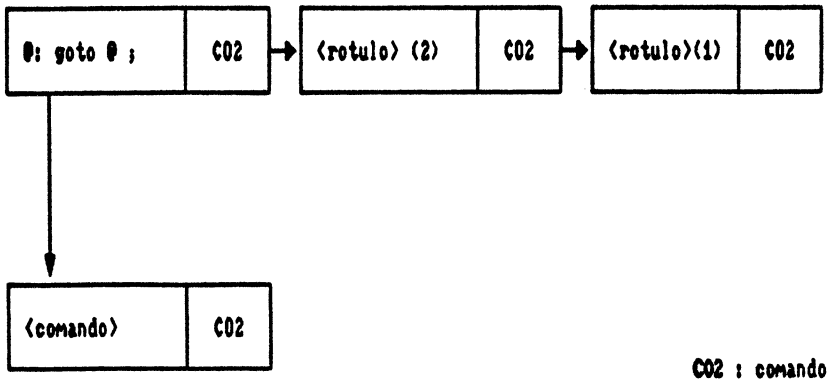
C02 : comando (while_r)

repeat_r:



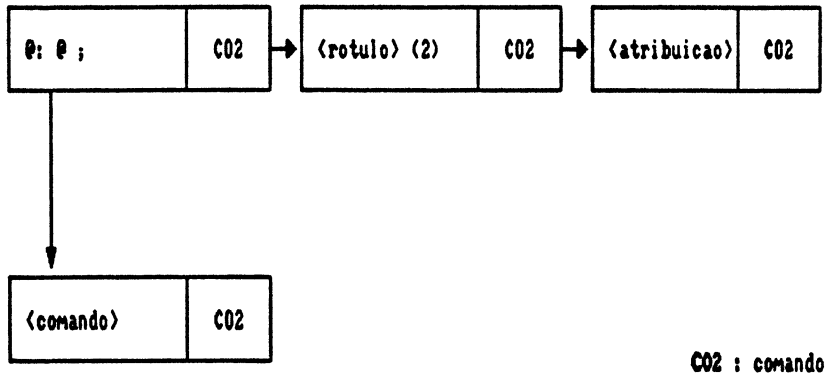
C02 : comando (repeat_r)

goto_r:



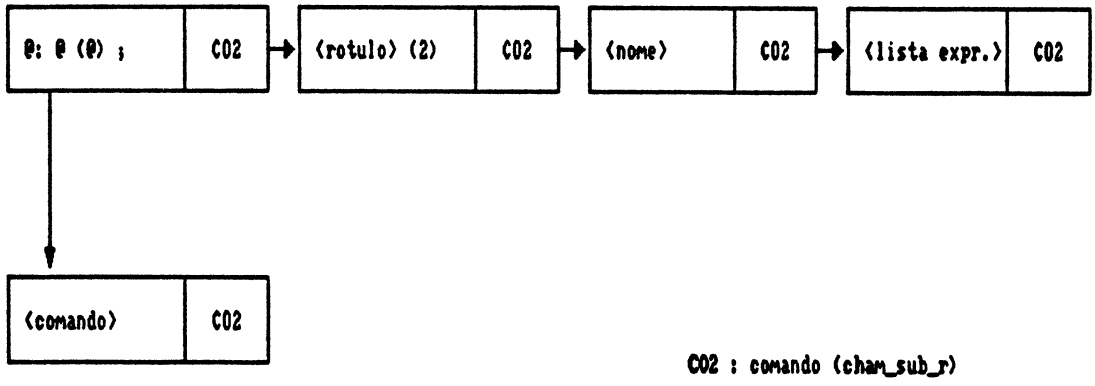
C02 : comando (goto_r)

atrib_r:



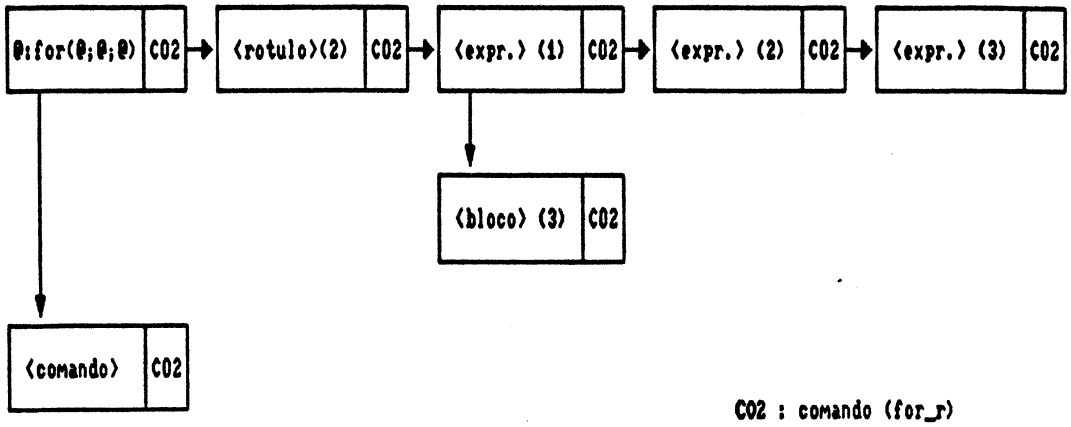
C02 : comando (atrib_r)

cham_sub_r:



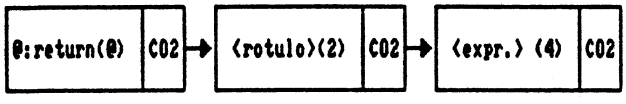
C02 : comando (cham_sub_r)

for_r:

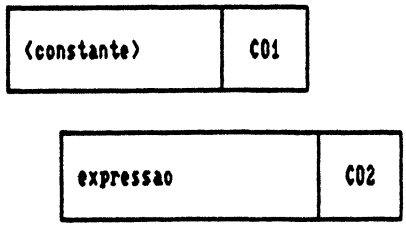


C02 : comando (for_r)

return_r:

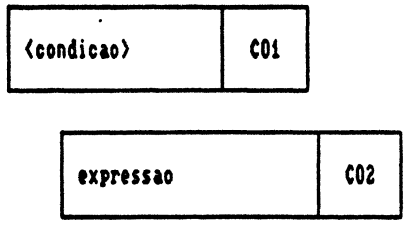


C02 : comando (return_r)



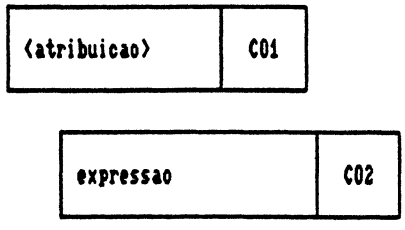
C01 : subrotina ou bloco ou def_const

C02 : expressao (constante)



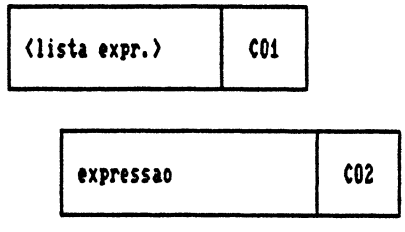
C01 : comando (ift, ift_r, ifte, ifte_r, do, do_r, while, while_r, repeat, repeat_r, ifl, ifl_r)

C02 : expressao (condicao)



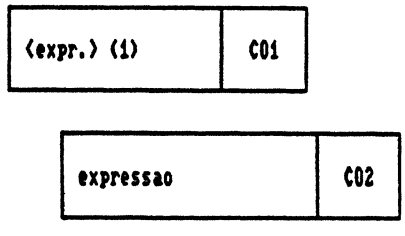
C01 : comando (atrib ou atrib_r)

C02 : expressao (atribuicao)



C01 : comando (cham_sub ou cham_sub_r)

C02 : expressao (lista_expr)



C01 : comando (for ou for_r)

C02 : expressao (inicio)

⟨expr.⟩ (2)	C01

expressao	C02

C01 : comando (for ou for_r)
C02 : expressao (parada)

⟨expr.⟩ (3)	C01

expressao	C02

C01 : comando (for ou for_r)
C02 : expressao (incremento)

⟨expr.⟩ (4)	C01

expressao	C02

C01 : comando (return ou return_r)
C02 : expressao (retorno)

⟨expr.⟩ (5)	C01

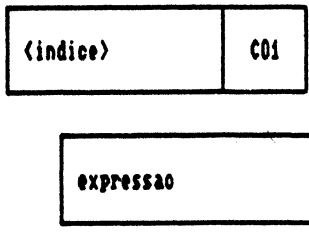
expressao	C02

C01 : comando (ifa ou ifa_r)
C02 : expressao (expr_aritm)

⟨expr.⟩ (6)	C01

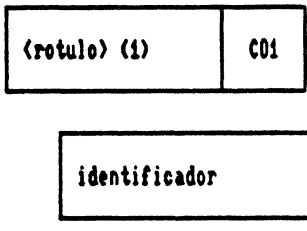
expressao	C02

C01 : comando (case ou case_r)
C02 : expressao (ativa)



C01 : indice

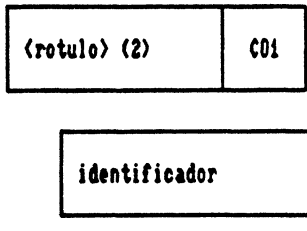
C02 : expressao (indice)



C01 : comando

C02 : rotulo

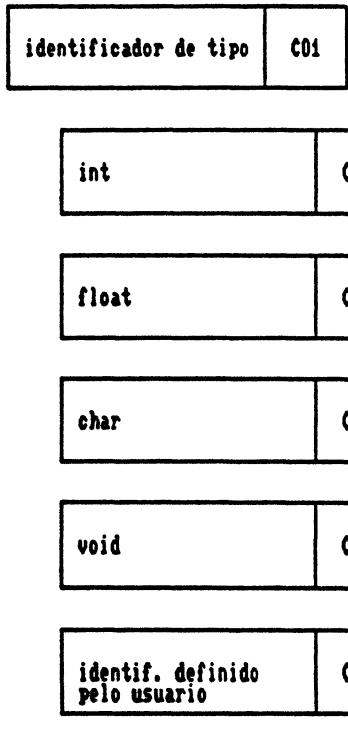
(1) : rotulo de desvio



C01 : comando

C02 : rotulo

(2) : rotulo do comando



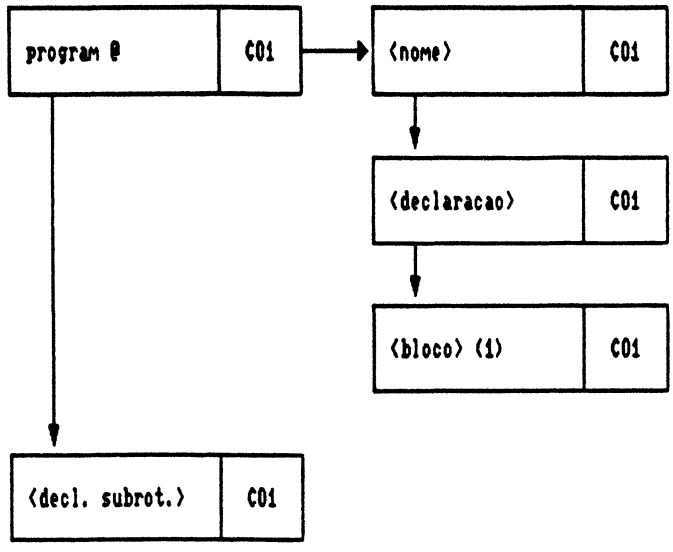
C01 : tipo_basico ou ident_tipo_novo

C02 : tipo_basico

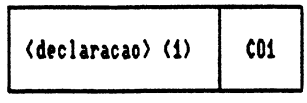
C03 : ident_tipo_novo

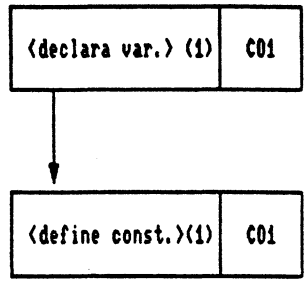
APÊNDICE A

Composições Hierárquicas dos Gabaritos da Linguagem FORTRAN

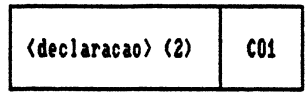


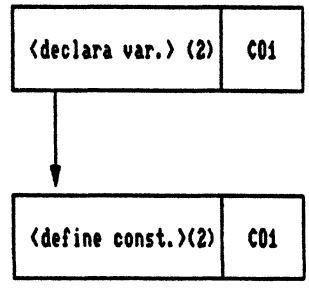
C01 : subrotina (principal)



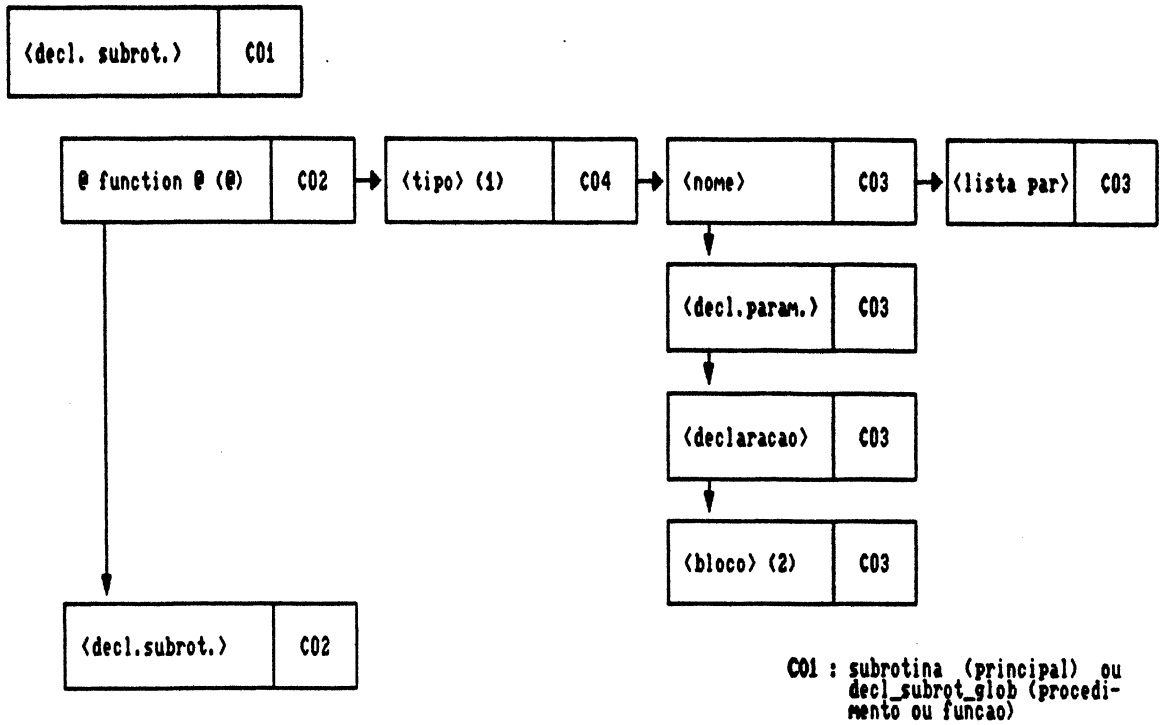


C01 : subrotina (principal)





C01 : subrotina (procedimento ou funcao)

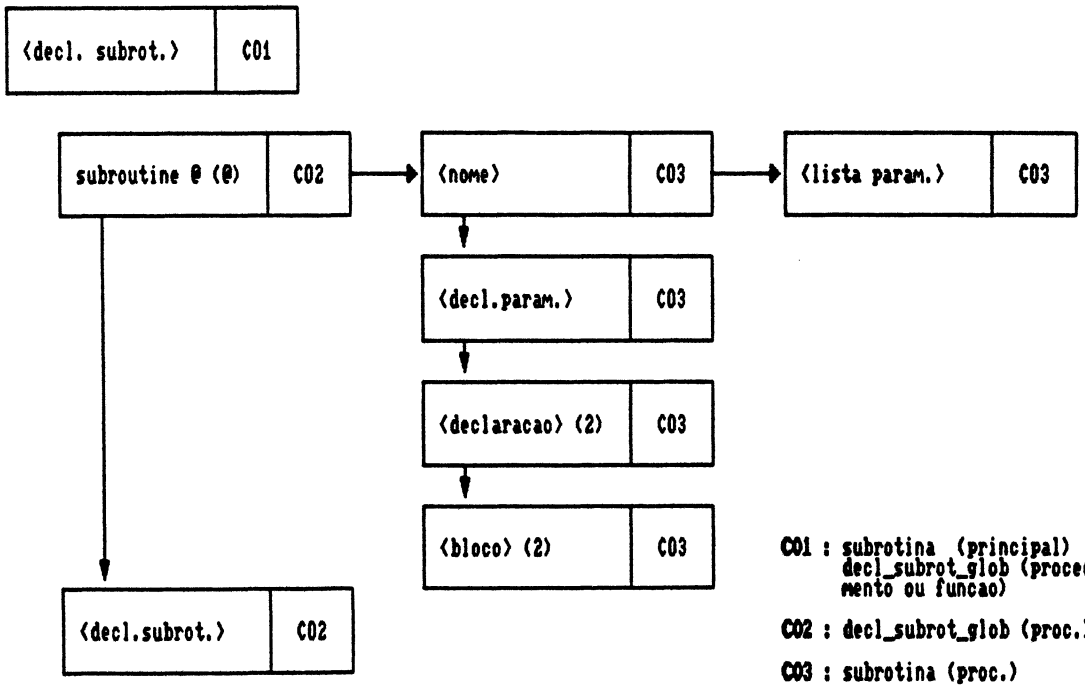


C01 : subrotina (principal) ou decl_subrot_glob (procedimento ou funcao)

C02 : decl_subrot_glob (funcao)

C03 : subrotina (funcao)

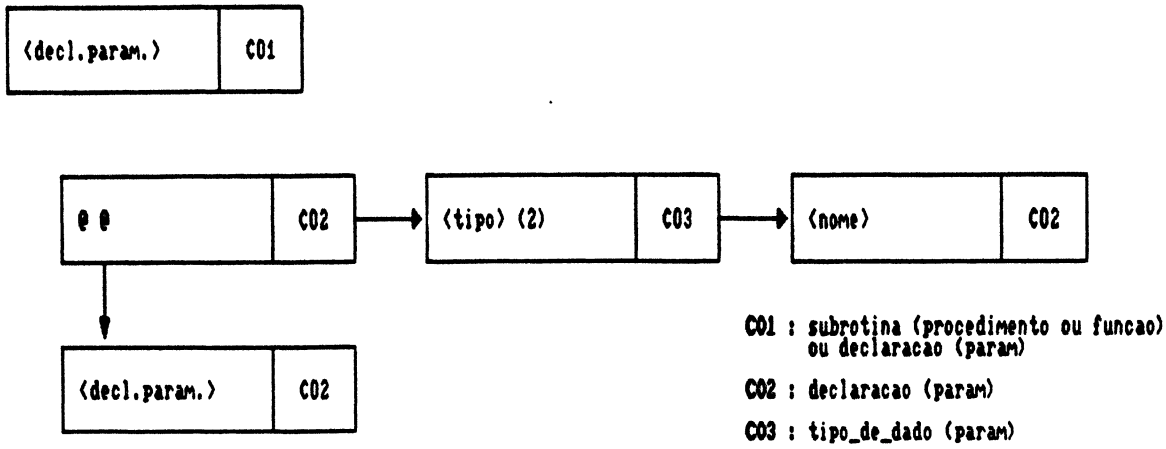
C04 : tipo_de_dado (param)

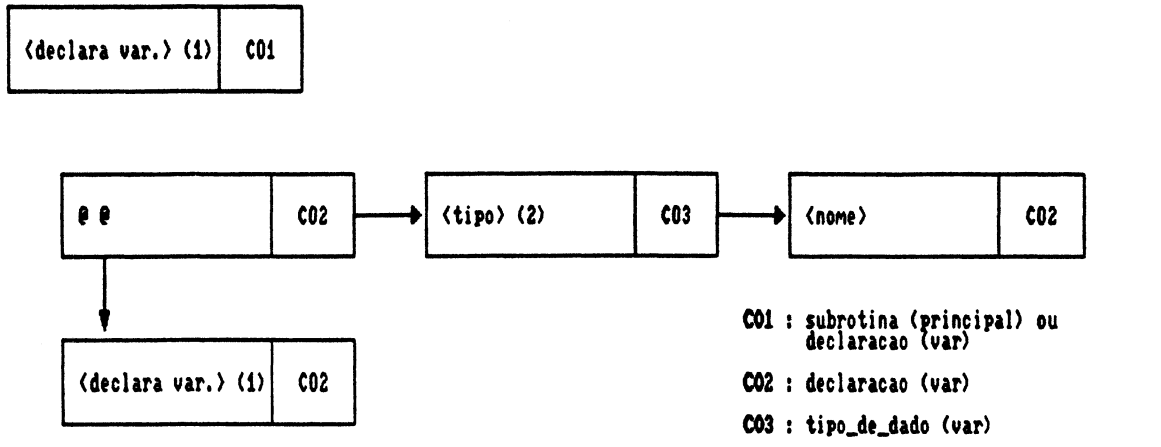


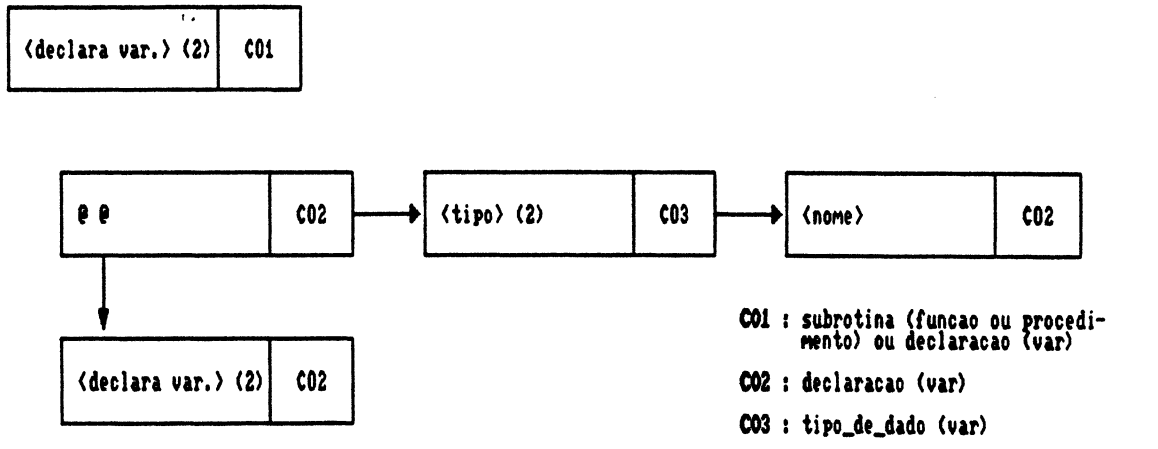
C01 : subrotina (principal) ou decl_subrot_glob (procedimento ou funcao)

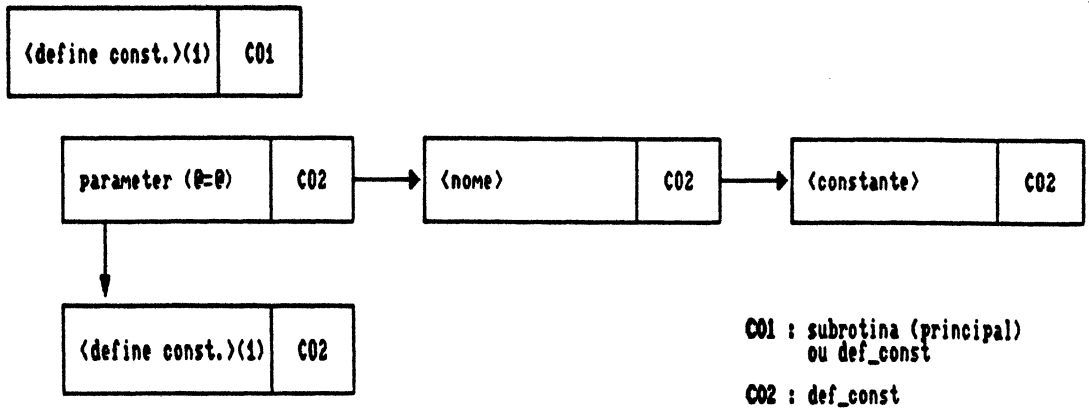
C02 : decl_subrot_glob (proc.)

C03 : subrotina (proc.)


(decl.param.)
C01
@ @
C02
(tipo) (2)
C03
(nome)
C02
(decl.param.)
C02
C01 : subrotina (procedimento ou funcao) ou declaracao (param)
C02 : declaracao (param)
C03 : tipo_de_dado (param)

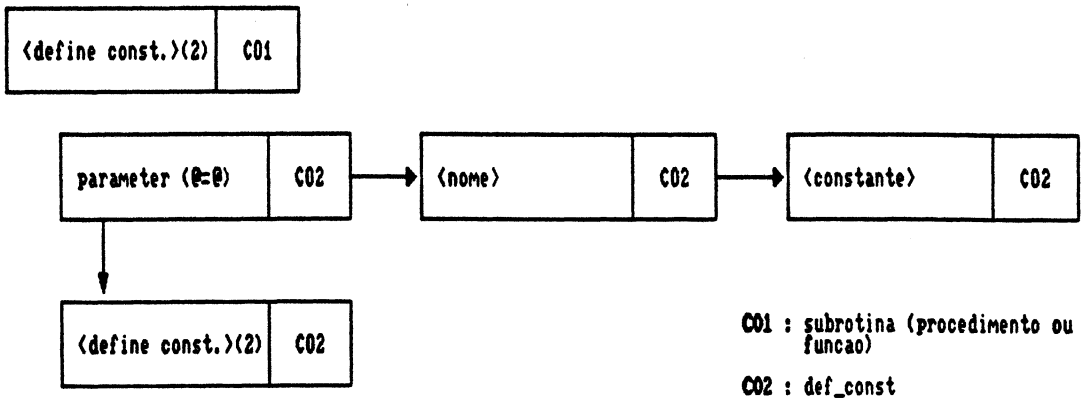

(declara var.) (1)
C01
@ @
C02
(tipo) (2)
C03
(nome)
C02
(declara var.) (1)
C02
C01 : subrotina (principal) ou declaracao (var)
C02 : declaracao (var)
C03 : tipo_de_dado (var)


(declara var.) (2)
C01
@ @
C02
(tipo) (2)
C03
(nome)
C02
(declara var.) (2)
C02
C01 : subrotina (funcao ou procedimento) ou declaracao (var)
C02 : declaracao (var)
C03 : tipo_de_dado (var)



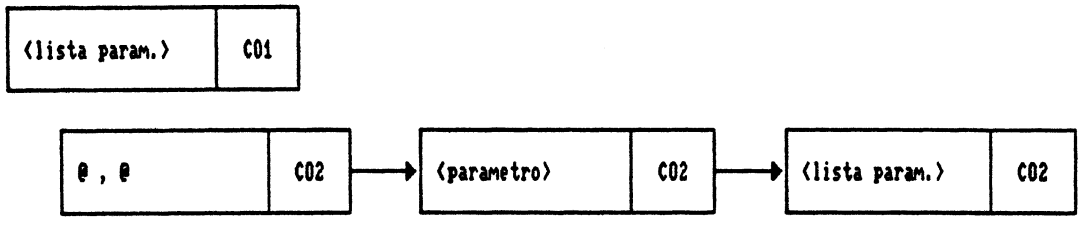
C01 : subrotina (principal) ou def_const

C02 : def_const



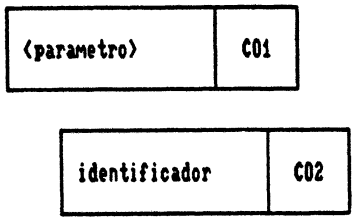
C01 : subrotina (procedimento ou funcao)

C02 : def_const



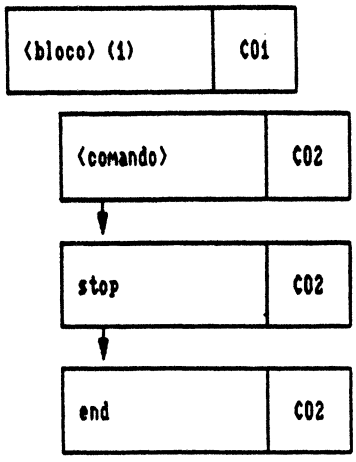
C01 : subrotina (procedimento ou funcao)

C02 : param_da_lista



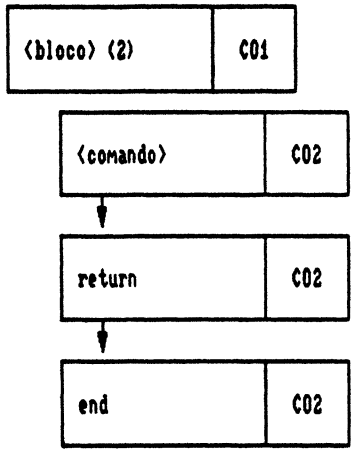
C01 : param_da_lista

C02 : nome_parametro



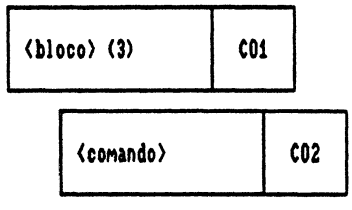
C01 : subrotina (principal)

C02 : bloco (inicio)



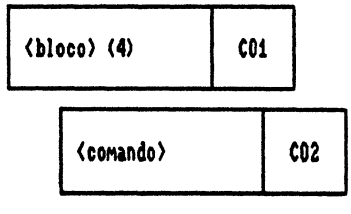
C01 : subrotina (procedimento ou funcao)

C02 : bloco (subrot)



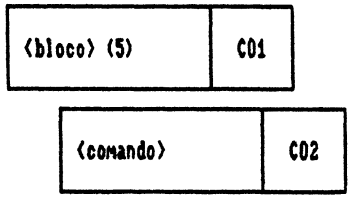
C01 : comando (ifte, ifte_r)

C02 : bloco (composto (1))



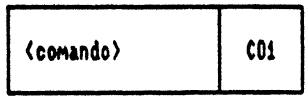
C01 : comando (ift, ift_r, ifte, ifte_r, do, do_r, while, while_r, for ou for_r)

C02 : bloco (composto (2))



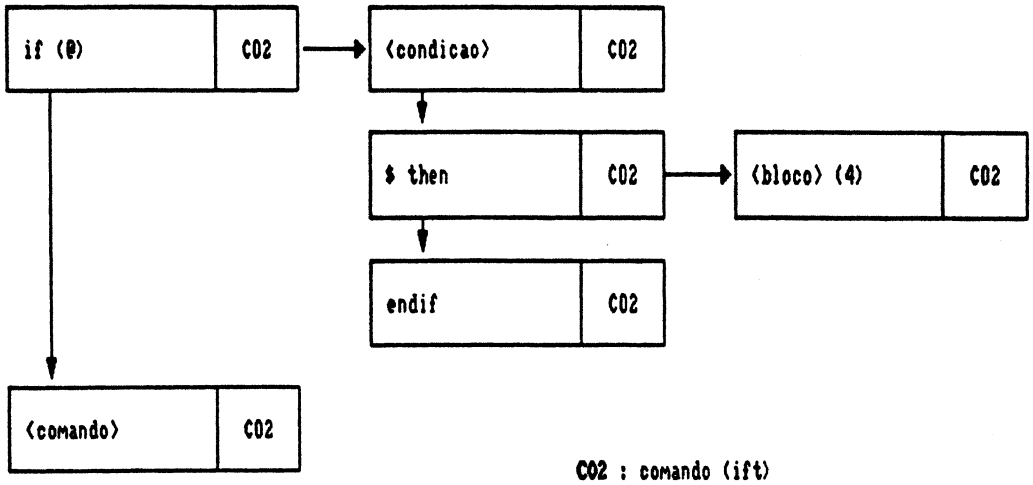
C01 : comando (repeat, repeat_r, case ou case_r)

C02 : bloco (composto (3))



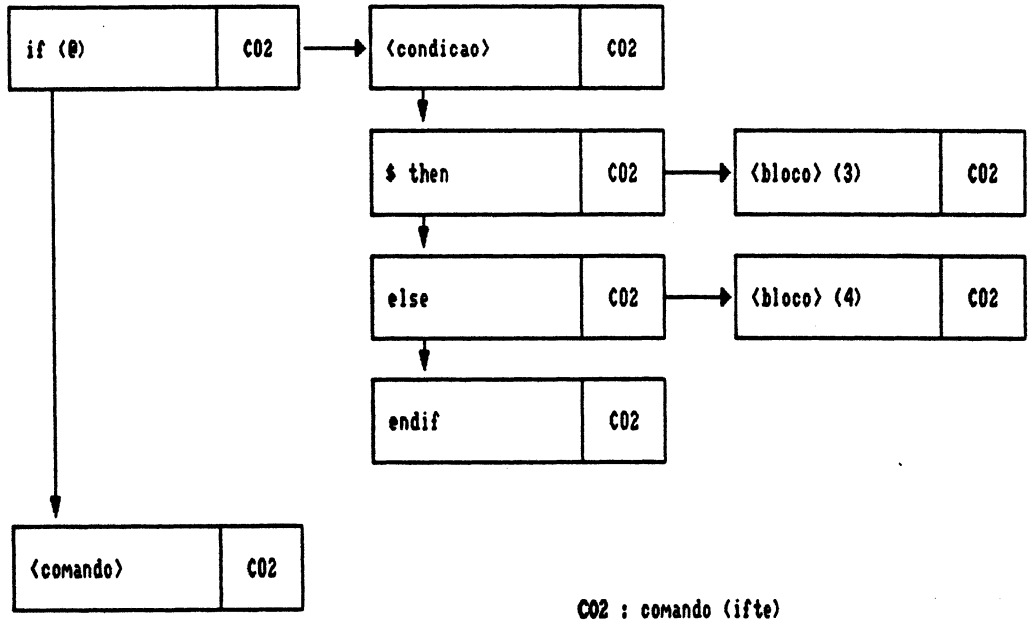
C01 : comando

ift:



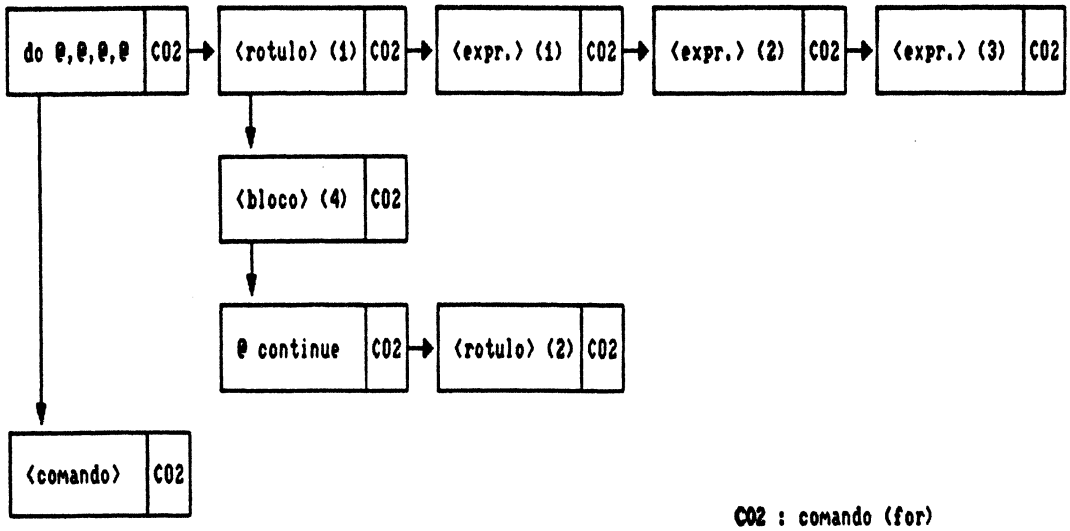
C02 : comando (ift)

ifte:



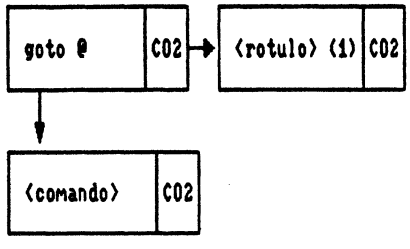
C02 : comando (ifte)

for:



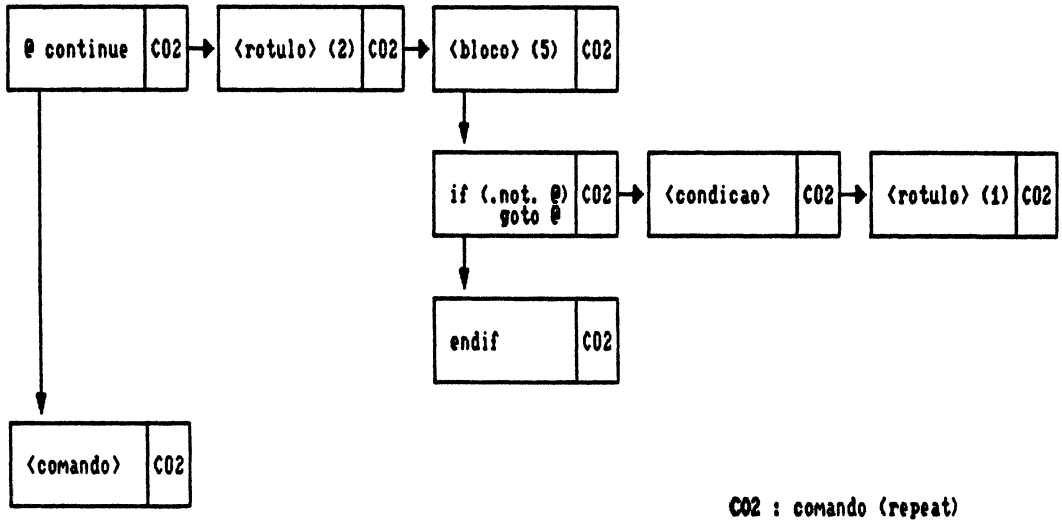
C02 : comando (for)

goto:



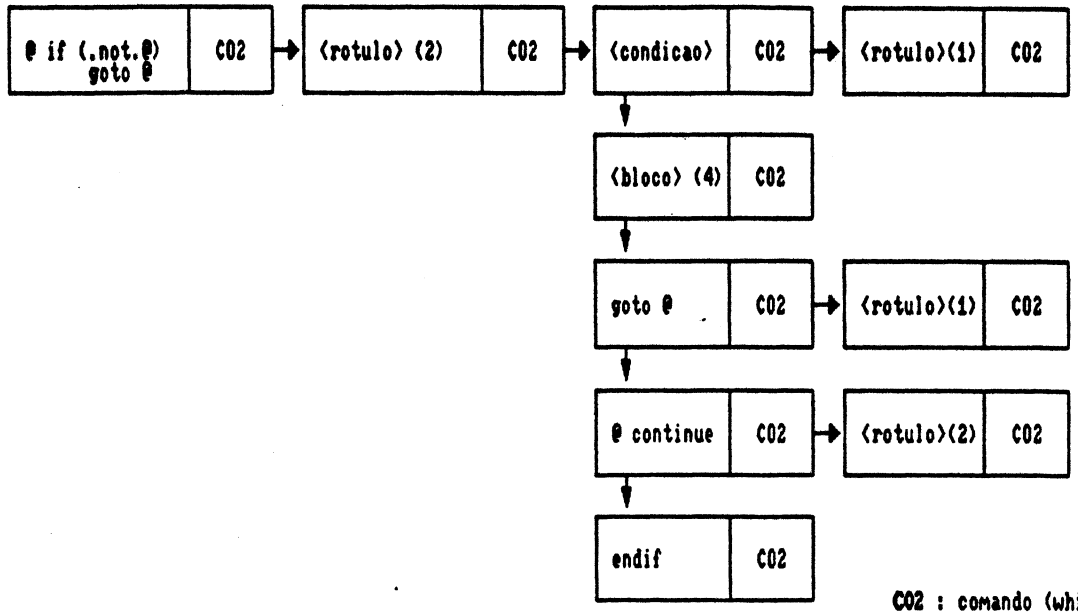
C02 : comando (goto)

repeat:



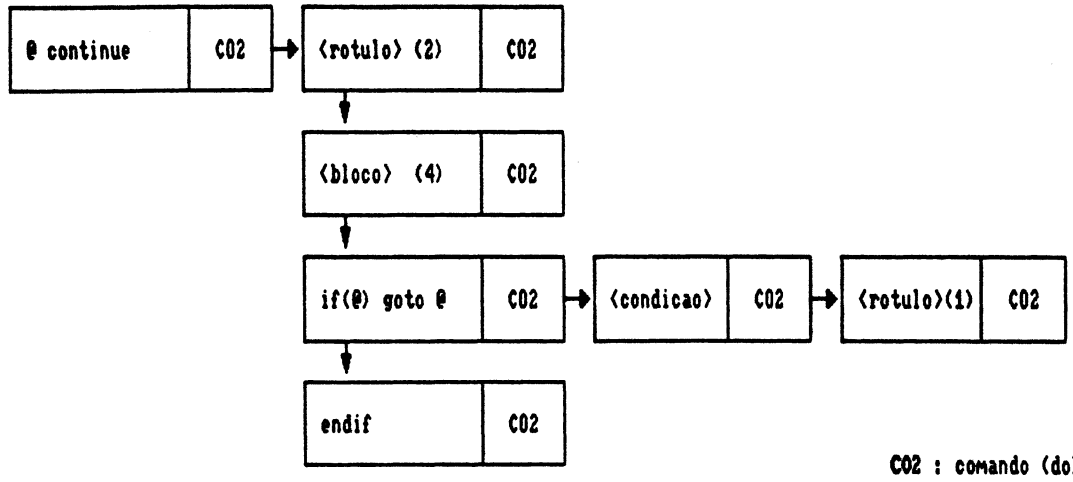
C02 : comando (repeat)

while:



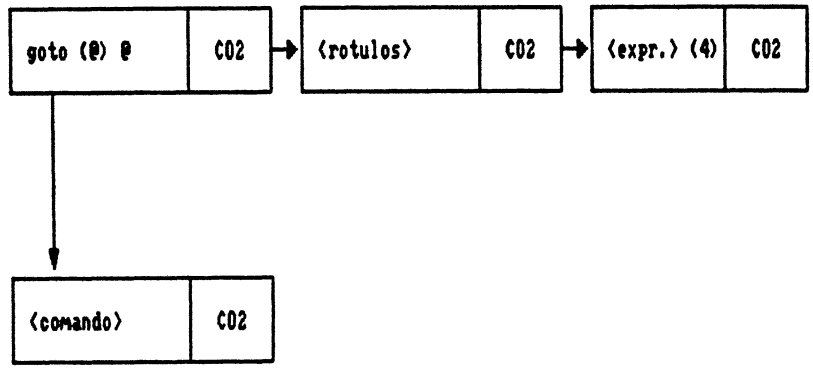
C02 : comando (while)

do:



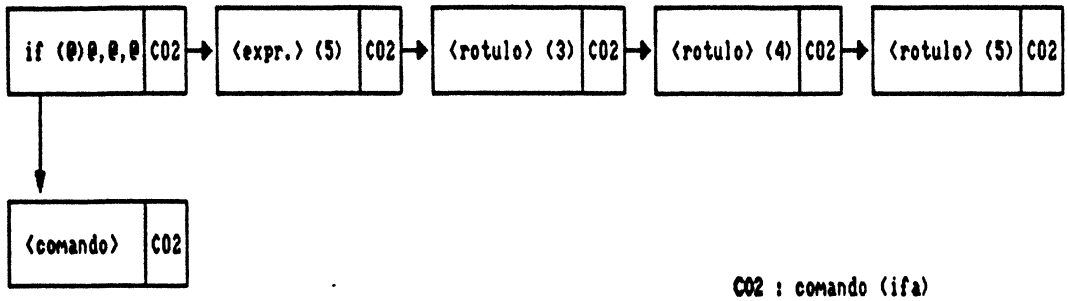
C02 : comando (do)

gotoc:



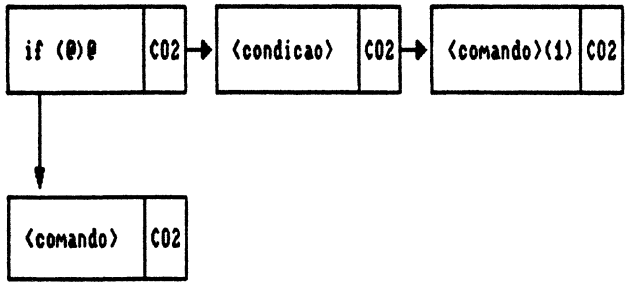
C02 : comando (gotoc)

ifa:



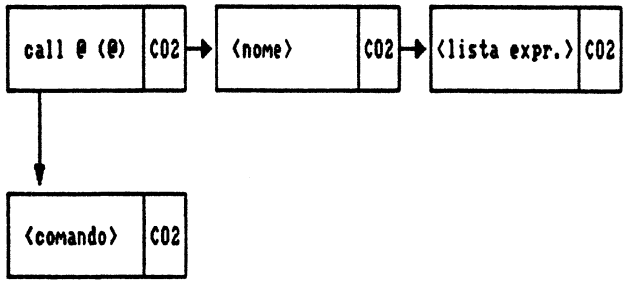
C02 : comando (ifa)

ifl:



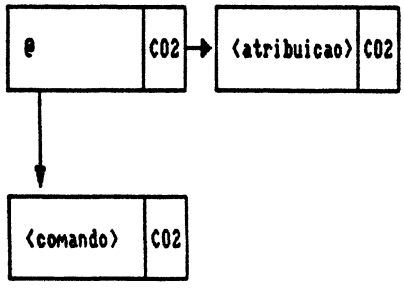
C02 : comando (ifl)

cham_sub:

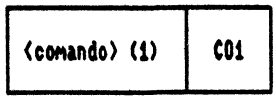


C02 : comando (cham_sub)

atrib:

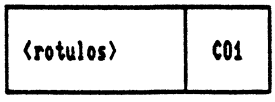


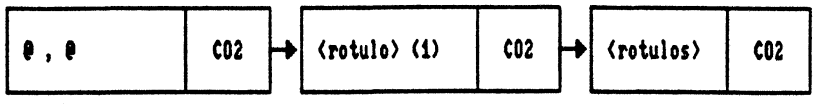
C02 : comando (atrib)



opcoes:
.goto
.gotoc
.atribuicao
.cham_subrot

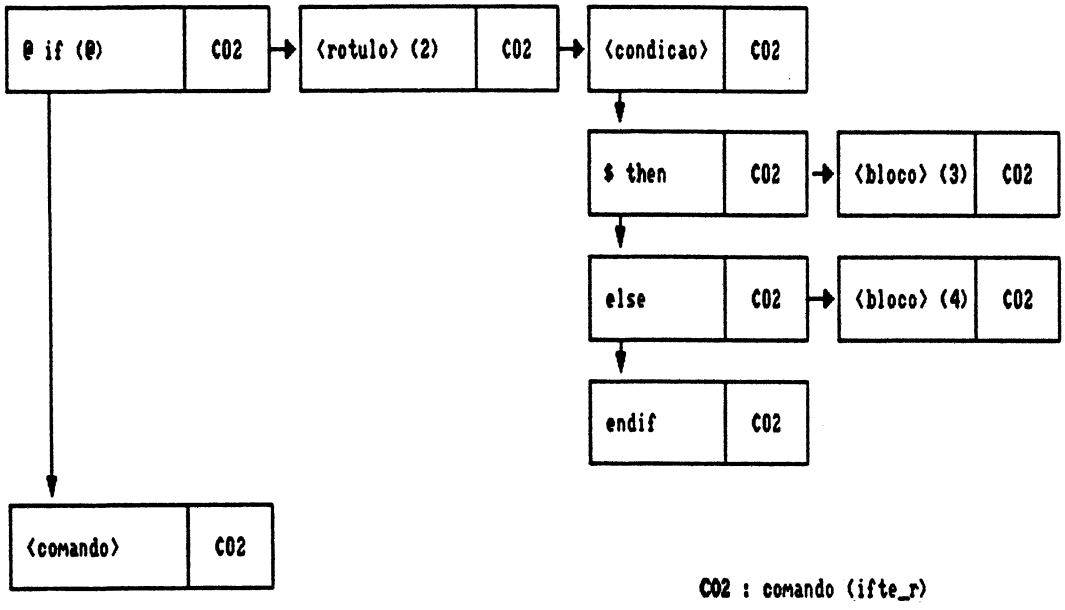
C01 : comando (ifl ou ifl_r)





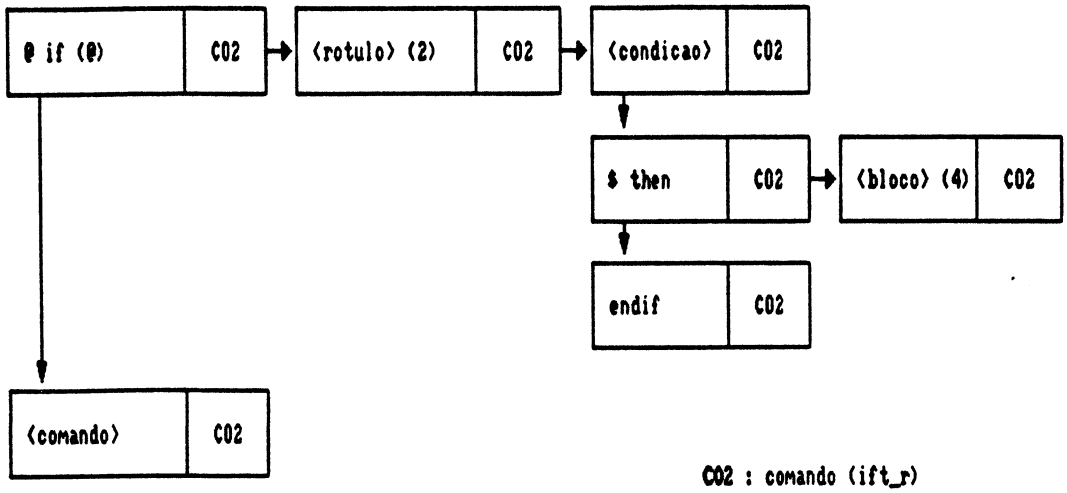
C01 : comando (gotoc) ou rotulos
C02 : rotulos

ifte_r:



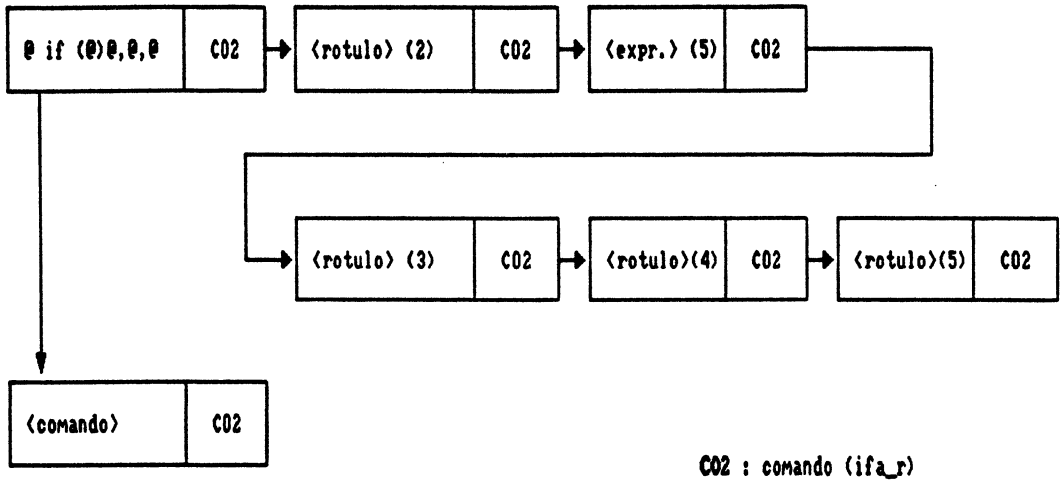
C02 : comando (ifte_r)

ift_r:



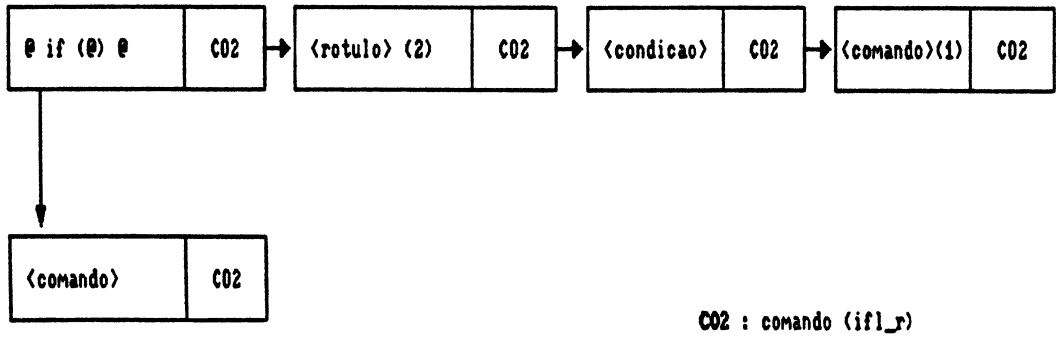
C02 : comando (ift_r)

ifa_r:



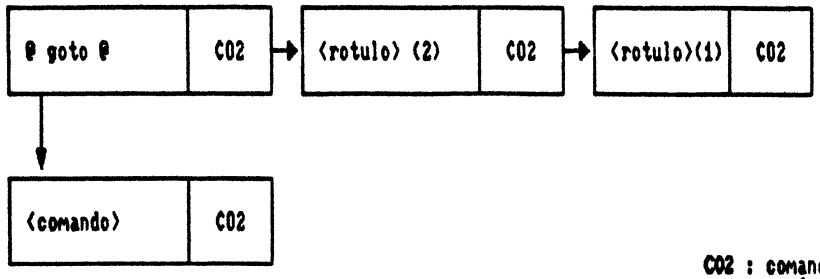
C02 : comando (ifa_r)

ifl_r:



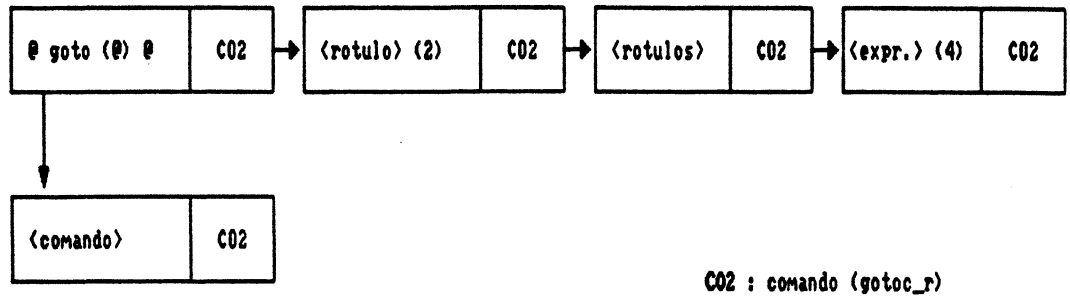
C02 : comando (ifl_r)

goto_r:



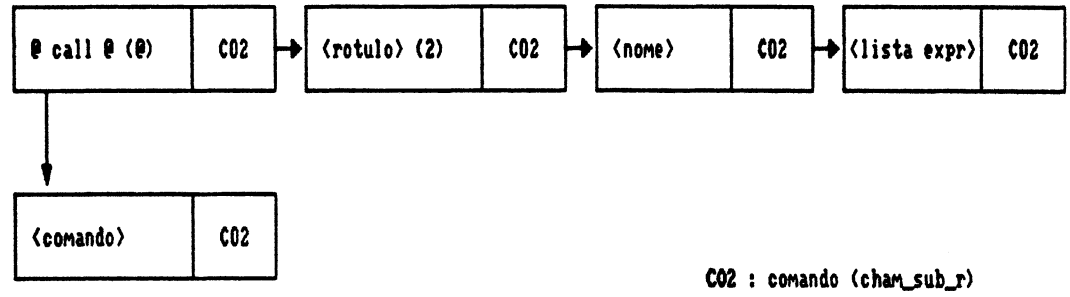
C02 : comando (goto_r)

gotoc_r:



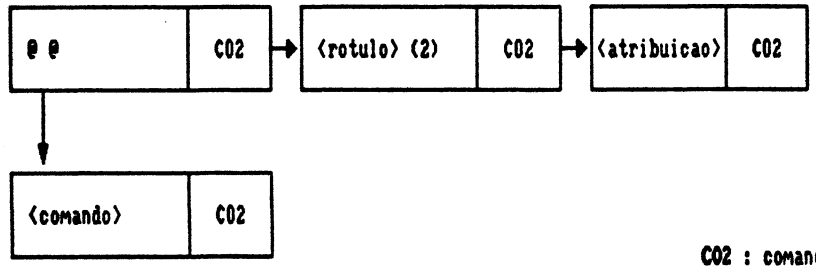
C02 : comando (gotoc_r)

cham_sub_r:



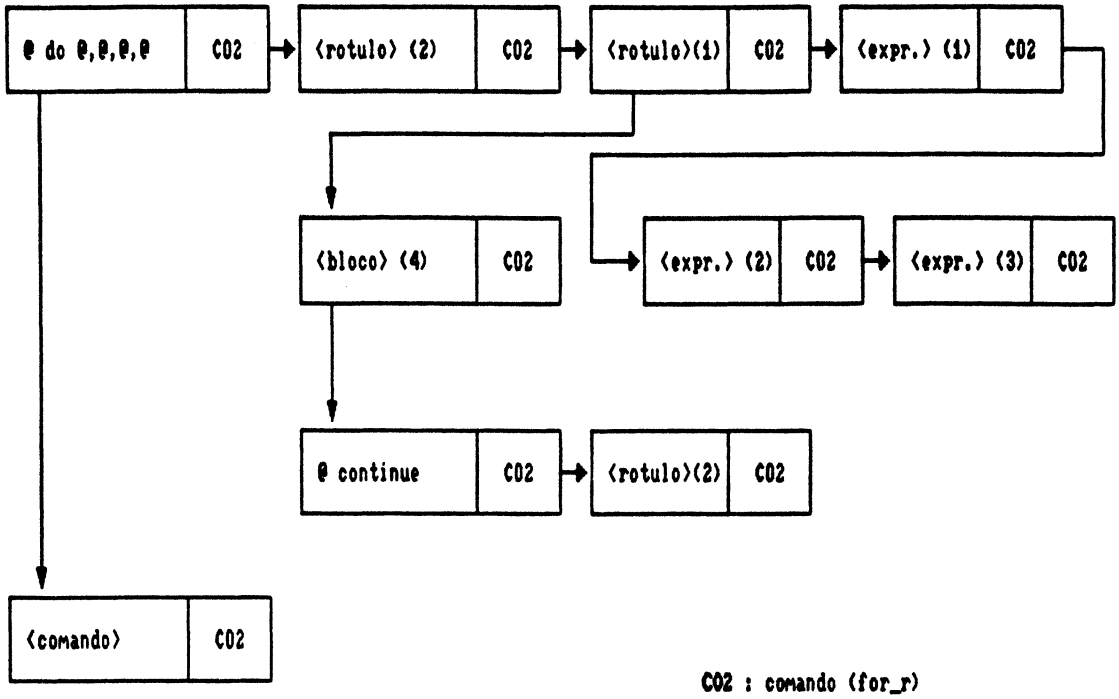
C02 : comando (cham_sub_r)

atrib_r:

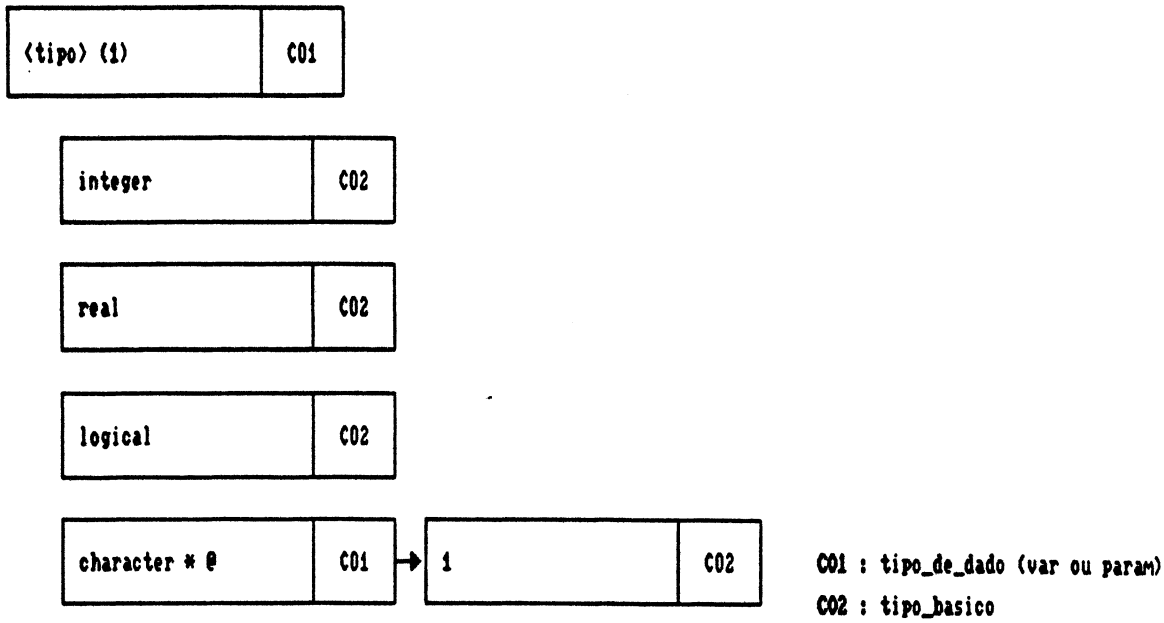


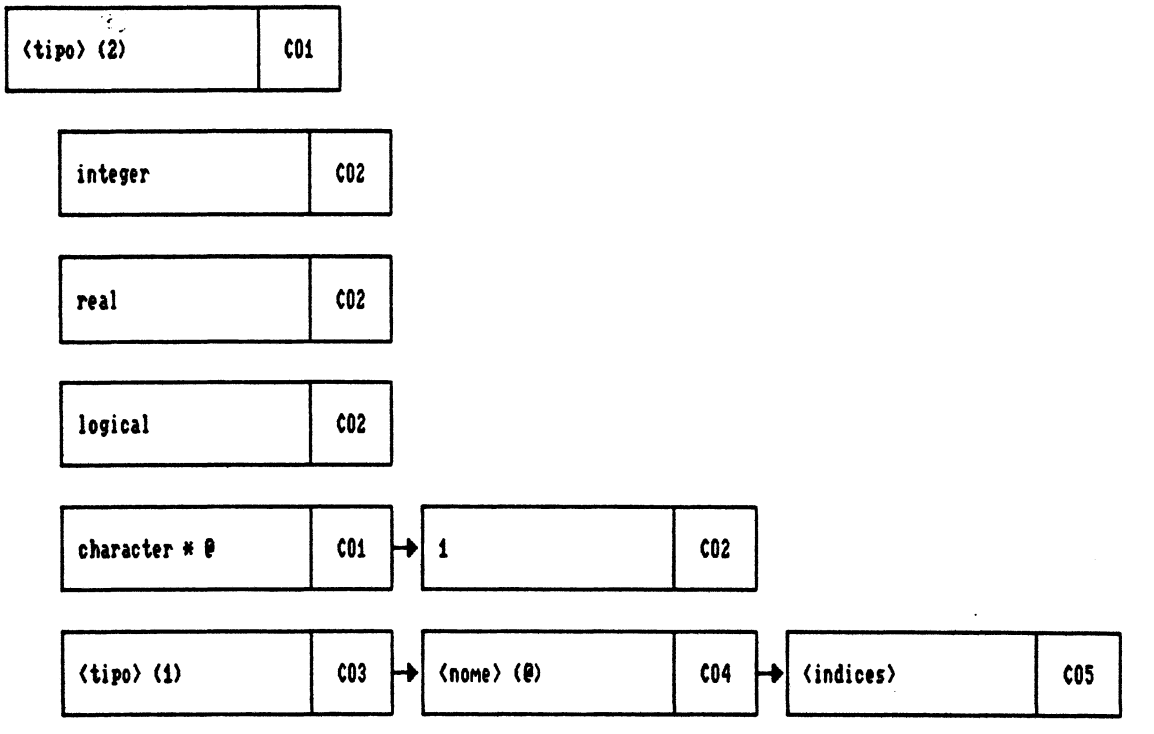
C02 : comando (atrib_r)

for_r:

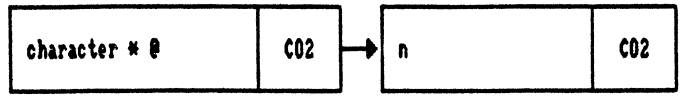


C02 : comando (for_r)


⟨tipo⟩ (1)
C01
integer
C02
real
C02
logical
C02
character * @
C01
1
C02
C01 : tipo_de_dado (var ou param)
C02 : tipo_basico


⟨tipo⟩ (2)
C01
integer
C02
real
C02
logical
C02
character * @
C01
1
C02
⟨tipo⟩ (1)
C03
⟨nome⟩ (@)
C04
⟨indices⟩
C05
C01 : tipo_de_dado (var ou param)
C02 : tipo_basico
C03 : tipo_de_dado (var)
C04 : declaracao (var)
C05 : array

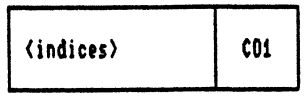
tam:



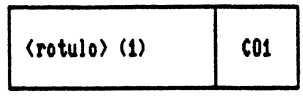
C01 : tipo_basico
C02 : charac

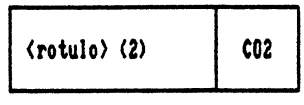
C01 : tipo_de_dado (var)
C02 : charac

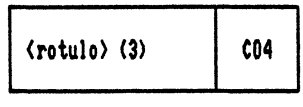
indices:

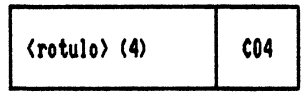


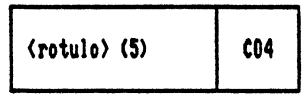
C01 : array ou indice
C02 : indice

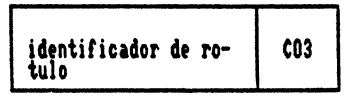




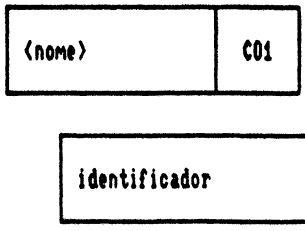








C01 : comando
C02 : comando com rotulo
C03 : rotulo
C04 : comando (ifa ou ifa_r)
(1) : rotulo de desvio
(2) : rotulo de comando

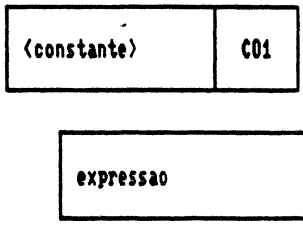


C01 :

- subrotina (principal ou função ou procedimento)
- declaracao (param)
- declaracao (var)
- def_const

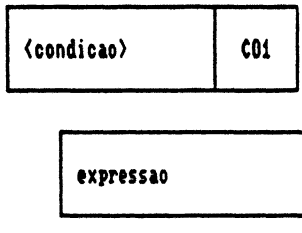
C02 :

- nome_subrot
- nome_parametro
- ident_var
- nome_const



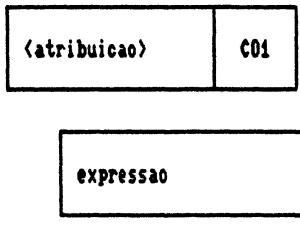
C01 : def_const

C02 : expressao (constante)



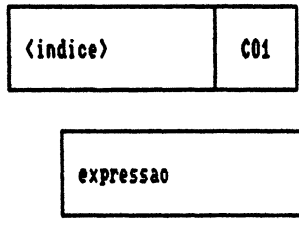
C01 : comando (ift, ifte, repeat, while, do, ifl, ift_r, ifte_r, repeat_r, while_r ou do_r)

C02 : expressao (condicao)



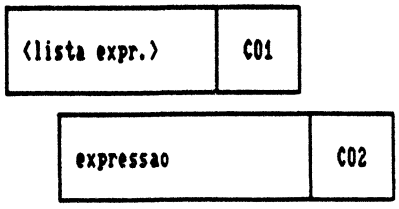
C01 : comando (atrib ou atrib_r)

C02 : expressao (atribuicao ou retorno)

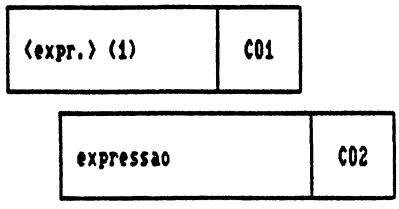


C01 : indice ou array

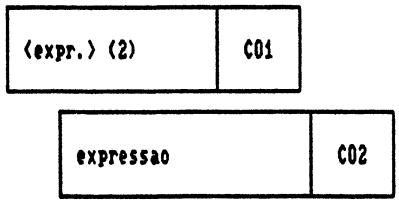
C02 : expressao (indice (indice_g ou indice_pf))



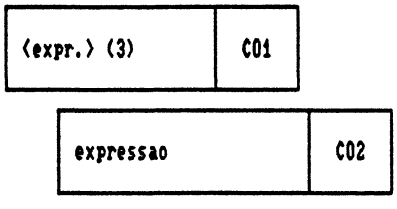
C01 : comando (cham_sub ou cham_sub_r)
C02 : expressao (lista_expr)



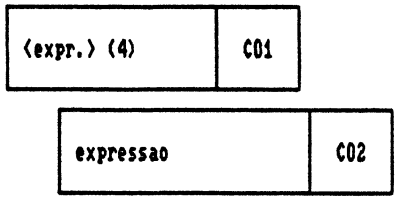
C01 : comando (for ou for_r)
C02 : expressao (inicio)



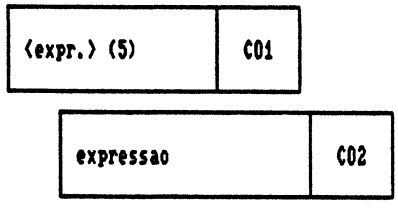
C01 : comando (for ou for_r)
C02 : expressao (parada)



C01 : comando (for ou for_r)
C02 : expressao (incremento)



C01 : comando (gotoc ou gotoc_r)
C02 : expressao (expr_gotoc)



C01 : comando (ifa ou ifa_r)
C02 : expressao (expr_aritm)

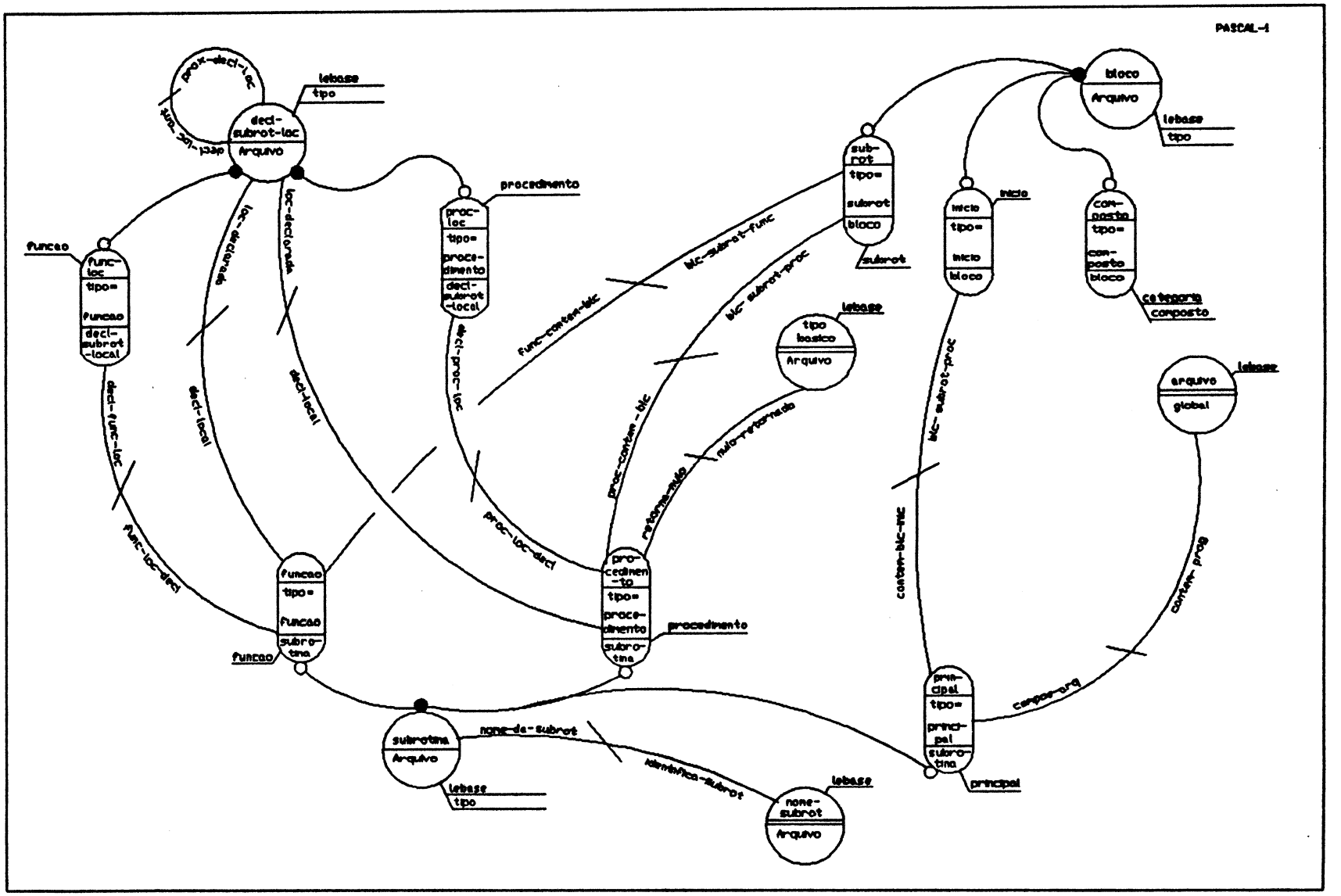
APÊNDICE B

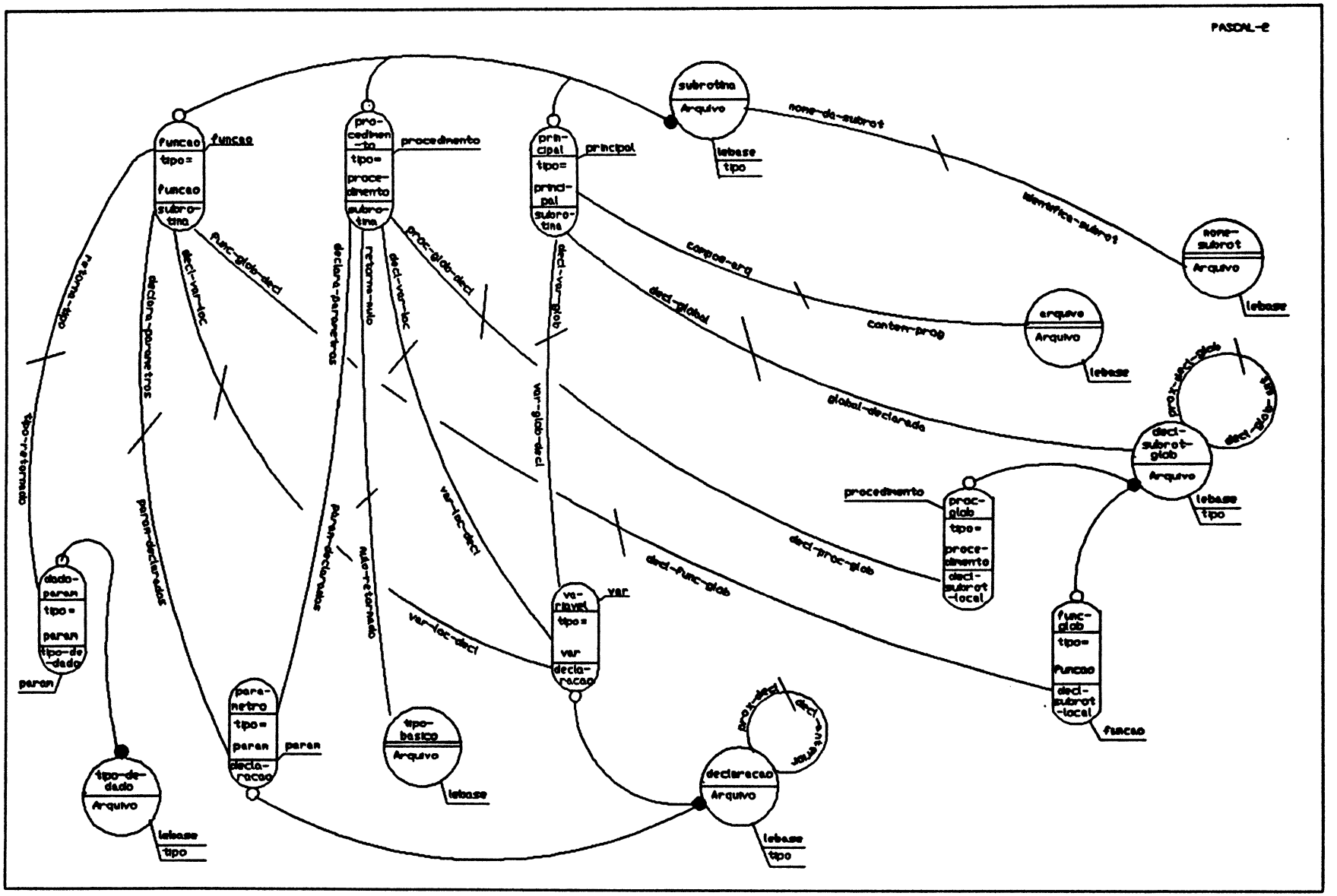
Definição da Linguagem Abstrata

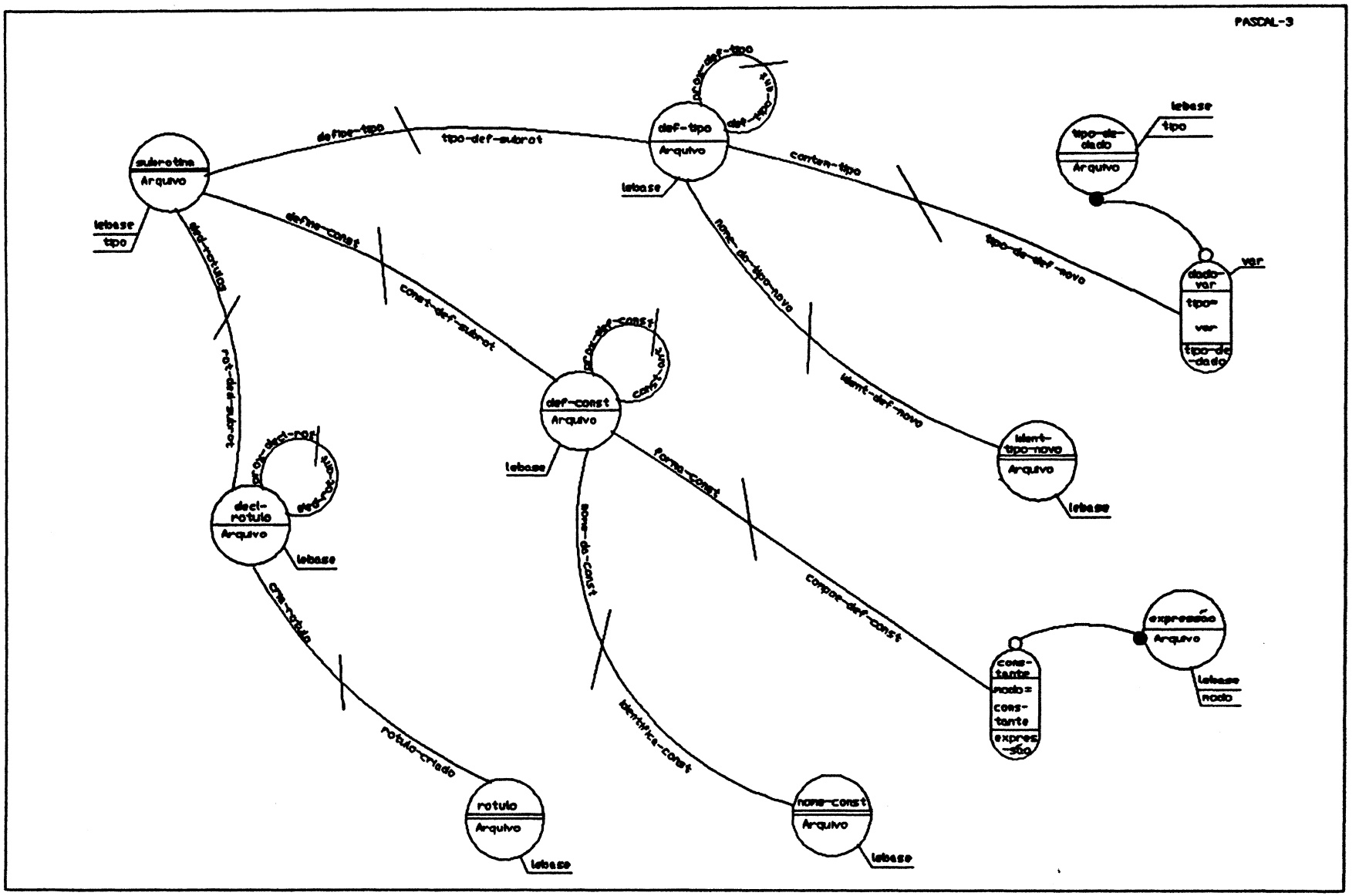
Considerações:

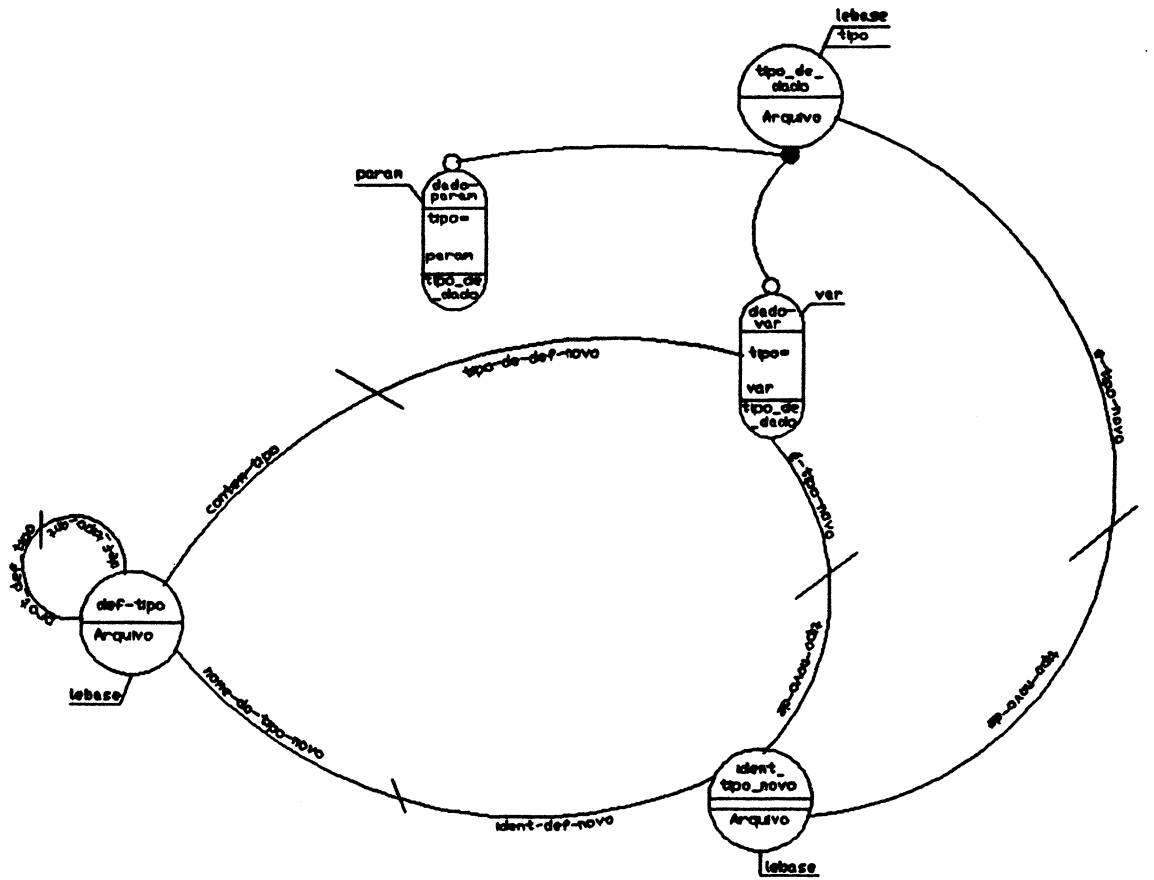
*É mostrada a concepção lógica da DLA utilizando-se o Diagrama de Representação de Objetos (DRO). São descritas para cada linguagem (**Pascal, C e FORTRAN**) apenas o que lhe é relevante sobre a DLA. Portanto, da figura Pascal-1 até Pascal-13 mostra-se a parte da DLA tratata pelo Editor quando se está sob o contexto da linguagem Pascal. O mesmo ocorre para as linguagens C e FORTRAN.*

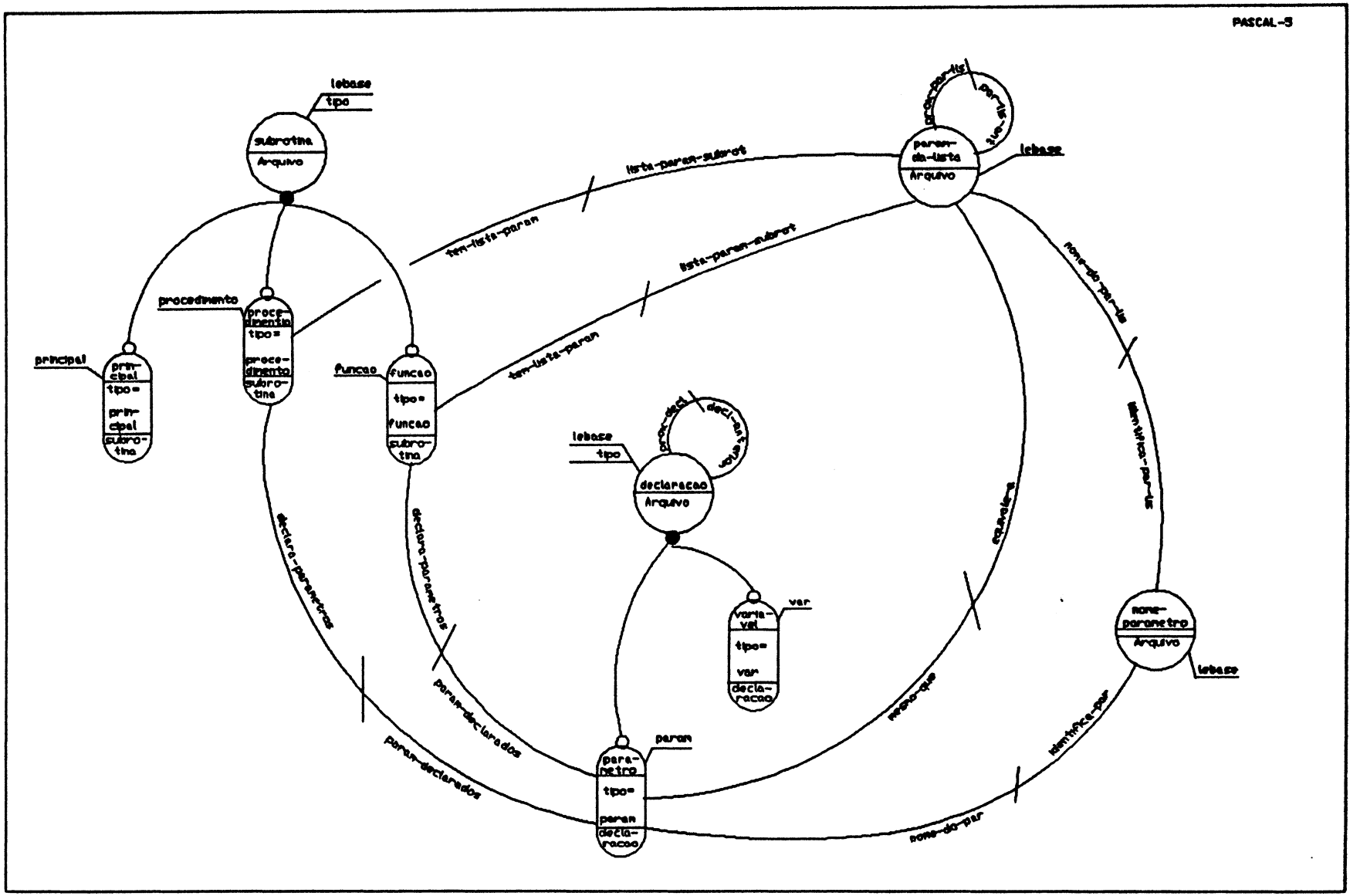
DLA
Pascal

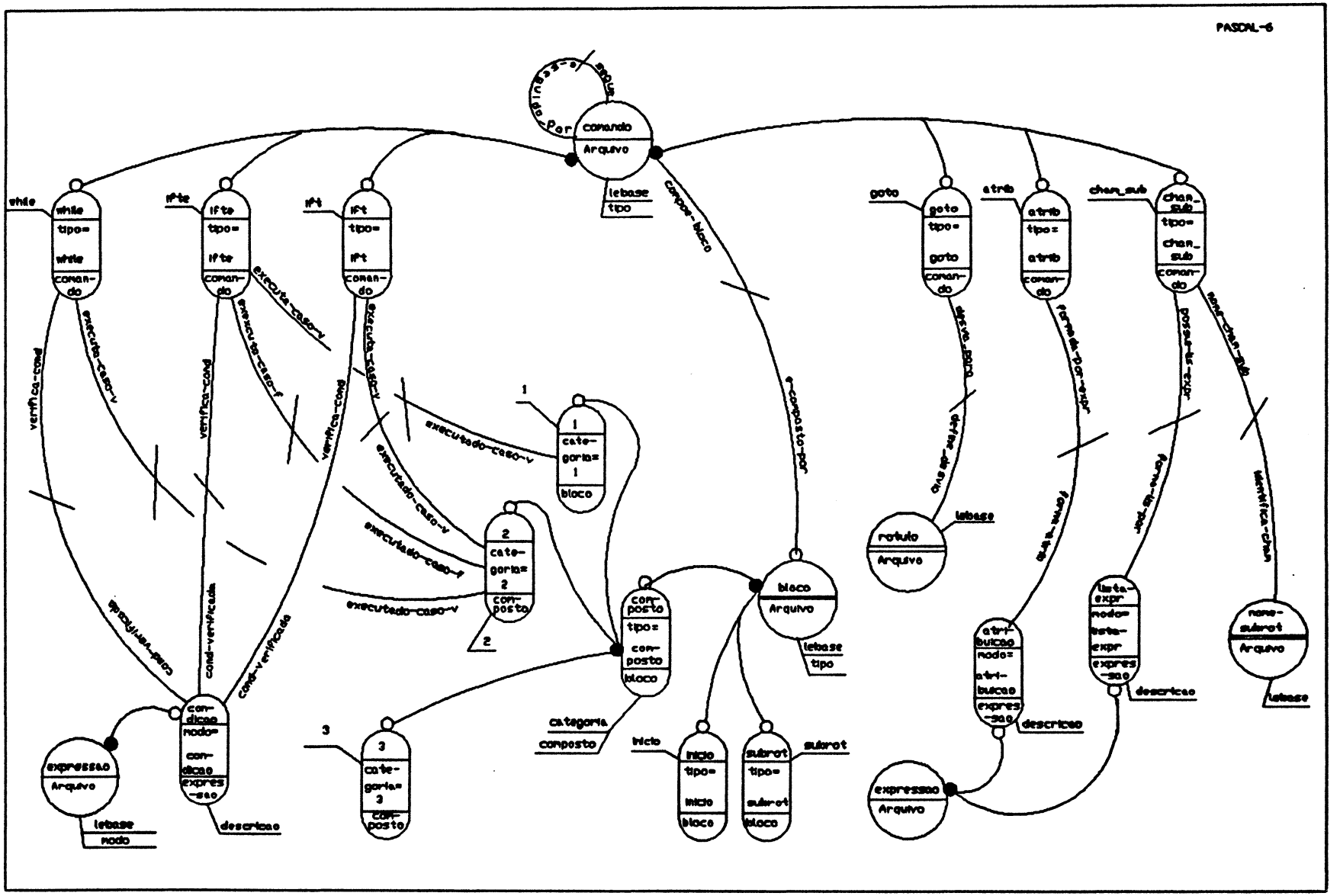

PASCAL-I
prox-decl-loc
decl-loc-rem
decl-subrot-loc
Arquivo
lebase
tipo
procedimento
proc-loc
tipo=
procedimento
decl-subrot-local
funcao
func-loc
tipo=
funcao
decl-subrot-local
loc-declarada
loc-declarada
decl-local
decl-local
decl-func-loc
func-loc-decl
decl-proc-loc
proc-loc-decl
func-contem-blc
blc-subrot-func
blc-subrot-proc
proc-contem-blc
retorna-valor
valor-retornado
sub-rot
tipo=
subrot
bloco
subrot
tipo
basico
Arquivo
bloco
Arquivo
lebase
tipo
inicio
tipo=
inicio
bloco
inicio
com-posto
tipo=
com-posto
bloco
categoria
composto
arquivo
global
lebase
blc-subrot-proc
contem-blc-mc
contem-prog
compos-prog
pro-cedimen-to
tipo=
proce-dimento
subro-tina
procedimento
funcao
tipo=
funcao
subro-tina
funcao
principal
tipo=
princi-pal
subro-tina
principal
subrotina
Arquivo
lebase
tipo
nome-de-subrot
identifica-subrot
none-subrot
Arquivo
lebase

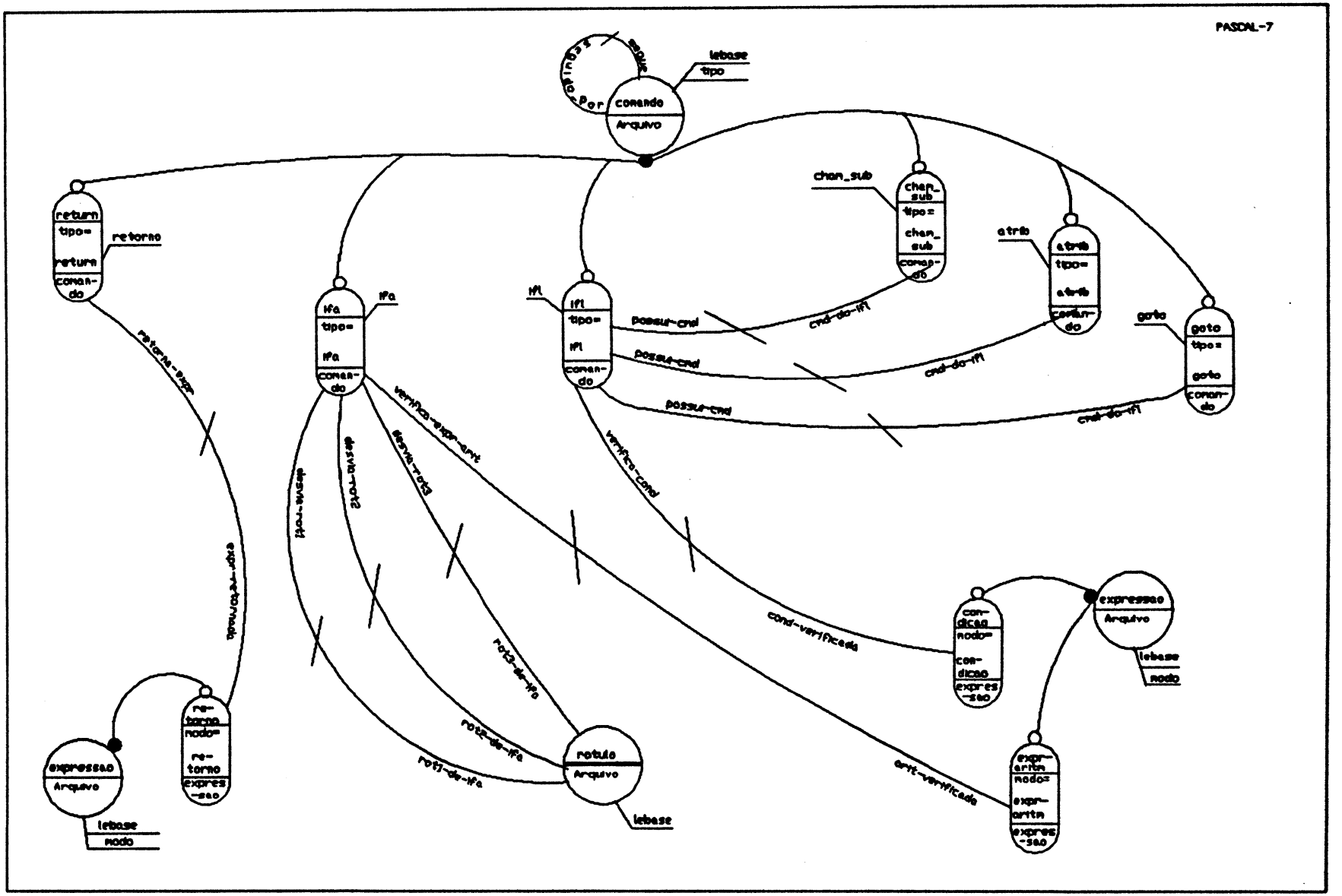

PASCAL-E
funcao
tipo=
funcao
subro-tina
funcao
procedimen-to
tipo=
proce-dimento
subro-tina
procedimento
principal
tipo=
princi-pal
subro-tina
principal
subrotina
Arquivo
lebase
tipo
none-subrot
Arquivo
lebase
arquivo
Arquivo
lebase
proc-glob
tipo=
proce-dimento
decl-subrot-local
procedimento
decl-subrot-glob
Arquivo
lebase
tipo
func-glob
tipo=
funcao
decl-subrot-local
funcao
va-riavel
tipo=
var
decla-racao
var
dado-paran
tipo=
paran
tipo-de-dado
paran
para-metro
tipo=
paran
decla-racao
paran
tipo-de-dado
Arquivo
lebase
tipo
tipo-basico
Arquivo
lebase
declaracao
Arquivo
lebase
tipo
proc-decl
decl-variavel
identifica-subrot
none-de-subrot
compoe-arq
contem-prog
decl-global
global-declarada
proc-decl-glob
decl-glob
decl-proc-glob
decl-func-glob
var-glob-decl
var-loc-decl
declara-parametros
retorna-valo
decl-var-loc
proc-glob-decl
func-glob-decl
declara-variaveis
parametros-declarados
valor-retornado
parametros-declarados

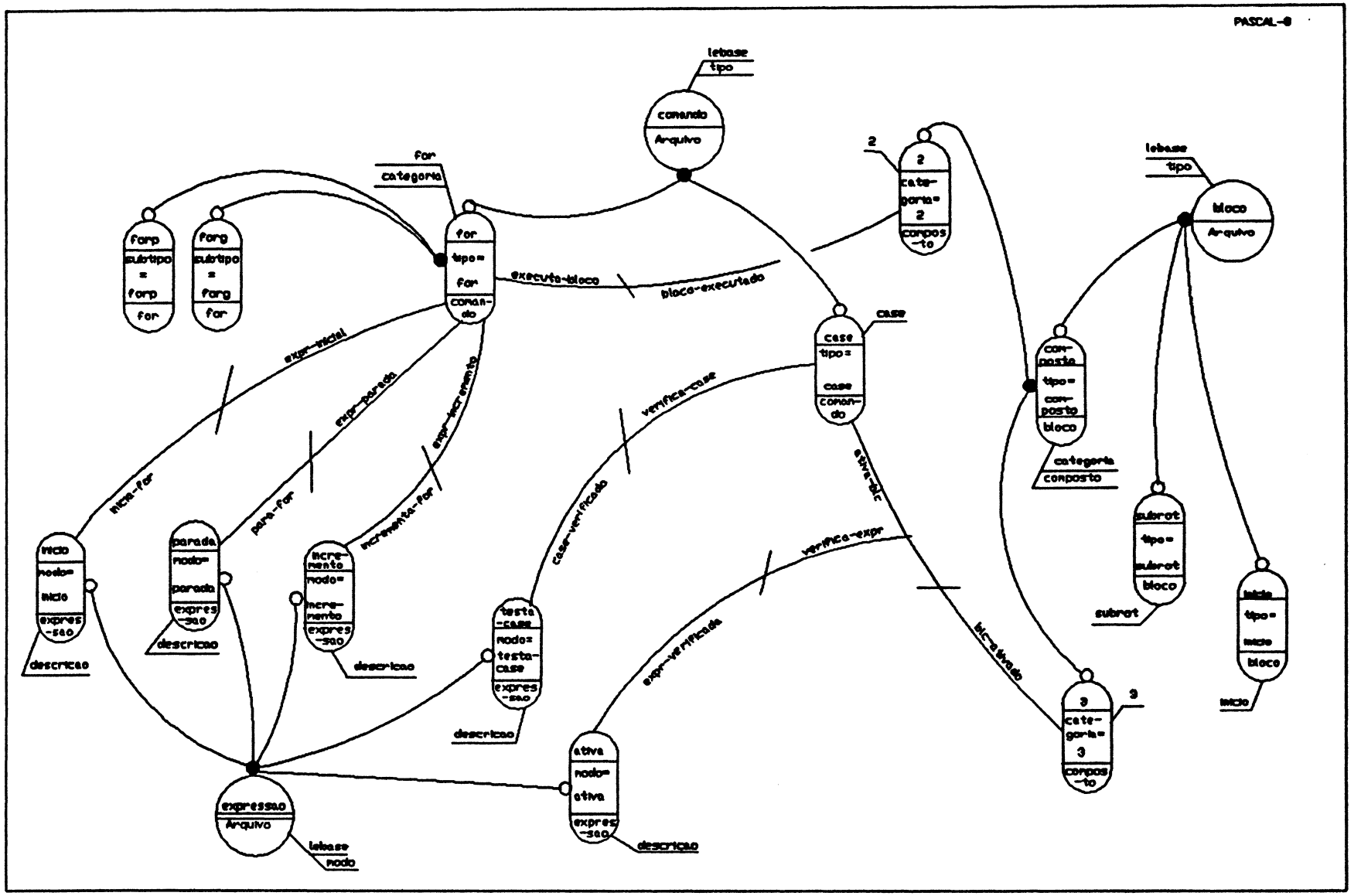

subrotina
Arquivo
lebase
tipo
defde-tipo
tipo-def-subrot
def-tipo
Arquivo
lebase
prox-def-tipo
def-tipo
corfen-tipo
tipo-de-dado
Arquivo
lebase
tipo
tipo-de-def-novo
dado-var
tipo=
var
tipo-de-dado
var
nome-do-tipo-novo
ident-def-novo
ident-tipo-novo
Arquivo
lebase
define-const
const-def-subrot
def-const
Arquivo
lebase
prox-def-const
const-def
define-const
compos-def-const
nome-do-const
identifica-const
nome-const
Arquivo
lebase
decl-rotulo
prox-decl-rot
decl-rot
decl-rotulo
Arquivo
lebase
cria-rotulo
rotulo-criado
rotulo
Arquivo
lebase
constante
modo=
constante
expressão
expressão
Arquivo
lebase
modo

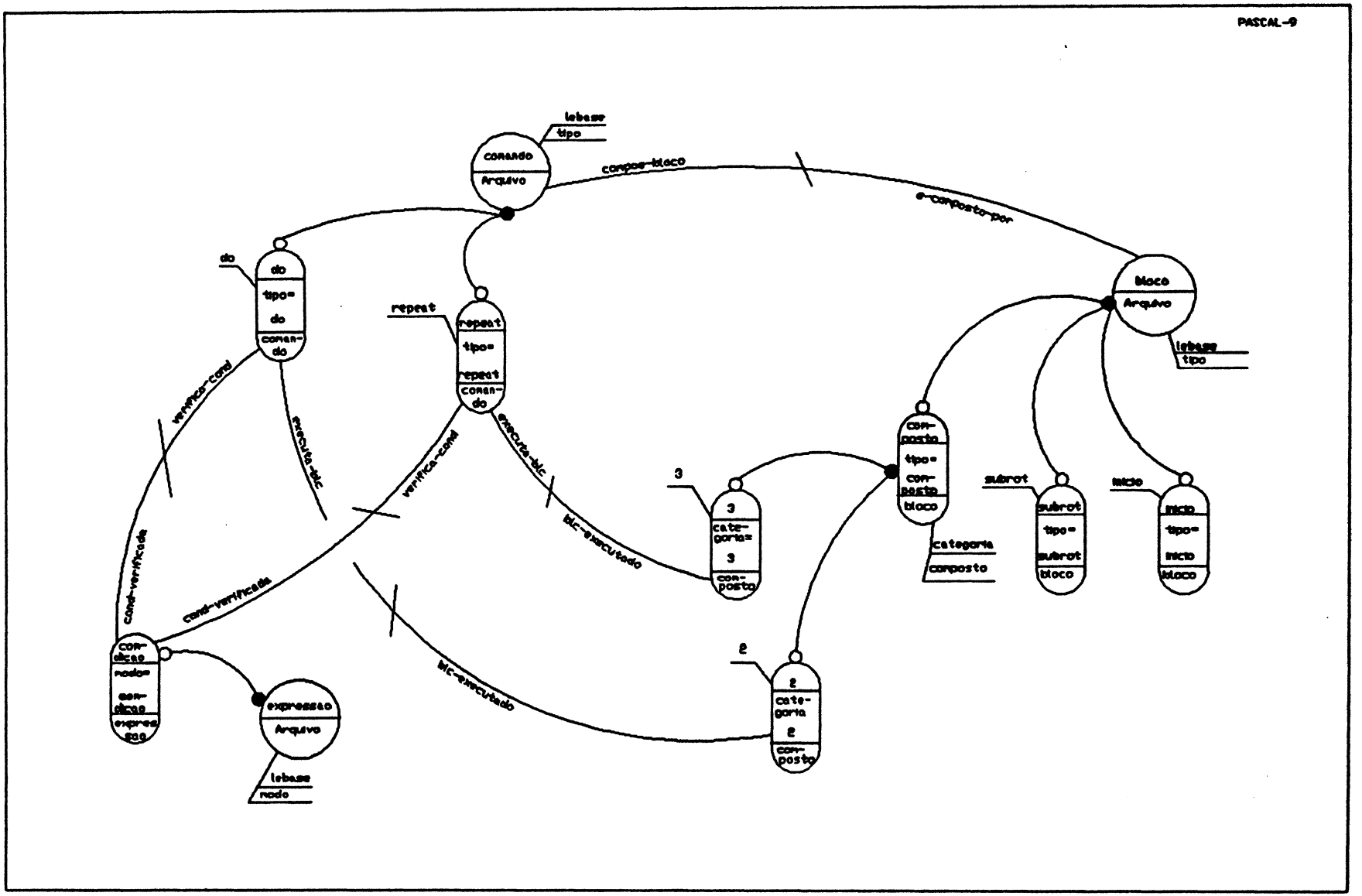

tebase
tipo
tipo_de_dado
Arquivo
paran
dado-paran
tipo=
paran
tipo_de_dado
var
dado-var
tipo=
var
tipo_de_dado
tipo-de-def-novo
contem-tipo
e-tipo-novo
e-tipo-novo
def-tipo
Arquivo
tebase
nome-do-tipo-novo
ident-def-novo
ident_tipo_novo
Arquivo
tebase

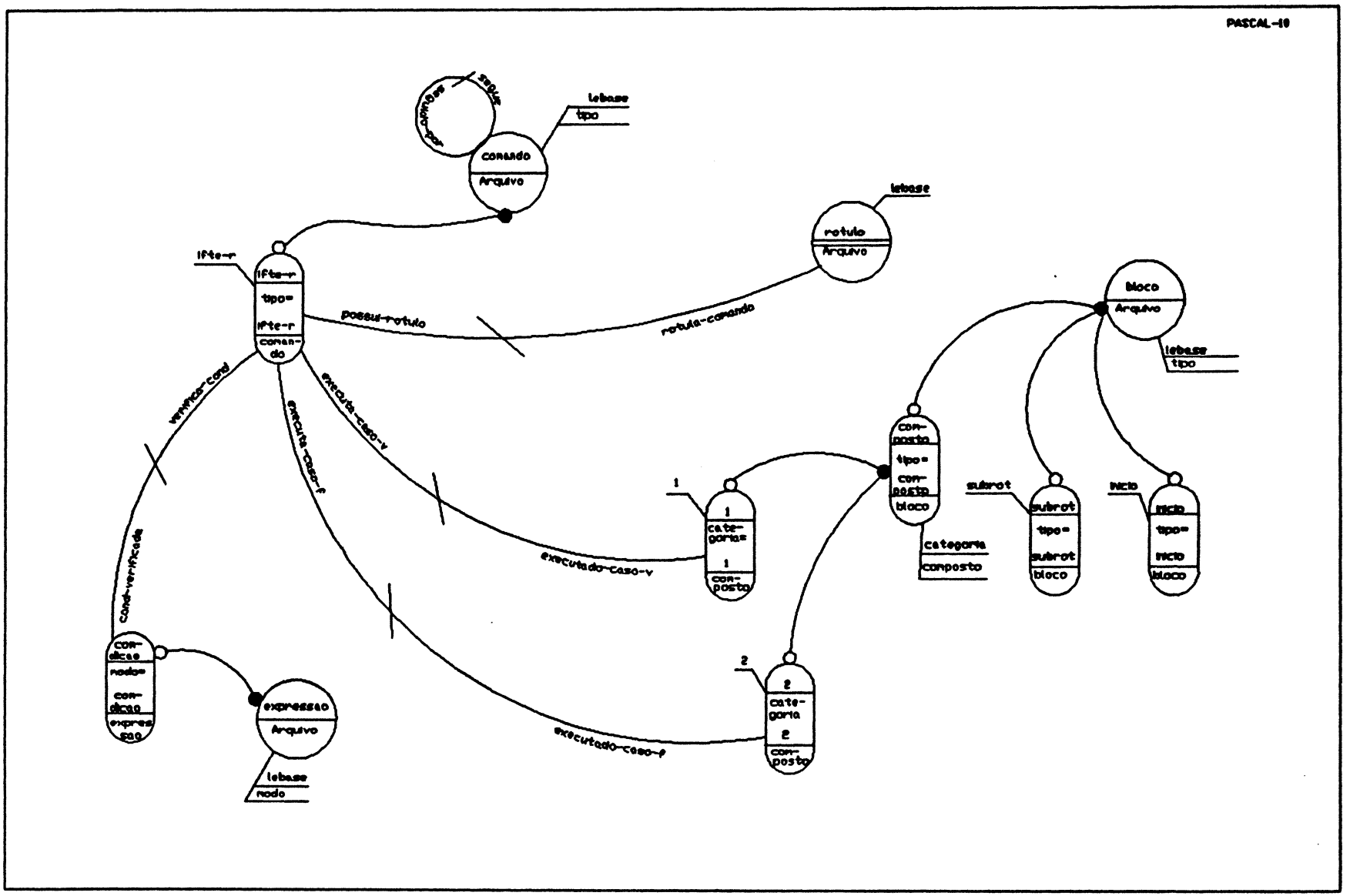

lebase
tipo
subrotina
Arquivo
lista-paran-subrot
tem-lista-paran
paran-da-lista
lebase
Arquivo
lista-paran-subrot
tem-lista-paran
nome-da-paran
identifica-paran
procedimento
principal
funcao
declaracao
Arquivo
declara-parametros
declara-parametros
var
tipo
var
declaracao
equivale
nome-parametro
Arquivo
lebase
paran-declarados
paran-declarados
paran
tipo
paran
declaracao
nome-do-par
identifica-par

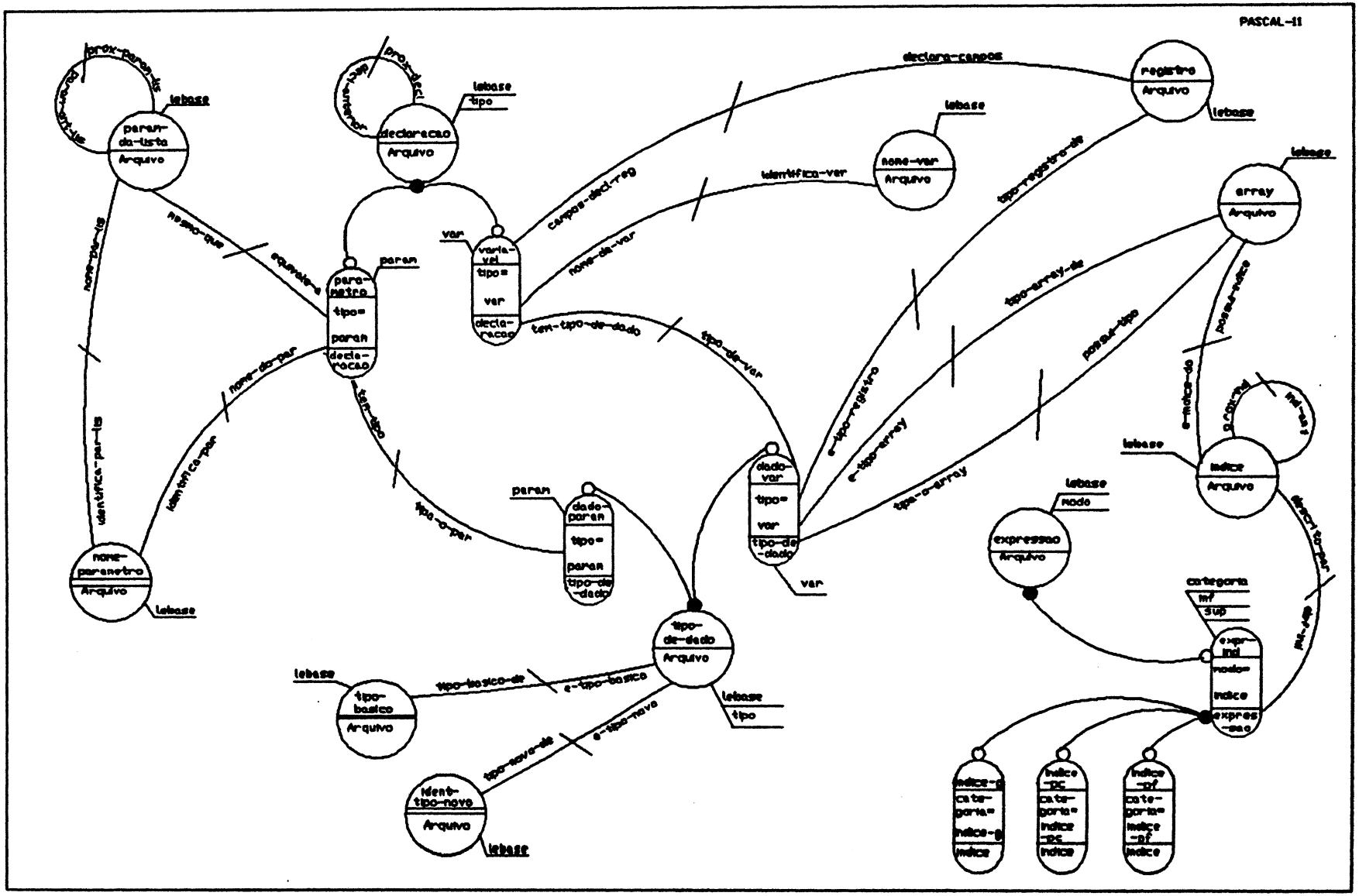

comando
Arquivo
lebase
tipo
proximo-comando
while
tipo=
while
comando
ifte
tipo=
ifte
comando
if
tipo=
if
comando
goto
tipo=
goto
comando
atrib
tipo=
atrib
comando
chan_sub
tipo=
chan_sub
comando
verifica-cond
executa-caso-v
executa-caso-f
executado-caso-v
executado-caso-f
cond-verificada
1
categoria=
1
bloco
2
categoria=
2
composto
3
categoria=
3
composto
composto
tipo=
composto
bloco
composto-por
bloco
Arquivo
lebase
tipo
rotulo
Arquivo
lebase
desvio-para
atribuicao
nodo=
atribuicao
expressao
descricao
atrib-por-expr
lista-expr
nodo=
lista-expr
expressao
descricao
poe-lista-expr
nome-subrot
Arquivo
lebase
nome-chan-sub
identifica-chan
expressao
Arquivo
lebase
nodo
condicao
nodo=
condicao
expressao
descricao
categoria
composto
inicio
tipo=
inicio
bloco
subrot
tipo=
subrot
bloco
subrot

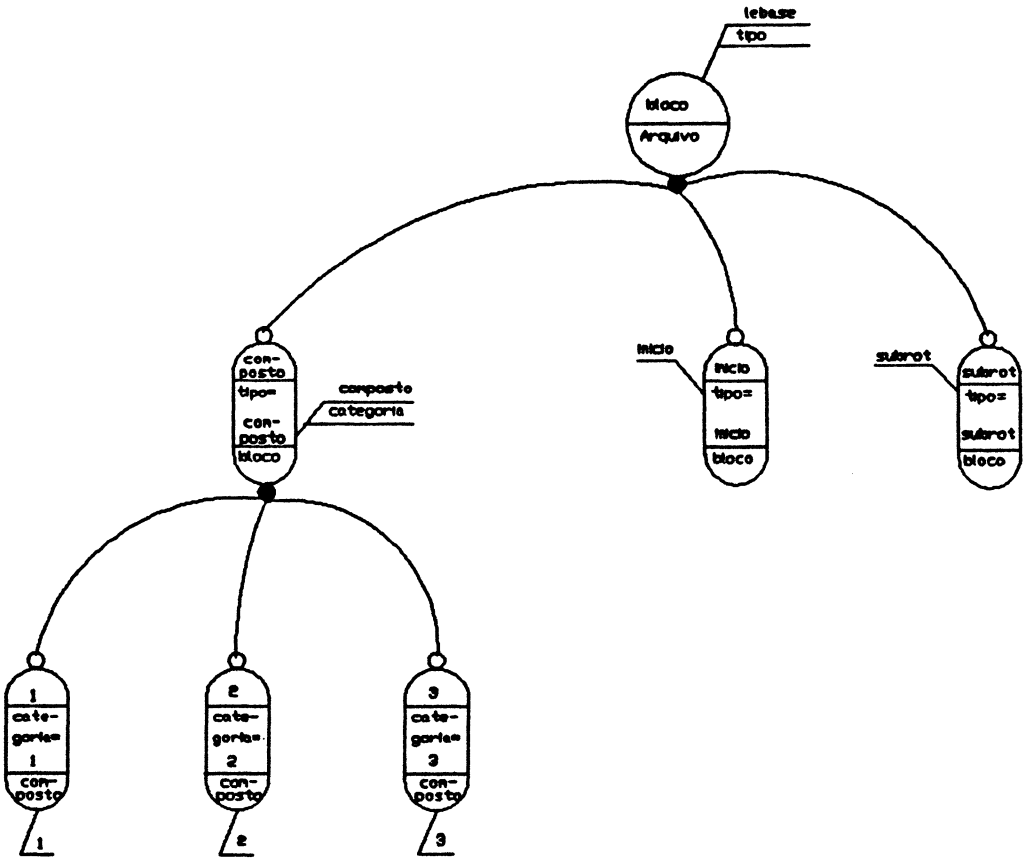

comando
Arquivo
lebase
tipo
sucessor
precedido_por
return
tipo=
return
comando
retorno
retorna-expr
retorna-nada
expressao
Arquivo
lebase
nodo
re-torno
nodo=
re-torno
expres-sao
ifa
tipo=
ifa
comando
ifa
verifica-expr-ari
desvia-fora
desvia-dentro
desvia-nada
nota-de-ifa
nota-de-ifa
nota-de-ifa
rotulo
Arquivo
lebase
ifl
tipo=
ifl
comando
ifl
possui-cmd
possui-cmd
possui-cmd
cmd-do-ifl
cmd-do-ifl
cmd-do-ifl
verifica-cond
cond-verificada
chan_sub
chan_sub
tipo=
chan_sub
comando
atrib
atrib
tipo=
atrib
comando
goto
goto
tipo=
goto
comando
condicao
nodo=
condicao
expres-sao
expressao
Arquivo
lebase
nodo
arit-verificada
expr-aritm
nodo=
expr-aritm
expres-sao

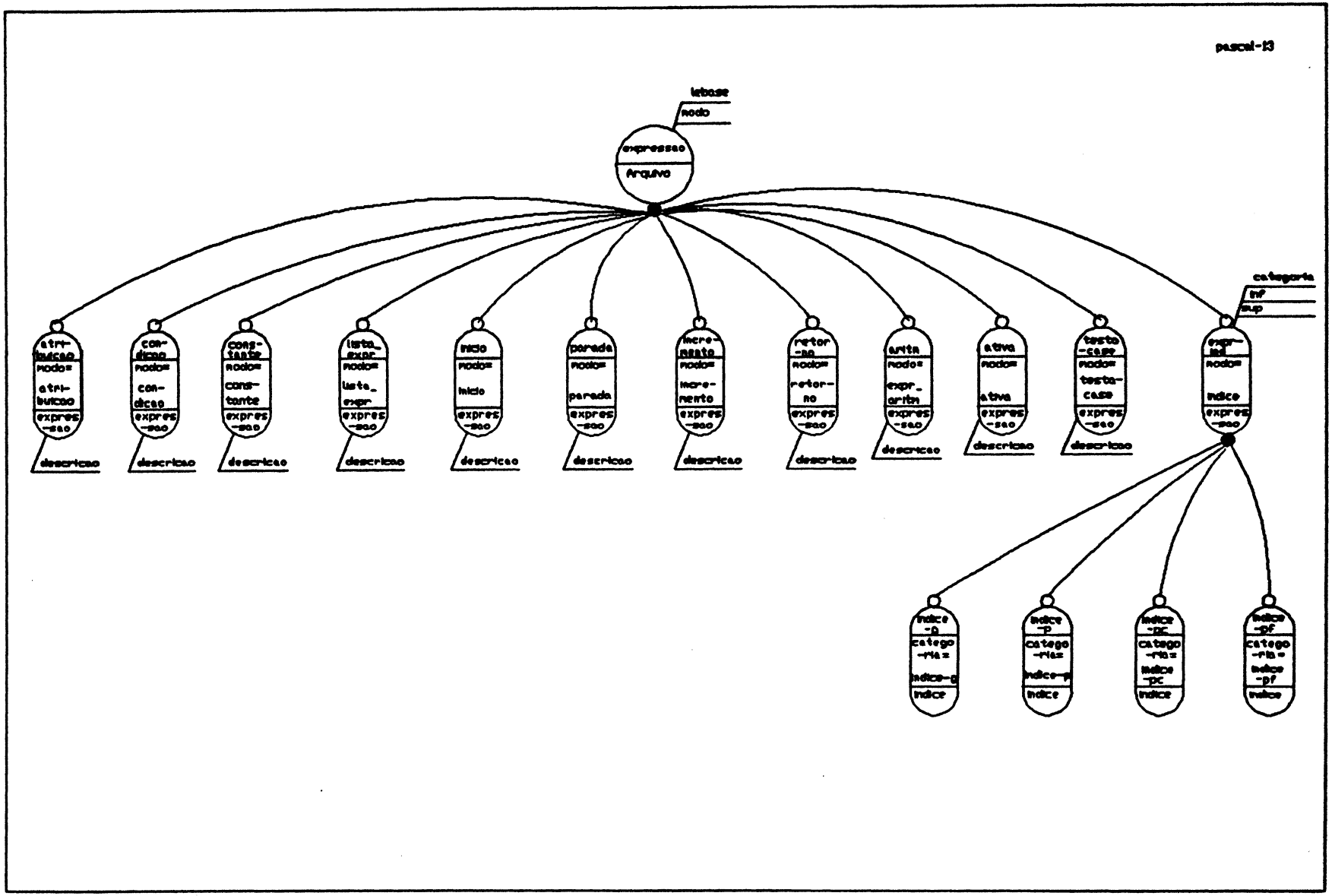

lebase
tipo
comando
Arquivo
for
categoria
forp
subtipo = forp
for
forg
subtipo = forg
for
for
tipo= for
comando
executa-bloco
bloco-executado
expr-inicial
expr-parada
expr-incremento
inicia-for
para-for
incrementa-for
início
nodo= início
expressão
descricao
parada
nodo= parada
expressão
descricao
incremento
nodo= incremento
expressão
descricao
case
case
tipo= case
comando
verifica-case
case-verificado
testa-case
nodo= testa-case
expressão
descricao
ativa
nodo= ativa
expressão
descricao
expr-verificada
verifica-expr
ativável
ativado
2
2
categoria= 2
composto
3
3
categoria= 3
composto
composto
tipo= composto
bloco
categoria
composto
lebase
tipo
bloco
Arquivo
subrot
tipo= subrot
bloco
subrot
início
tipo= início
bloco
início
expressão
Arquivo
lebase
nodo


lebase
tipo
comando
Arquivo
compos-bloco
e-composto-por
bloco
Arquivo
lebase
tipo
do
do
tipo=
do
coman-
do
repeat
repeat
tipo=
repeat
coman-
do
verifica-cond
executa-blc
cond-verificada
verifica-cond
executa-blc
blc-executado
3
3
cate-
goria=
3
com-
posto
com-
posto
tipo=
com-
posto
bloco
categoria
composto
subrot
subrot
tipo=
subrot
bloco
inicio
inicio
tipo=
inicio
bloco
cond-verificada
blc-executado
e
e
cate-
goria
e
com-
posto
con-
dicao
nodo=
con-
dicao
expres-
sao
expressao
Arquivo
lebase
nodo

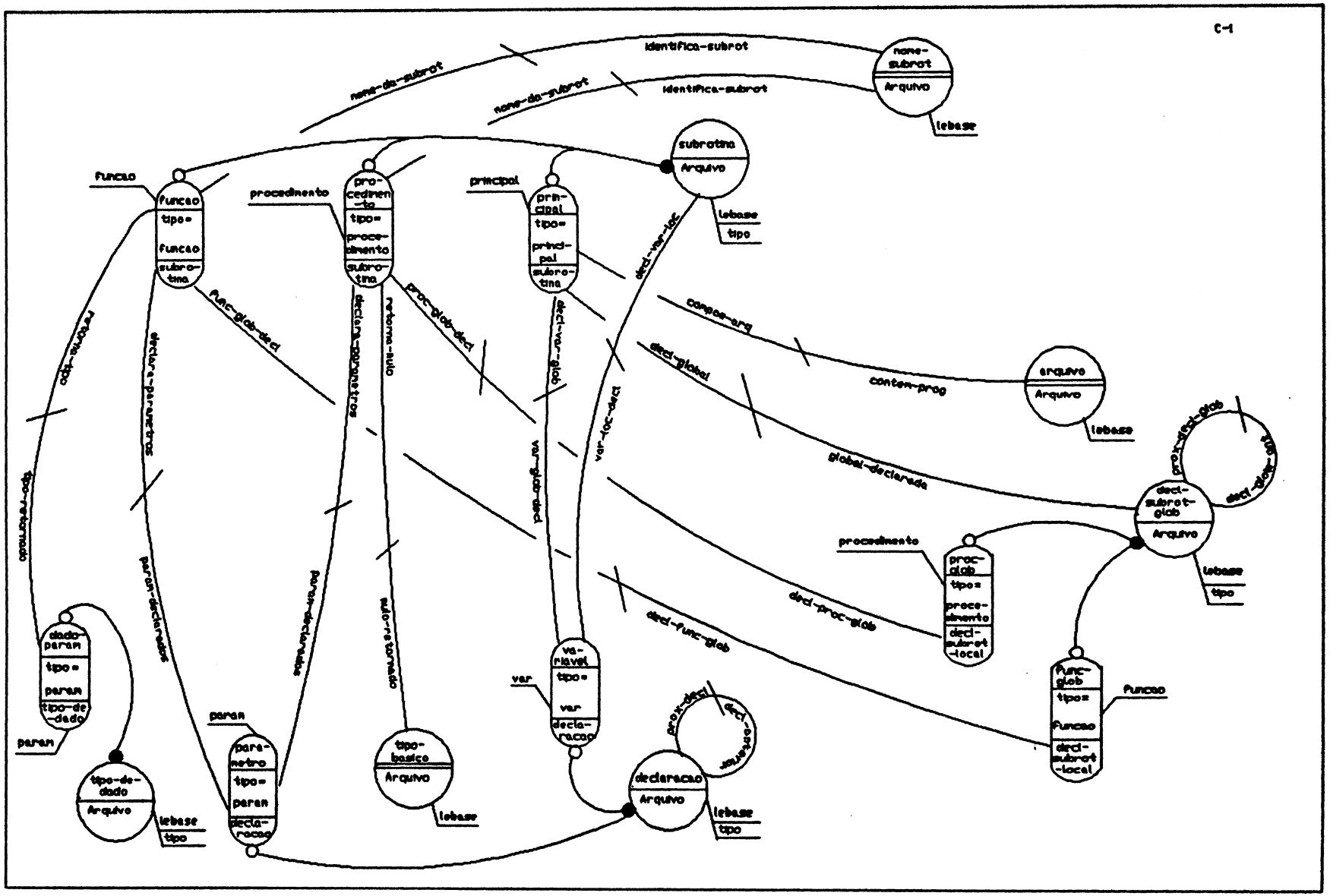

seguinte
status
lebase
tipo
comando
Arquivo
lfte-r
lfte-r
tipo=
lfte-r
coman-
do
possui-rotulo
rotulo-comando
rotulo
Arquivo
lebase
verifica-cond
executa-caso-v
executa-caso-f
cond-verificada
executado-caso-v
executado-caso-f
con-
dicao
nodo=
con-
dicao
expres
sao
expressao
Arquivo
lebase
nodo
1
1
cate-
goria=
1
com-
posto
2
2
cate-
goria
2
com-
posto
com-
posto
tipo=
com-
posto
bloco
categoria
composto
Bloco
Arquivo
lebase
tipo
subrot
subrot
tipo=
subrot
bloco
inicio
inicio
tipo=
inicio
bloco

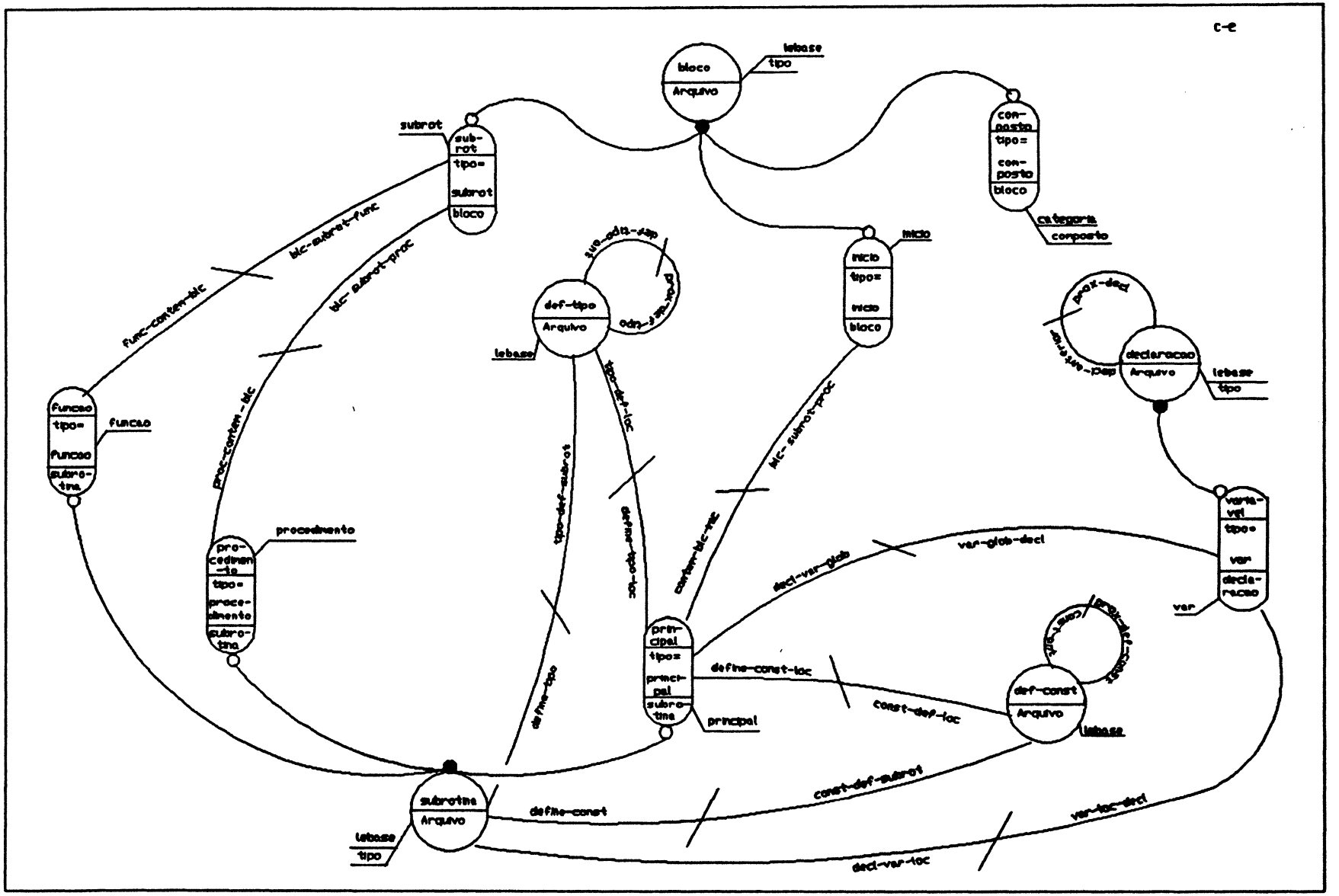

PASCAL-II
prox-parametro
lebase
parametro
Arquivo
prox-declaracao
declaracao
lebase
tipo
declara-campos
registro
Arquivo
lebase
nome-var
Arquivo
lebase
identifica-var
campos-decl-reg
nome-de-var
tipo-registro-de
array
Arquivo
lebase
nome-parametro
mesmo-que
equivale-a
parametro
tipo
var
declaracao
var
variavel
tipo
var
declaracao
ten-tipo-de-dado
tipo-de-var
tipo-array-de
posse-indice
e-indice-de
proximo
indice
Arquivo
lebase
e-tipo-registro
e-tipo-array
tipo-e-array
posse-tipo
identifica-par-ls
nome-do-par
identifica-par
ter-algo
tipo-e-par
paran
dado-paran
tipo
paran
tipo-de-dado
dado-var
tipo
var
tipo-de-dado
var
expressao
Arquivo
lebase
nodo
categoria
inf
sup
expr-ini
nodo
indice
expres-sao
tipo-de-dado
Arquivo
lebase
tipo
tipo-basico
Arquivo
lebase
tipo-basico-de
e-tipo-basico
ident-tipo-nova
Arquivo
lebase
tipo-nova-de
e-tipo-nova
indice-g
cate-goria
indice-g
indice
indice-pc
cate-goria-indice-pc
indice
indice-pf
cate-goria-indice-pf
indice

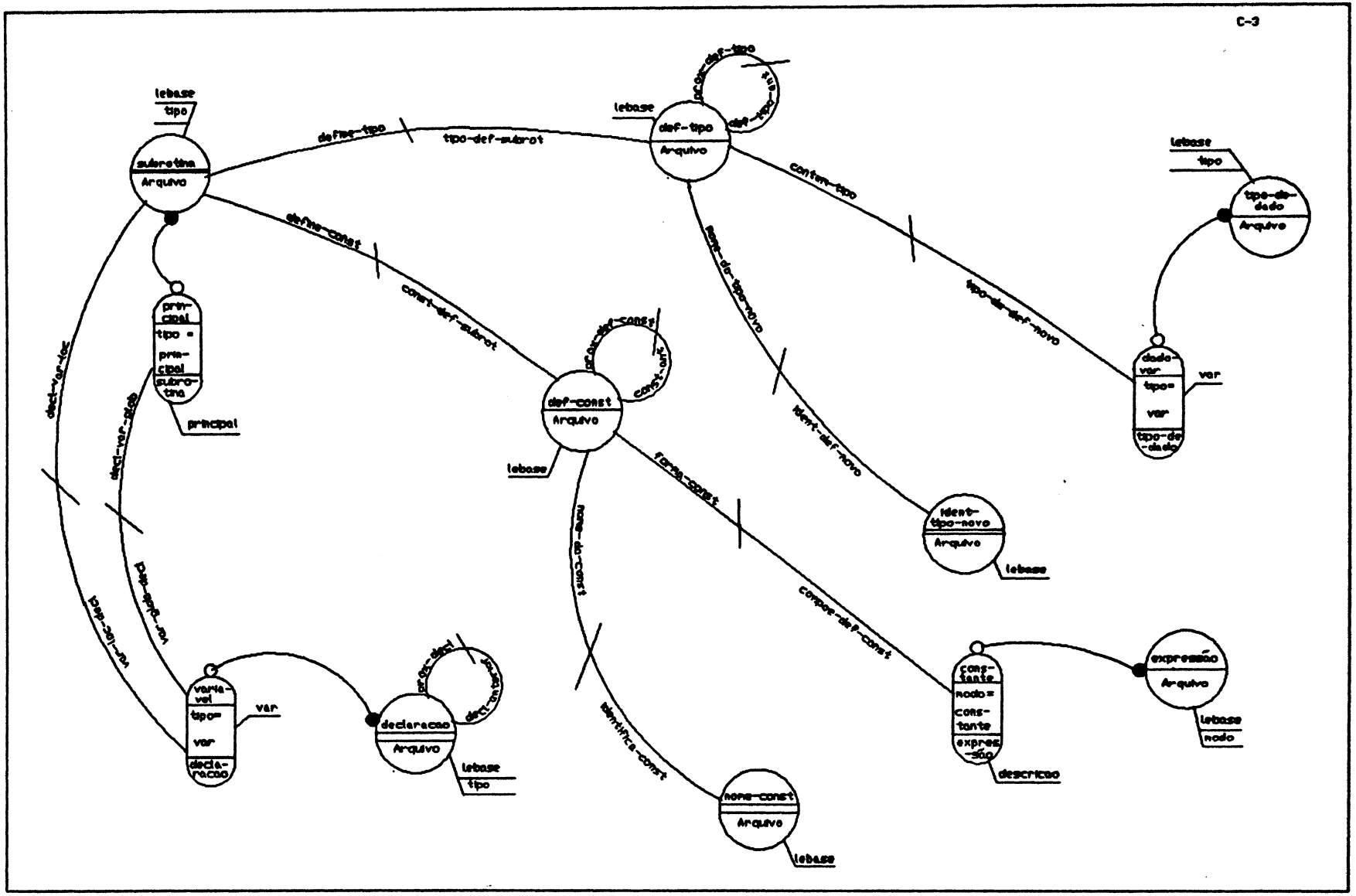

lebase
tipo
bloco
Arquivo
con-
posto
tipo=
con-
posto
bloco
composto
categoria
inicio
inicio
tipo=
inicio
bloco
subrot
subrot
tipo=
subrot
bloco
1
cate-
goria=
1
con-
posto
2
cate-
goria=
2
con-
posto
3
cate-
goria=
3
con-
posto
1
2
3

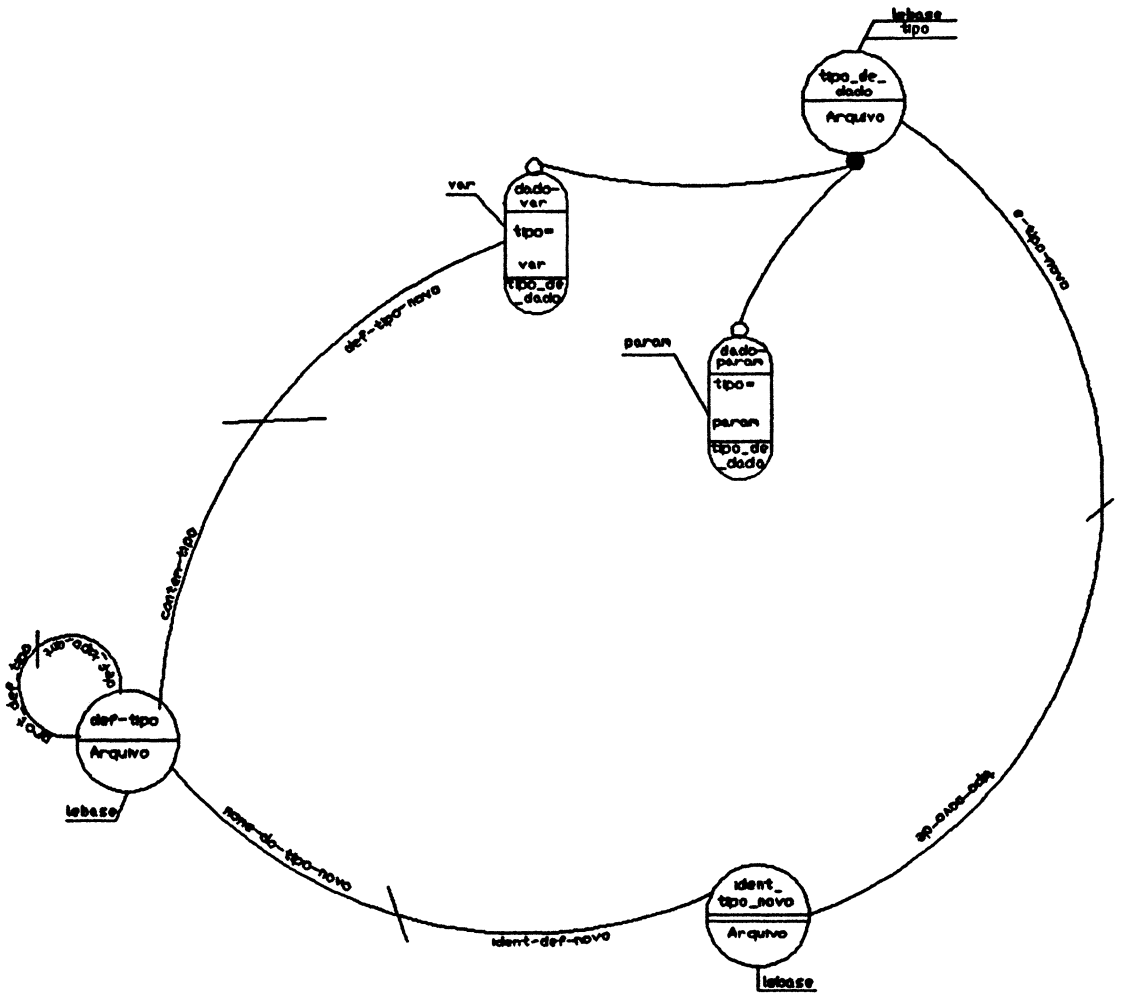

pascal-13
tabase
nodo
expressao
Arquivo
atri-buicao
nodo=
atri-buicao
expres-sao
descricao
con-dicao
nodo=
con-dicao
expres-sao
descricao
cons-tante
nodo=
cons-tante
expres-sao
descricao
lista_expr
nodo=
lista_expr
expres-sao
descricao
inicio
nodo=
inicio
expres-sao
descricao
parada
nodo=
parada
expres-sao
descricao
incre-mento
nodo=
incre-mento
expres-sao
descricao
retor-no
nodo=
retor-no
expres-sao
descricao
aritm
nodo=
expr_aritm
expres-sao
descricao
ativa
nodo=
ativa
expres-sao
descricao
testa-caso
nodo=
testa-caso
expres-sao
descricao
expr-ind
nodo=
indice
expres-sao
categoria
inf
sup
indice-g
categ-ria=
indice-g
indice
indice-p
categ-ria=
indice-p
indice
indice-pc
categ-ria=
indice-pc
indice
indice-pf
categ-ria=
indice-pf
indice

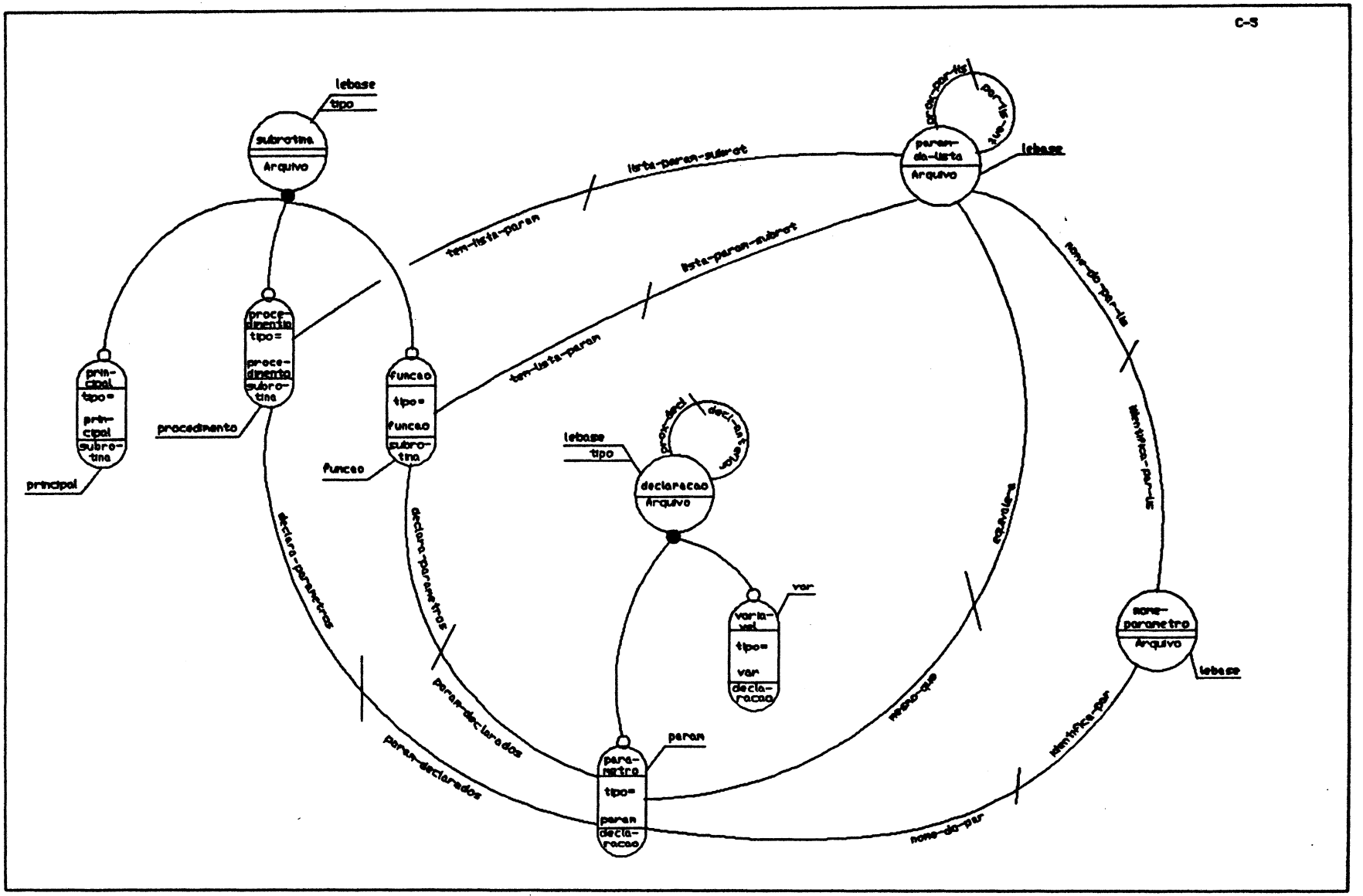
DLA
C

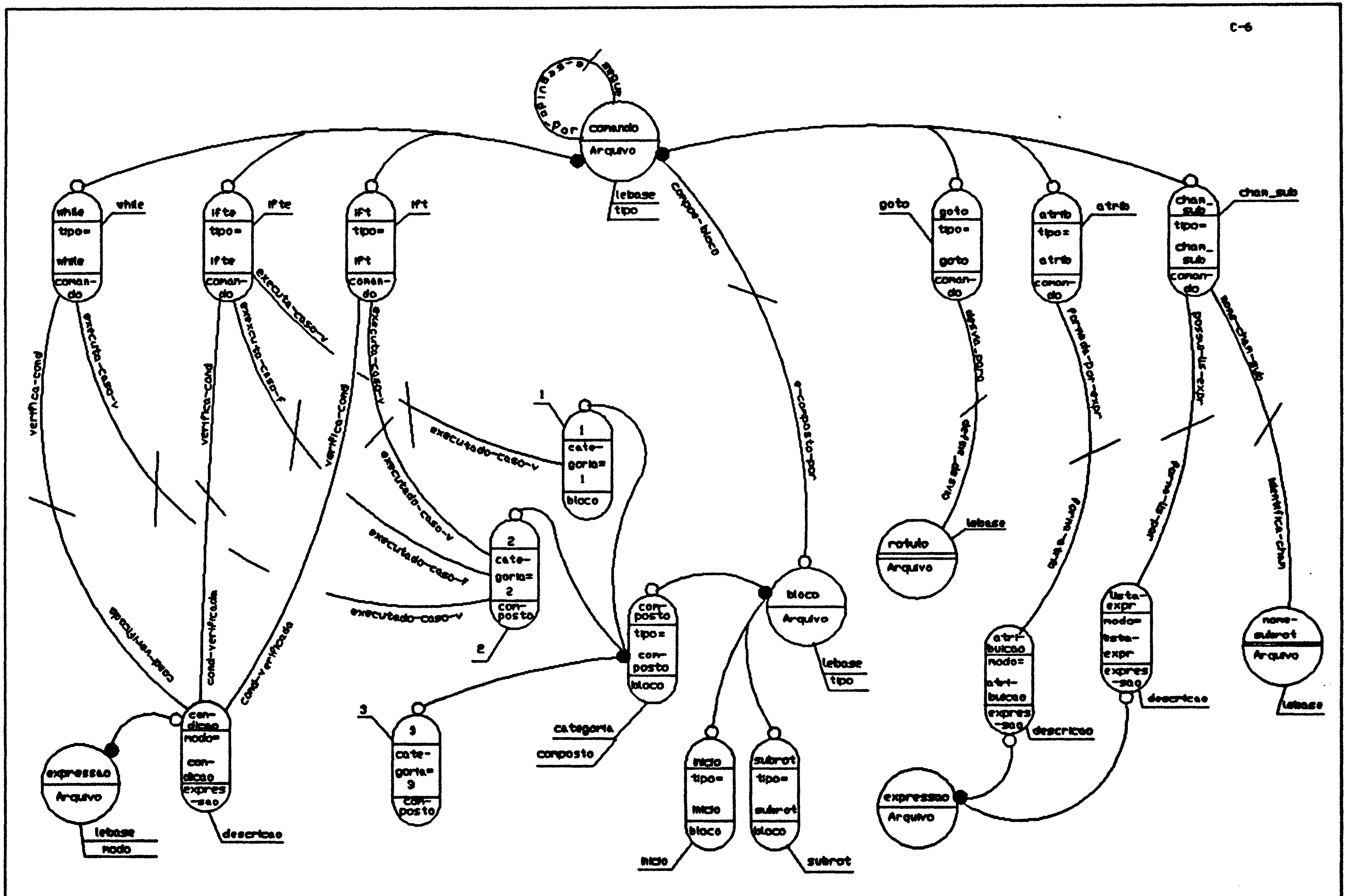

identifica-subrot
nome-da-subrot
nome-da-subrot
identifica-subrot
nome-subrot
Arquivo
lebase
subrotina
Arquivo
lebase
tipo
funcao
funcao
tipo=
funcao
subro-tina
procedimento
pro-cedimen-to
tipo=
proce-dimento
subro-tina
principal
prin-cipal
tipo=
princi-pal
subro-tina
decl-var-loc
var-loc-decl
campos-arg
contem-prog
arquivo
Arquivo
lebase
decl-global
global-declarada
decl-subrot-glob
Arquivo
lebase
tipo
procedimento
proc-glob
tipo=
proce-dimento
decl-subrot-local
decl-proc-glob
Func-glob
tipo=
funcao
decl-subrot-local
decl-func-glob
funcao
cab-subprog
tipo-retornado
declara-parametros
parametros
parametros-declarados
func-glob-decl
declara-parametros
retorna-valor
proc-glob-decl
decl-var-glob
var-glob-decl
valor-retornado
dado-param
tipo=
param
tipo-de-dado
tipo-de-dado
Arquivo
lebase
tipo
param
para-metro
tipo=
param
declaracao
tipo-basico
Arquivo
lebase
var
va-riavel
tipo=
var
declaracao
declaracao
Arquivo
lebase
tipo
prot-decl
decl-parametros

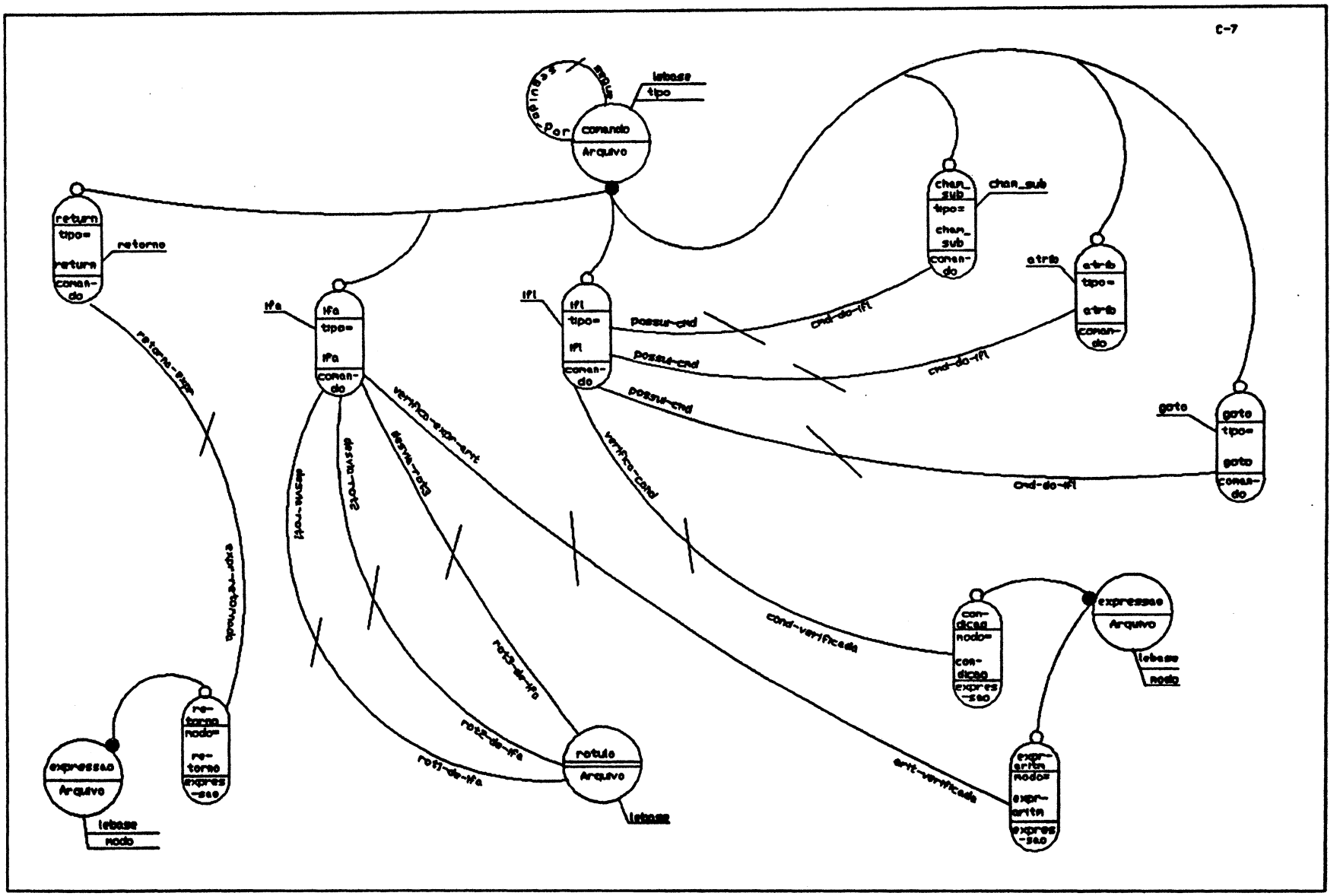

c-e
bloco
Arquivo
tipo
lebase
sub-rot
tipo=
subrot
bloco
subrot
con-posto
tipo=
con-posto
bloco
categoria
composto
inicio
inicio
tipo=
inicio
bloco
def-tipo
Arquivo
lebase
sub_dda_dep
pai_dda_dep
declaracao
Arquivo
lebase
tipo
prox_decl
decl_dep
funcao
tipo=
funcao
subro-tina
funcao
pro-cedimen-to
tipo=
proce-dimento
subro-tina
procedimento
princi-pal
tipo=
princi-pal
subro-tina
principal
varia-vel
tipo=
var
decla-racao
var
subrotina
Arquivo
lebase
tipo
def-const
Arquivo
lebase
const_dep
prox_const
func-contem-blc
blc-subrot-func
proc-contem-blc
blc-subrot-proc
tipo-def-subrot
tipo-def-loc
define-tipo
define-tipo-loc
contem-blc-inic
blc-subrot-proc
decl-var-glob
var-glob-decl
define-const-loc
const-def-loc
define-const
const-def-subrot
decl-var-loc
var-loc-decl

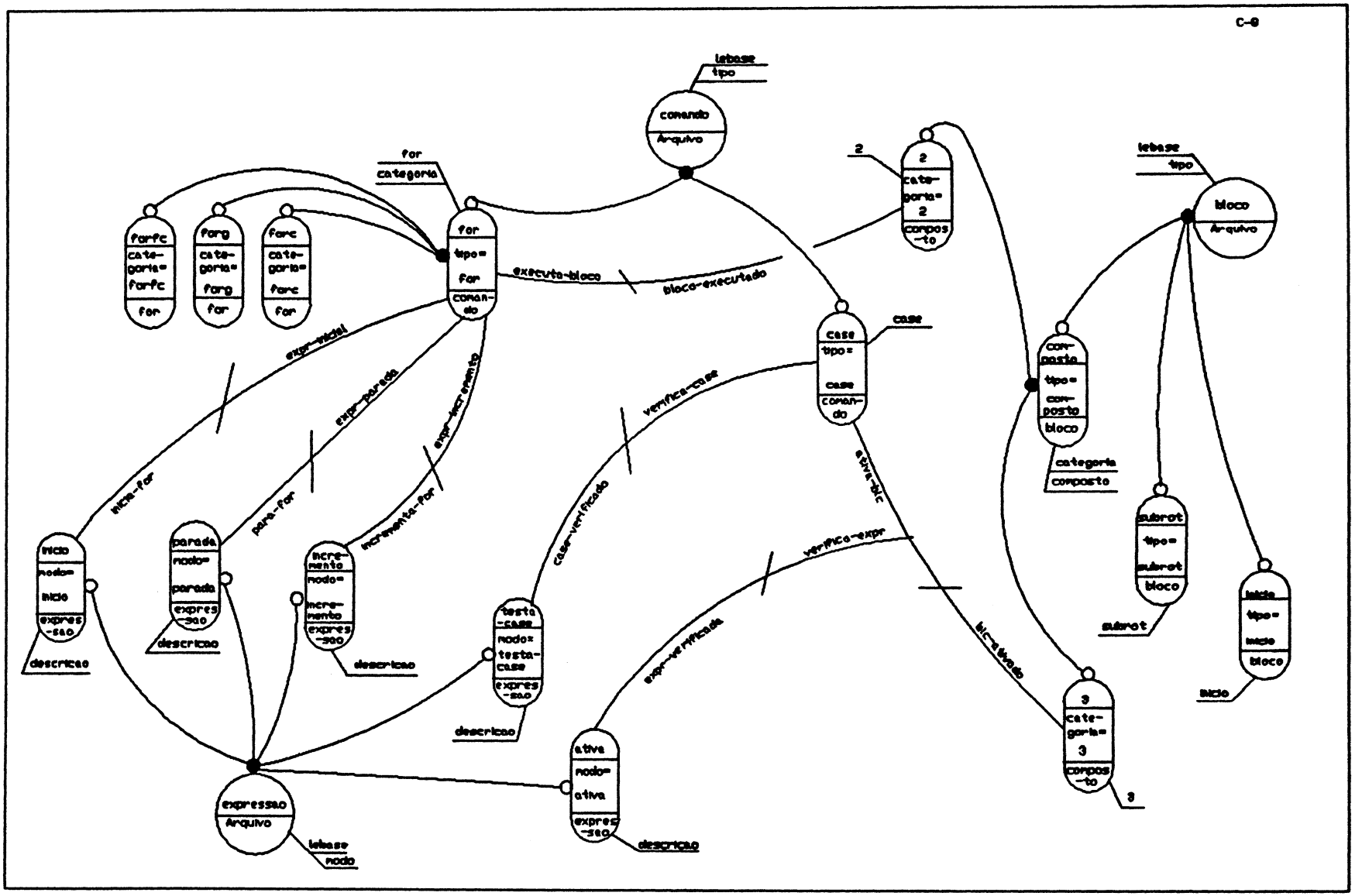

lebase
tipo
subrotina
Arquivo
define-tipo
tipo-def-subrot
def-tipo
Arquivo
def-def-tipo
def-tipo
lebase
contem-tipo
lebase
tipo
tipo-de-dado
Arquivo
define-const
const-def-subrot
def-const
Arquivo
def-def-const
const-var
lebase
const-de-tipo-novo
tipo-de-def-novo
ident-def-novo
dado-var
tipo=
var
tipo-de-dado
var
ident-tipo-novo
Arquivo
lebase
principal
pri-cipal
tipo =
pri-cipal
subro-tina
decl-var-glob
decl-var-loc
var-glob-decl
var-loc-decl
nome-de-const
forma-const
compoe-def-const
identifica-const
varia-vel
tipo=
var
decla-racao
var
declaracao
Arquivo
def-declaracao
declaracao
lebase
tipo
nome-const
Arquivo
lebase
cons-tante
nodo=
cons-tante
expres-são
descricao
expressão
Arquivo
lebase
nodo

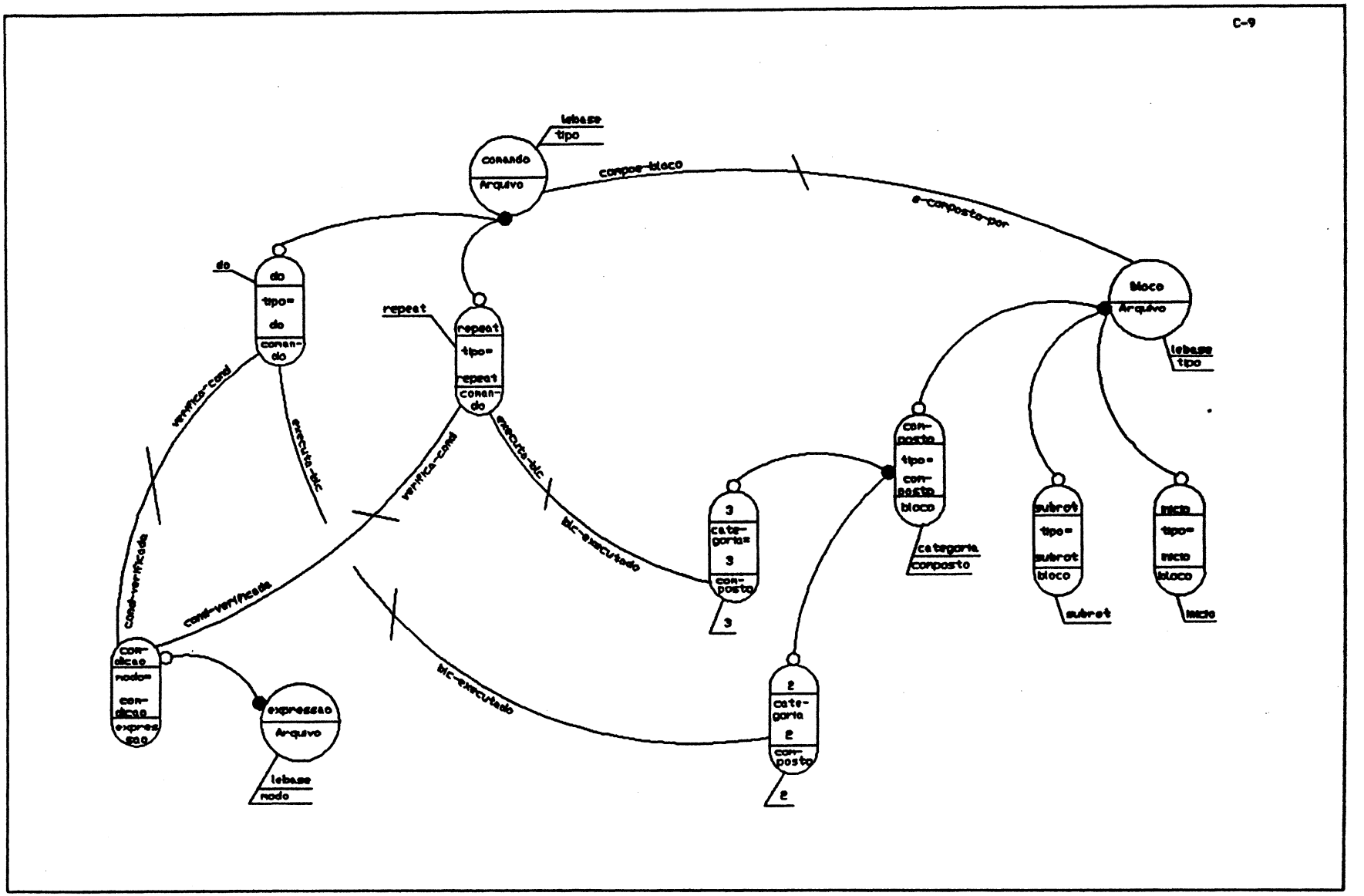

tipo_de_dado
Arquivo
tabase
tipo
dado-var
tipo=
var
tipo_de_dado
var
dado-paran
tipo=
paran
tipo_de_dado
paran
e-tipo-novo
def-tipo-novo
contem-tipo
def-tipo
Arquivo
tabase
nome-do-tipo-novo
ident-def-novo
ident_tipo_novo
Arquivo
tabase
tip-dado-adic

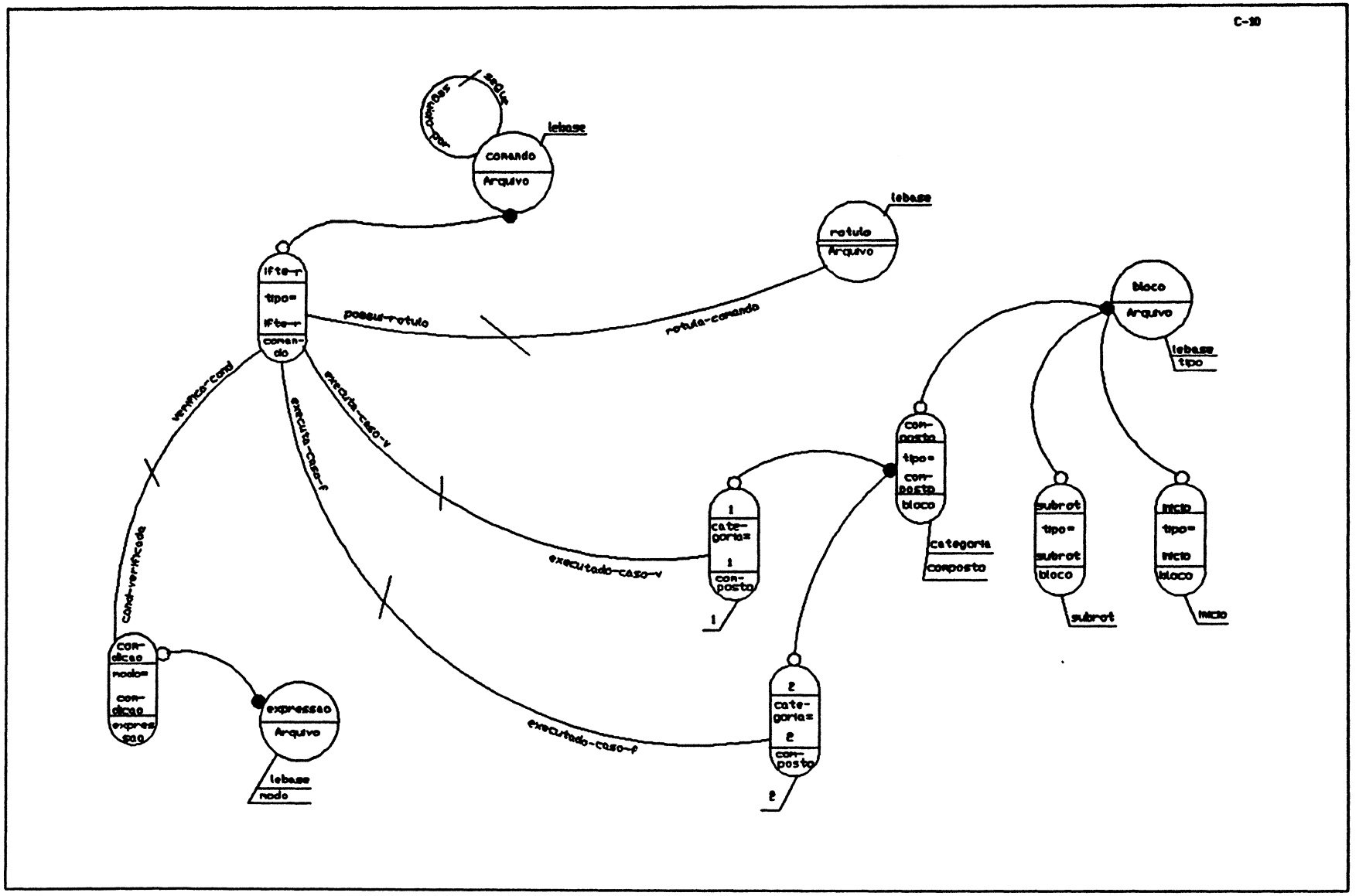

lebase
tipo
subrotina
Arquivo
principal
procedimento
funcao
lista-paran-subrot
tem-lista-paran
lista-paran-subrot
tem-lista-paran
declara-parametros
declara-parametros
paran-declarados
paran-declarados
lebase
tipo
declaracao
Arquivo
var
variavel
tipo=
var
declaracao
paran
parametro
tipo=
paran
declaracao
equivalente
mesmo-que
nome-do-par-lis
identifica-par-lis
nome-parametro
Arquivo
lebase
identifica-par
nome-do-par


C-6
comando
Arquivo
lebase
tipo
e_seguido_Por
segue
compoe-bloco
e-composto-por
bloco
Arquivo
lebase
tipo
while
tipo=
while
coman-
do
ifte
tipo=
ifte
coman-
do
ift
tipo=
ift
coman-
do
goto
tipo=
goto
coman-
do
atrib
tipo=
atrib
coman-
do
chan_
sub
tipo=
chan_
sub
coman-
do
chan_sub
verifica-cond
executa-caso-v
executa-caso-f
executado-caso-v
executado-caso-f
cond-verificada
desvia_para
define_desvio
rotulo
Arquivo
forma-atrib
possui-lis-expr
forma-lis-por
nome-chan-sub
identifica-chan
nome-
subrot
Arquivo
atri-
buicao
modo=
atri-
buicao
expres
-sao
descricao
lista-
expr
modo=
lista-
expr
expres
-sao
expressao
Arquivo
1
cate-
goria=
1
bloco
2
cate-
goria=
2
com-
posto
3
cate-
goria=
3
com-
posto
com-
posto
tipo=
com-
posto
bloco
categoria
composto
inicio
tipo=
inicio
bloco
subrot
tipo=
subrot
bloco
con-
dicao
modo=
con-
dicao
expres
-sao
expressao
Arquivo
lebase
modo

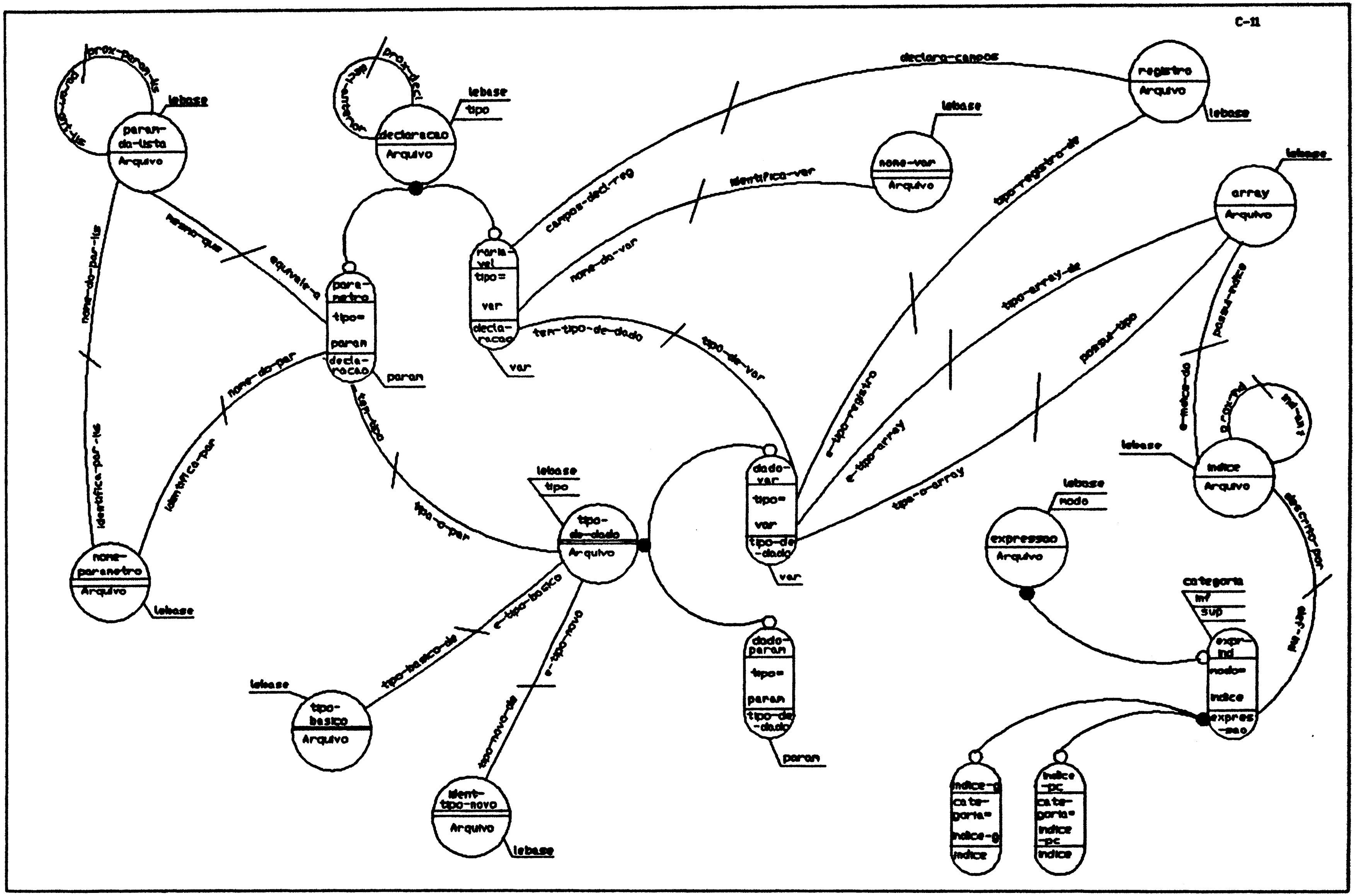

sequencia_Per
sons
lebase
tipo
comando
Arquivo
return
tipo=
return
conan-do
retorno
retorna-exp
exp-retornada
ifa
tipo=
ifa
conan-do
ifl
tipo=
ifl
conan-do
possui-cmd
possui-cmd
possui-cmd
cmd-do-ifl
cmd-do-ifl
cmd-do-ifl
chan_sub
tipo=
chan_sub
conan-do
chan_sub
atrib
tipo=
atrib
conan-do
atrib
goto
tipo=
goto
conan-do
goto
verifica-expr-arit
verifica-cond
cond-verificada
arit-verificada
rot1-de-ifa
rot2-de-ifa
rot3-de-ifa
desvia-rot3
desvia-rot2
desvia-rot1
rotulo
Arquivo
lebase
rot-torno
nodo=
rot-torno
expres-sao
expressao
Arquivo
lebase
nodo
con-dicao
nodo=
con-dicao
expres-sao
expressao
Arquivo
lebase
nodo
expr-aritm
nodo=
expr-aritm
expres-sao


lebase
tipo
comando
Arquivo
for
categoria
forfc
categoria=
forfc
for
forg
categoria=
forg
for
forc
categoria=
forc
for
for
tipo=
for
comando
executa-bloco
bloco-executado
expr-inicial
expr-parada
expr-incremento
inicio
nodo=
inicio
expressao
descricao
parada
nodo=
parada
expressao
descricao
incremento
nodo=
incremento
expressao
descricao
inicia-for
para-for
incrementa-for
case
tipo=
case
comando
case
verifica-case
caso-verificado
testa-case
nodo=
testa-case
expressao
descricao
ativa-blc
verifica-expr
expr-verificada
ative
nodo=
ative
expressao
descricao
2
2
categoria=
2
composto
3
categoria=
3
composto
3
composto
tipo=
composto
bloco
categoria
composto
lebase
tipo
bloco
Arquivo
subrot
tipo=
subrot
bloco
subrot
inicio
tipo=
inicio
bloco
inicio
bloco-selecionado

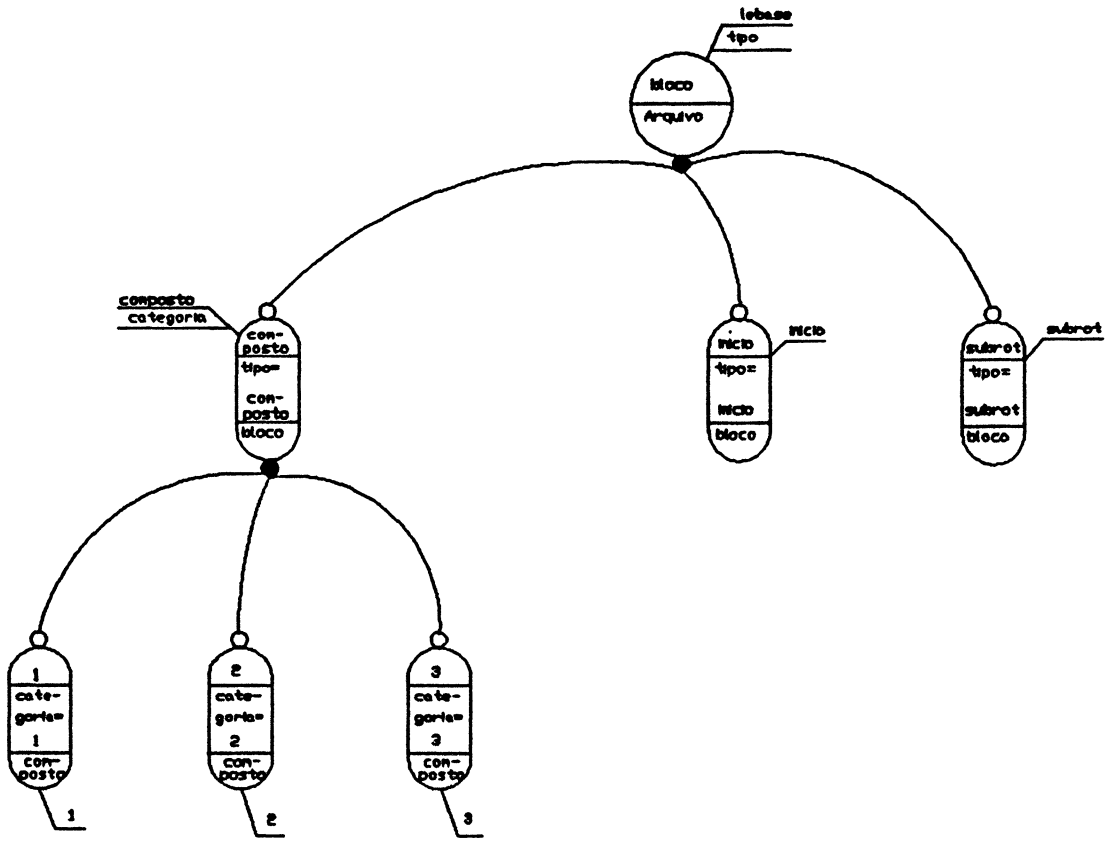

lebase
tipo
comando
Arquivo
compos-bloco
e-composto-por
do
do
tipo=
do
comando
repeat
repeat
tipo=
repeat
comando
bloco
Arquivo
lebase
tipo
verifica-cond
executa-blc
verifica-cond
executa-blc
cond-verificada
cond-verificada
blc-executado
blc-executado
3
categoria=
3
composto
3
composto
tipo=
composto
bloco
categoria
composto
subrot
tipo=
subrot
bloco
subrot
inicio
tipo=
inicio
bloco
inicio
condicao
nodo=
condicao
expressao
expressao
Arquivo
lebase
nodo
2
categoria
2
composto
2

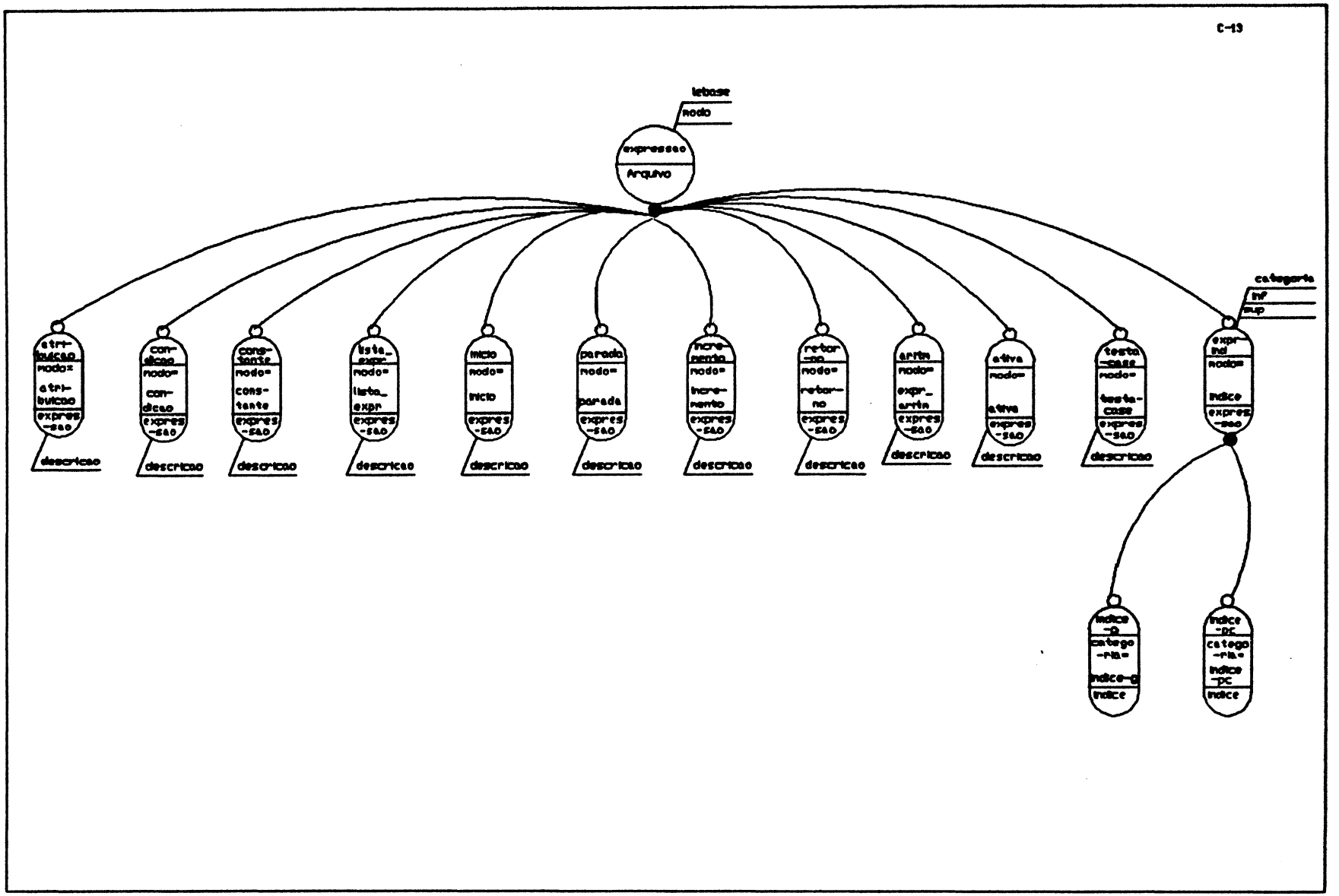

comando
Arquivo
lebase
sucessor
pai
ifte-r
tipo=
ifte-r
comando
rotulo
Arquivo
lebase
possui-rotulo
rotulo-comando
verifica-cond
cond-verificada
executa-caso-v
executado-caso-v
executa-caso-f
executado-caso-f
condicao
nodo=
condicao
expressao
expressao
Arquivo
lebase
nodo
1
categoria=
1
composto
1
2
categoria=
2
composto
2
composto
tipo=
composto
bloco
categoria
composto
bloco
Arquivo
lebase
tipo
subrot
tipo=
subrot
bloco
subrot
inicio
tipo=
inicio
bloco
inicio

C-11



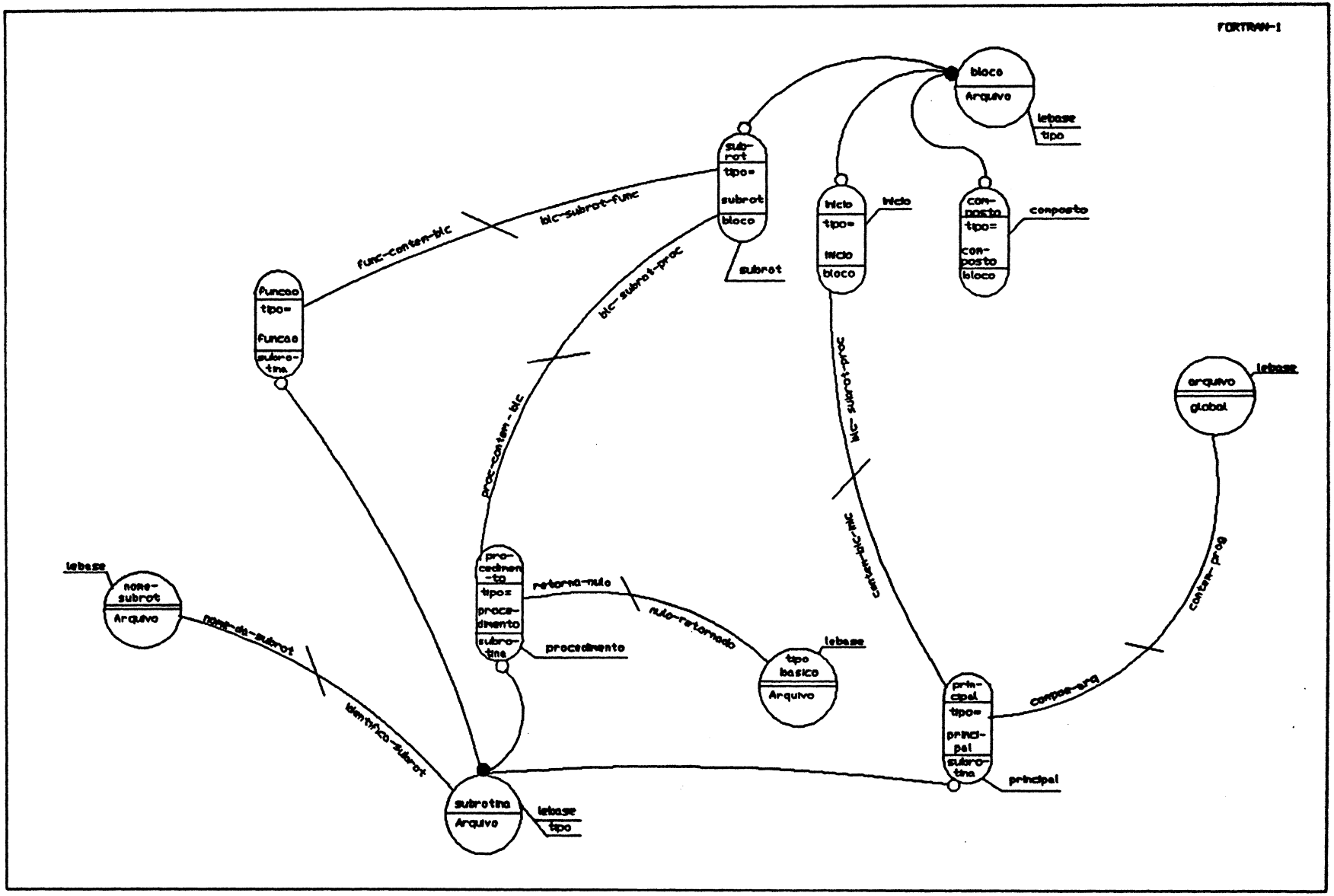
param-da-lista
Arquivo
lebase
prox-param-lis
declaracao
Arquivo
lebase
tipo
prox-decl
decl-anterior
registro
Arquivo
lebase
declara-campos
nome-var
Arquivo
lebase
array
Arquivo
lebase
parametro
tipo=
param
declaracao
param
variavel
tipo=
var
declaracao
var
campos-decl-reg
nome-da-var
identifica-var
ten-tipo-de-dado
tipo-de-var
tipo-registro-de
tipo-array-de
possui-tipo
possui-indice
e-tipo-registro
e-tipo-array
tipo-o-array
nome-de-par-lis
identifica-par-lis
nome-do-par
identifica-par
equivale-a
ten-tipo
tipo-o-par
nome-parametro
Arquivo
lebase
tipo-de-dado
Arquivo
lebase
tipo
dado-var
tipo=
var
tipo-de-dado
var
dado-param
tipo=
param
tipo-de-dado
param
indice
Arquivo
lebase
e-indice-do
descrito-por
expressao
Arquivo
lebase
modo
categoria
inf
sup
expr-ind
modo=
indice
expressao
tipo-basico
Arquivo
lebase
e-tipo-basico
tipo-basico-de
e-tipo-novo
tipo-novo-de
ident-tipo-novo
Arquivo
lebase
indice-g
categoria=
indice-g
indice
indice-pc
categoria=
indice-pc
indice

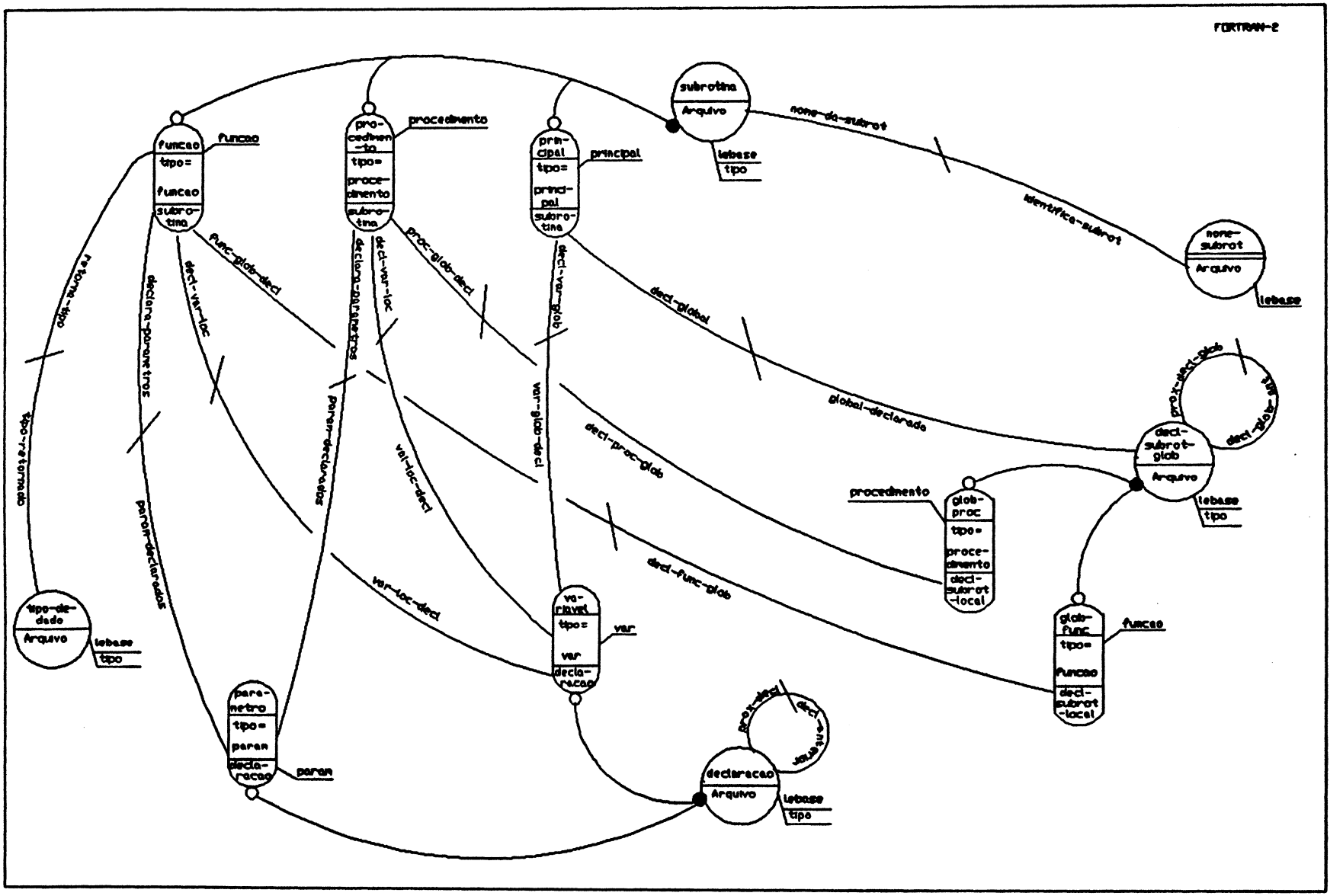

tobase
tipo
bloco
Arquivo
composto
categoria
com-
posto
tipo=
com-
posto
bloco
inicio
tipo=
inicio
bloco
subrot
tipo=
subrot
bloco
1
cate-
goria=
1
com-
posto
2
cate-
goria=
2
com-
posto
3
cate-
goria=
3
com-
posto
1
2
3

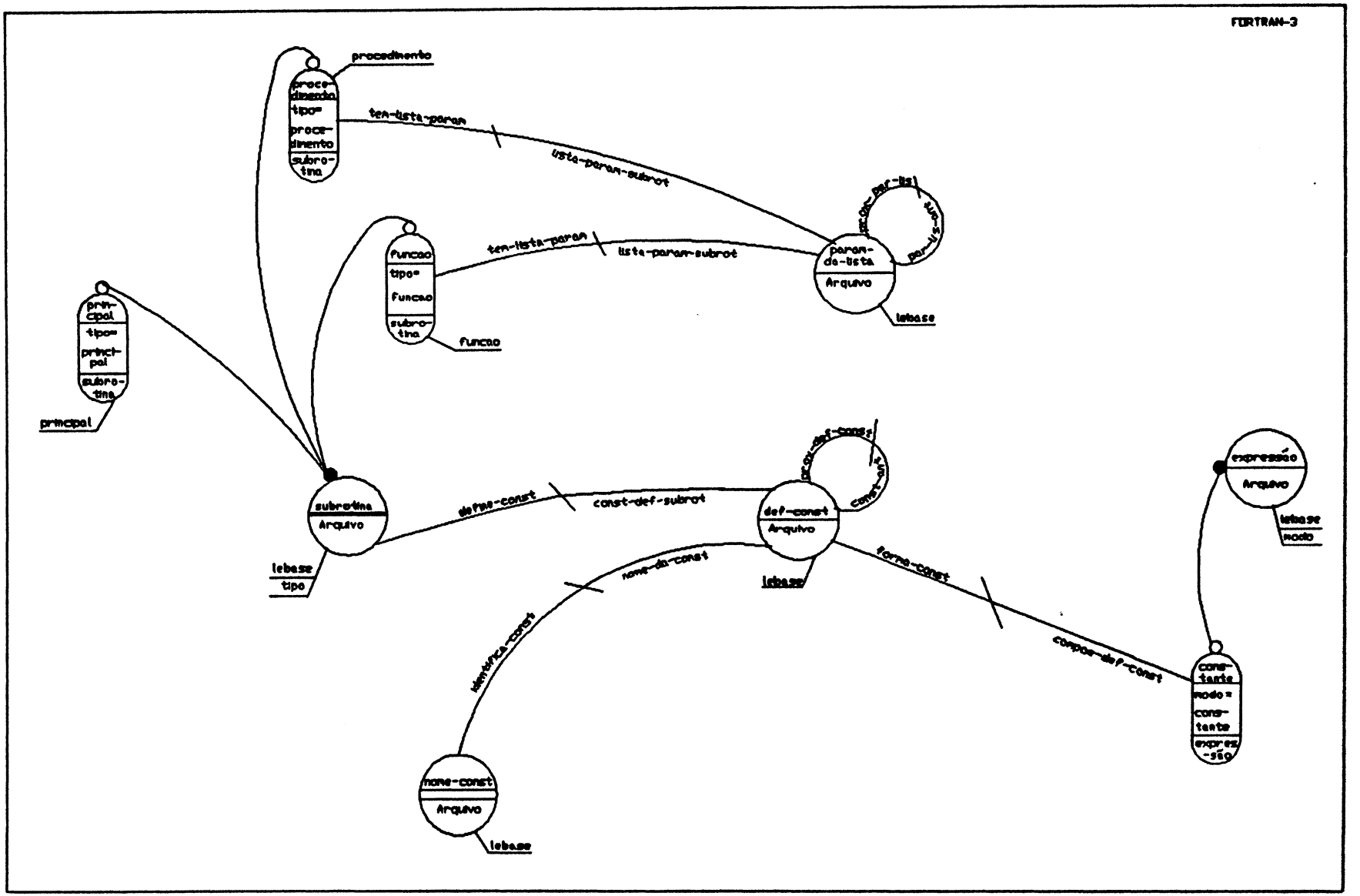

lebase
nodo
expressao
Arquivo
atri-buicao
nodo=
atri-buicao
expres-sao
descricao
con-dicao
nodo=
con-dicao
expres-sao
descricao
cons-tante
nodo=
cons-tante
expres-sao
descricao
lista_expr
nodo=
lista_expr
expres-sao
descricao
inicio
nodo=
inicio
expres-sao
descricao
parada
nodo=
parada
expres-sao
descricao
incre-mento
nodo=
incre-mento
expres-sao
descricao
retor-no
nodo=
retor-no
expres-sao
descricao
arrin
nodo=
expr_arrin
expres-sao
descricao
ativa
nodo=
ativa
expres-sao
descricao
testa-case
nodo=
testa-case
expres-sao
descricao
categoria
inf
sup
expr ind
nodo=
indice
expres-sao
indice-a
catego-ria=
indice-a
indice
indice-pc
catego-ria=
indice-pc
indice

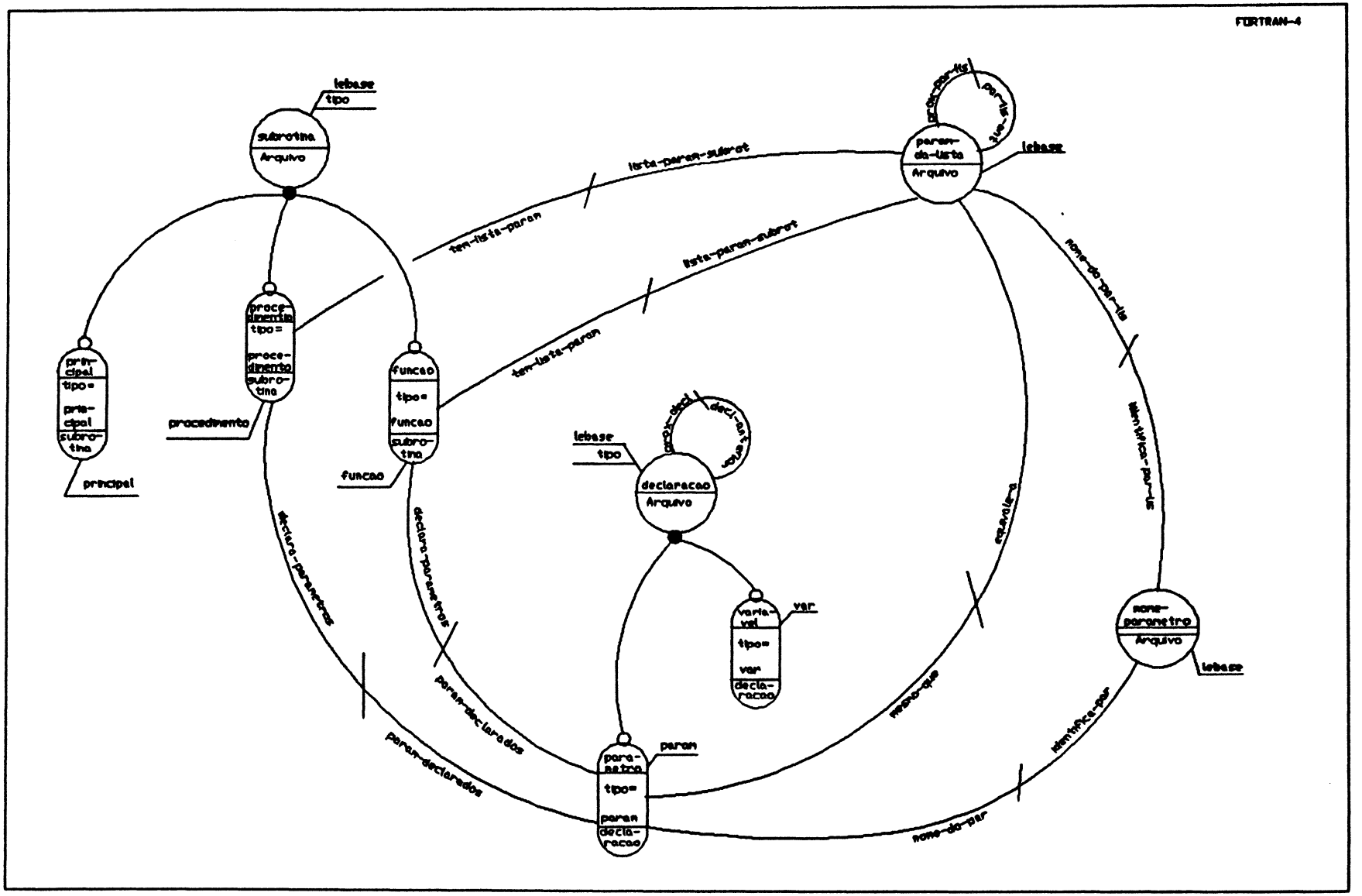
APÊNDICE B

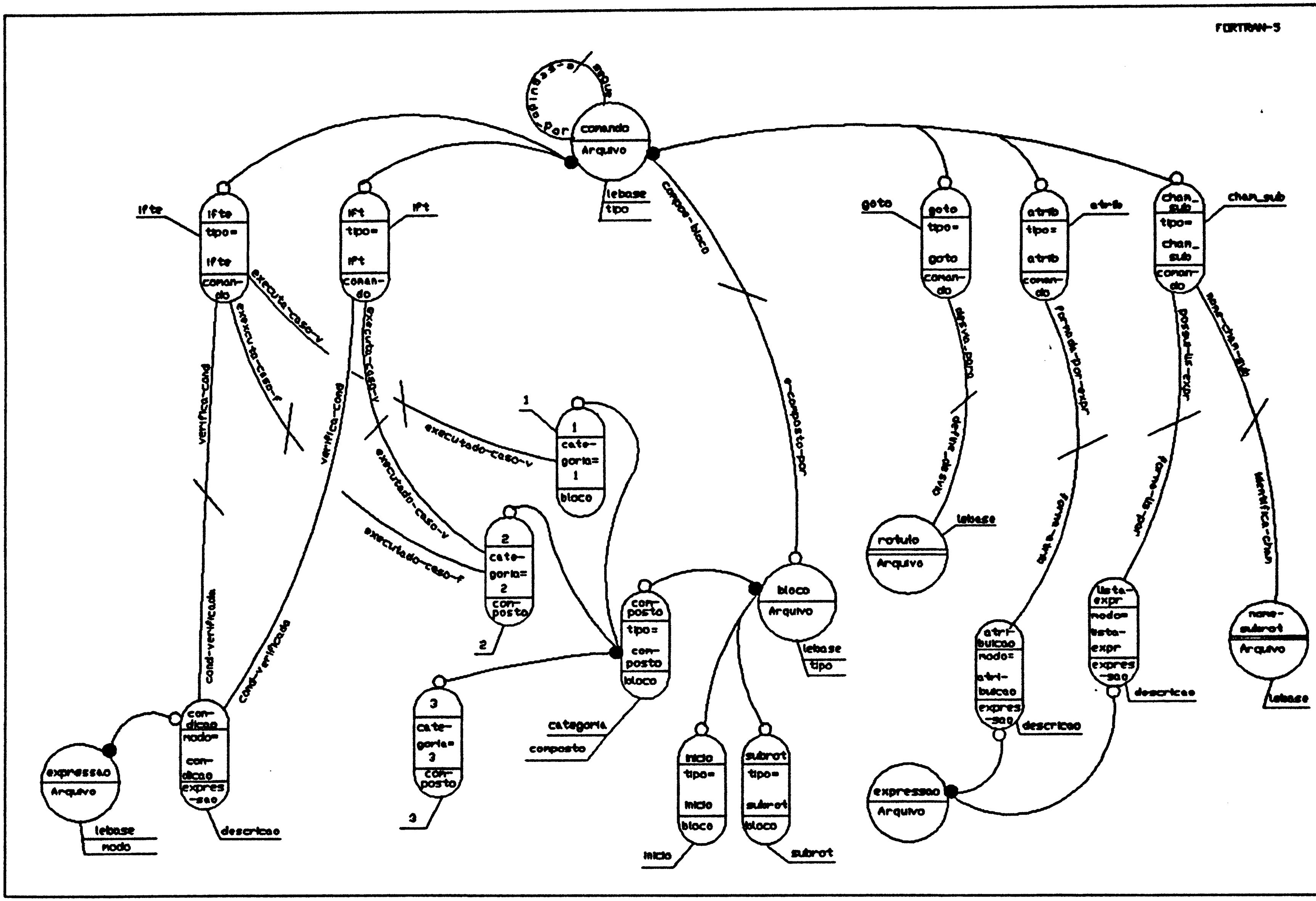
DLA
FORTRAN

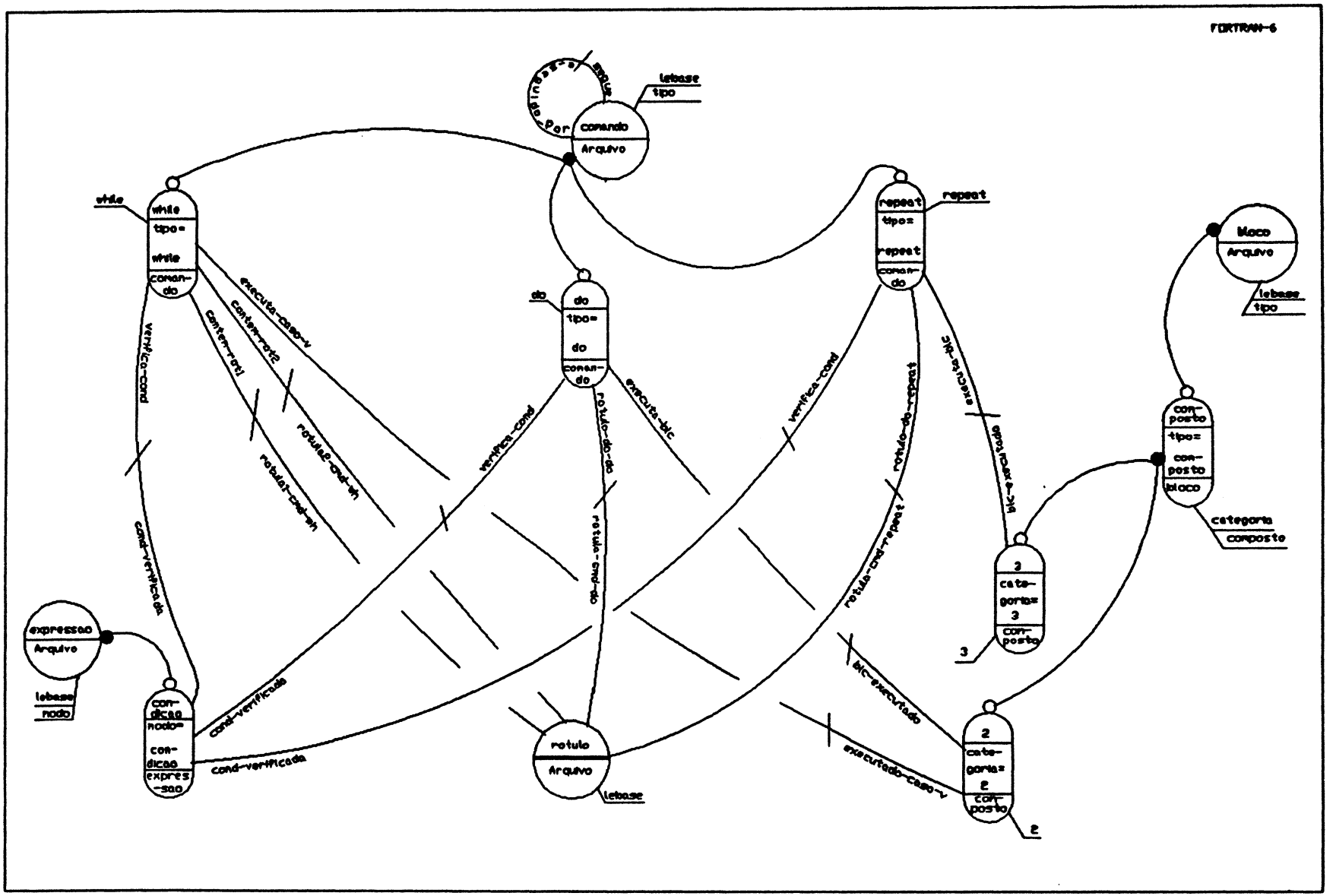

FORTRAN-1
bloco
Arquivo
lebase
tipo
sub-rot
tipo=
subrot
bloco
subrot
inicio
tipo=
inicio
bloco
inicio
com-posto
tipo=
com-posto
bloco
composto
funcao
tipo=
funcao
subro-tina
func-contem-blc
blc-subrot-func
blc-subrot-proc
proc-contem-blc
blc-subrot
contem-blc-blc
arquivo
global
lebase
contem-prog
pro-cedimen-to
tipo=
proce-dimento
subro-tina
procedimento
retorna-nulo
nulo-retornado
tipo basico
Arquivo
lebase
nome-subrot
Arquivo
lebase
nome-de-subrot
identifica-subrot
prin-cipal
tipo=
princi-pal
subro-tina
principal
compoe-arq
subrotina
Arquivo
lebase
tipo

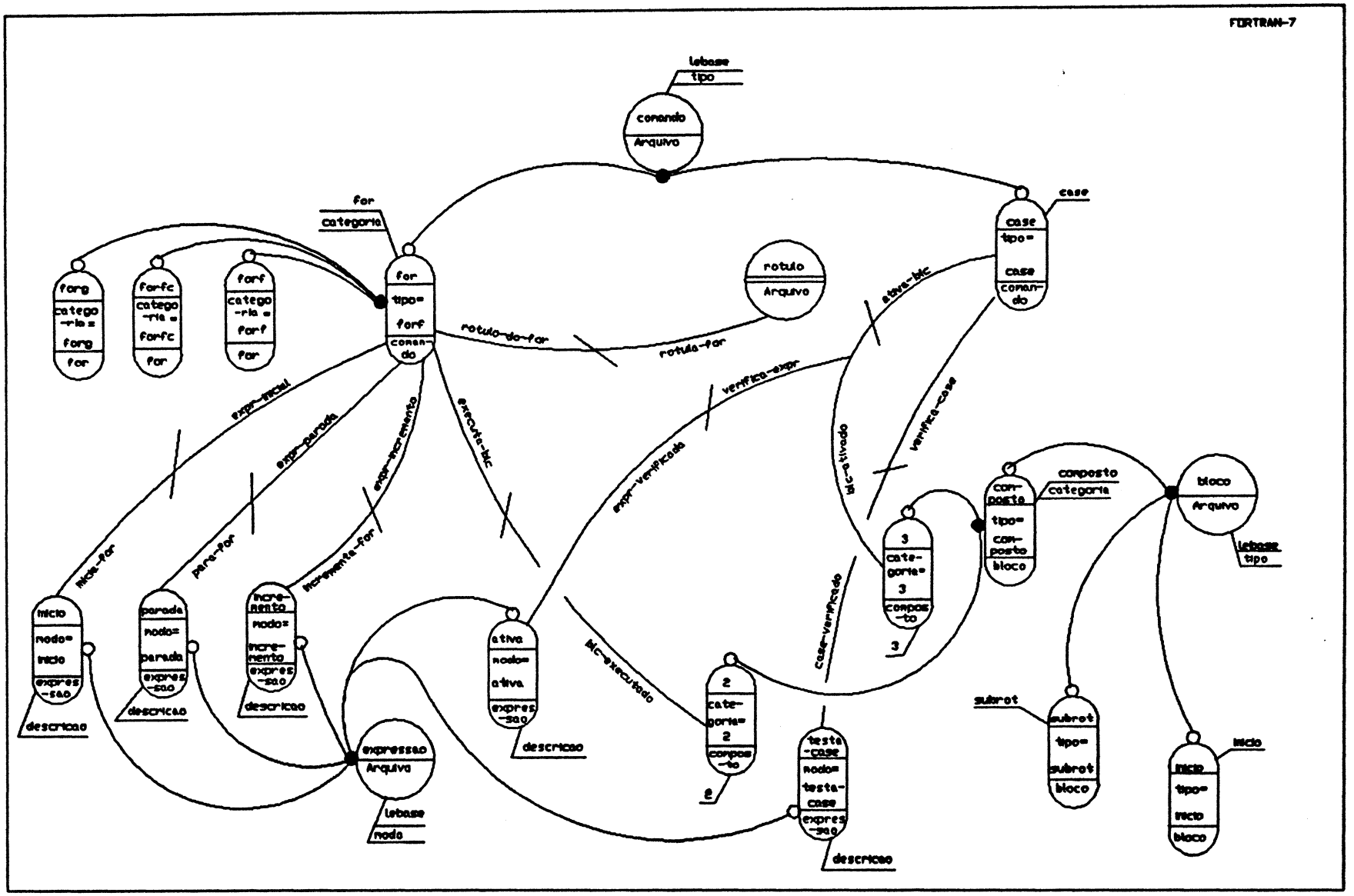

FORTRAN-2
funcao
tipo=
funcao
subro-tina
funcao
pro-cedimen-to
tipo=
proce-dimento
subro-tina
procedimento
prin-cipal
tipo=
princi-pal
subro-tina
principal
subrotina
Arquivo
lebase
tipo
nome-da-subrot
identifica-subrot
nome-subrot
Arquivo
lebase
func-globo-decl
decl-var-loc
declara-parametros
proc-globo-decl
decl-var-loc
declara-parametros
decl-var-glob
decl-global
global-declarado
prox-decl-glob
decl-glob
decl-subrot-glob
Arquivo
lebase
tipo
tipo-retornado
tipo-de-dado
Arquivo
lebase
tipo
parametros-declarados
parametros-declarados
var-loc-decl
var-glob-decl
decl-proc-glob
procedimento
glob-proc
tipo=
proce-dimento
decl-subrot-local
var-loc-decl
decl-func-glob
glob-func
tipo=
funcao
decl-subrot-local
funcao
va-riavel
tipo=
var
decla-racao
var
parametro
tipo=
param
decla-racao
param
prox-decl
decl-anterior
declaracao
Arquivo
lebase
tipo

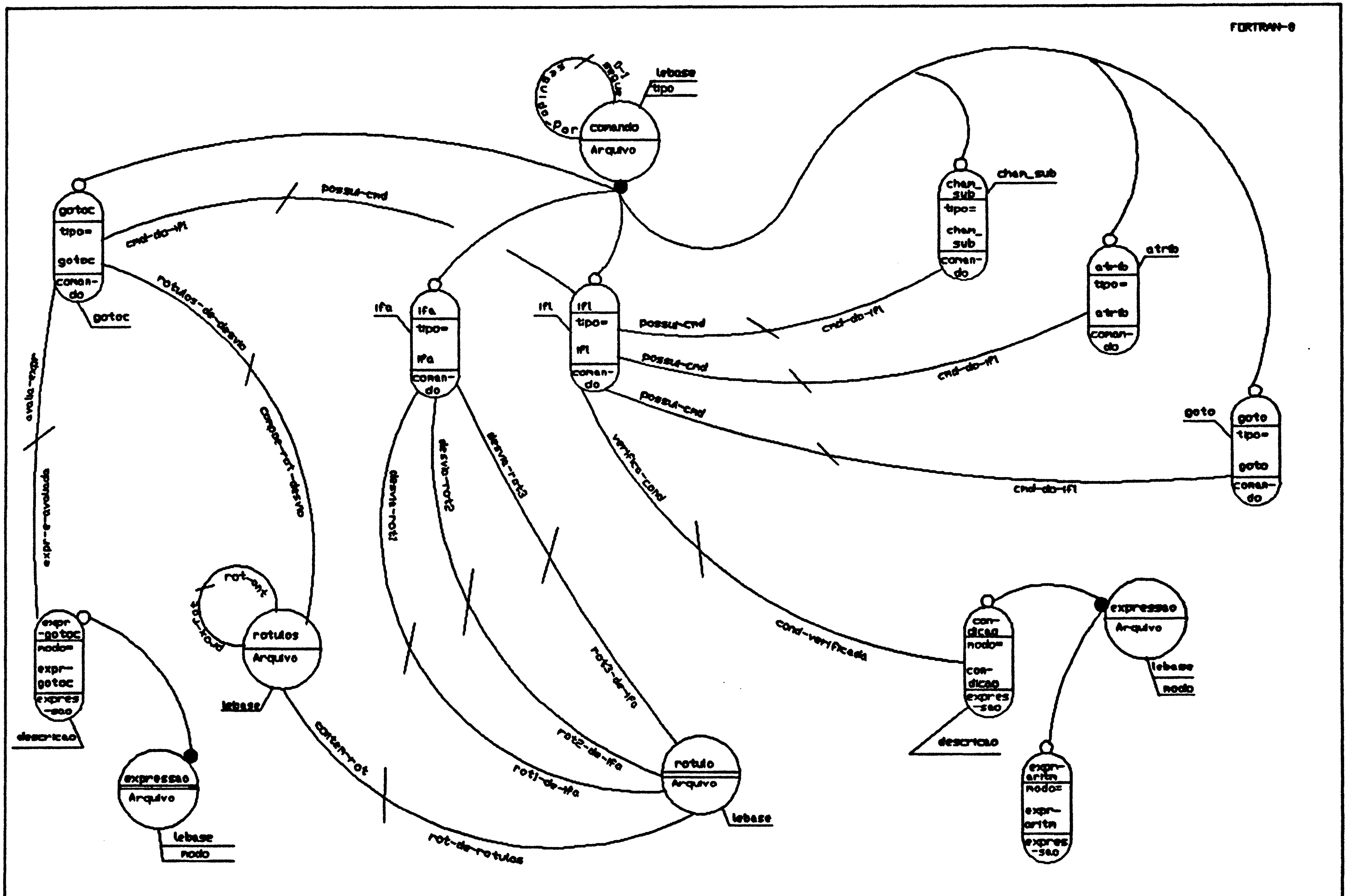

FORTRAN-3
procedimento
proce-dimento
tipo=
proce-dimento
subro-tina
tem-lista-paran
lista-paran-subrot
funcao
tipo=
funcao
subro-tina
funcao
tem-lista-paran
lista-paran-subrot
paran-de-lista
Arquivo
lebase
princ-par-lis
suces-par-lis
princi-pal
tipo=
princi-pal
subro-tina
principal
subrotina
Arquivo
lebase
tipo
defpa-const
const-def-subrot
def-const
Arquivo
lebase
princ-def-const
const-def
nome-da-const
identifica-const
nome-const
Arquivo
lebase
forma-const
compor-def-const
expressão
Arquivo
lebase
modo
cons-tante
modo=
cons-tante
expres-são

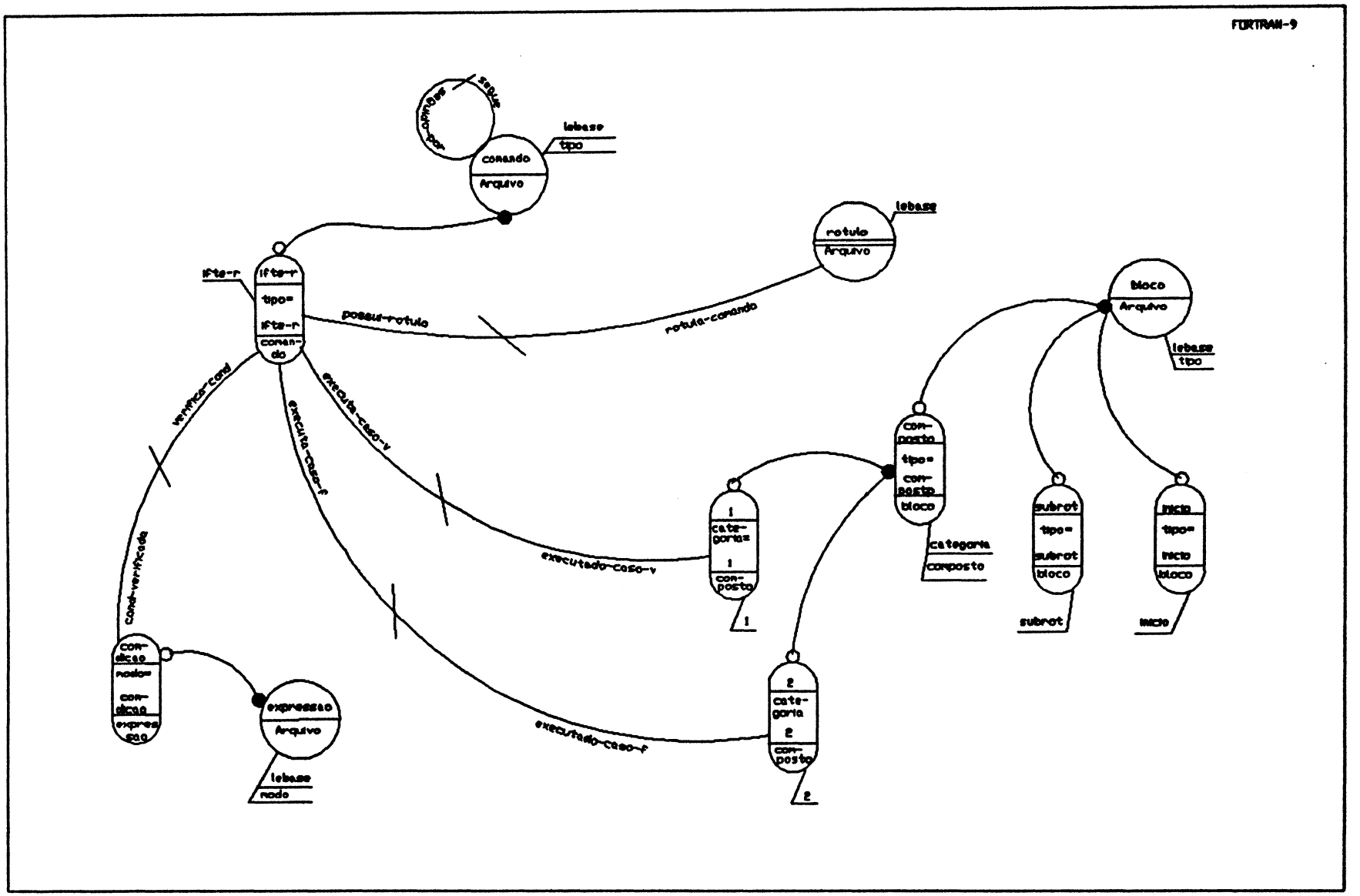

FORTRAN-4
lebase
tipo
subrotina
Arquivo
principal
tipo=
principal
subrotina
principal
procedimento
tipo=
procedimento
subrotina
procedimento
funcao
tipo=
funcao
subrotina
funcao
tem-lista-paran
lista-paran-subrot
declaracao
Arquivo
lebase
tipo
declara-parametros
declara-parametros
declara-decl
decl-anterior
variavel
tipo=
var
declaracao
var
parametro
tipo=
paran
declaracao
paran
paran-declarados
paran-declarados
paran-da-lista
Arquivo
lebase
declara-decl
decl-anterior
paran
lista-paran
lista-paran-subrot
nome-de-paran
identifica-paran
nome-parametro
Arquivo
lebase
identifica-par
nome-do-par

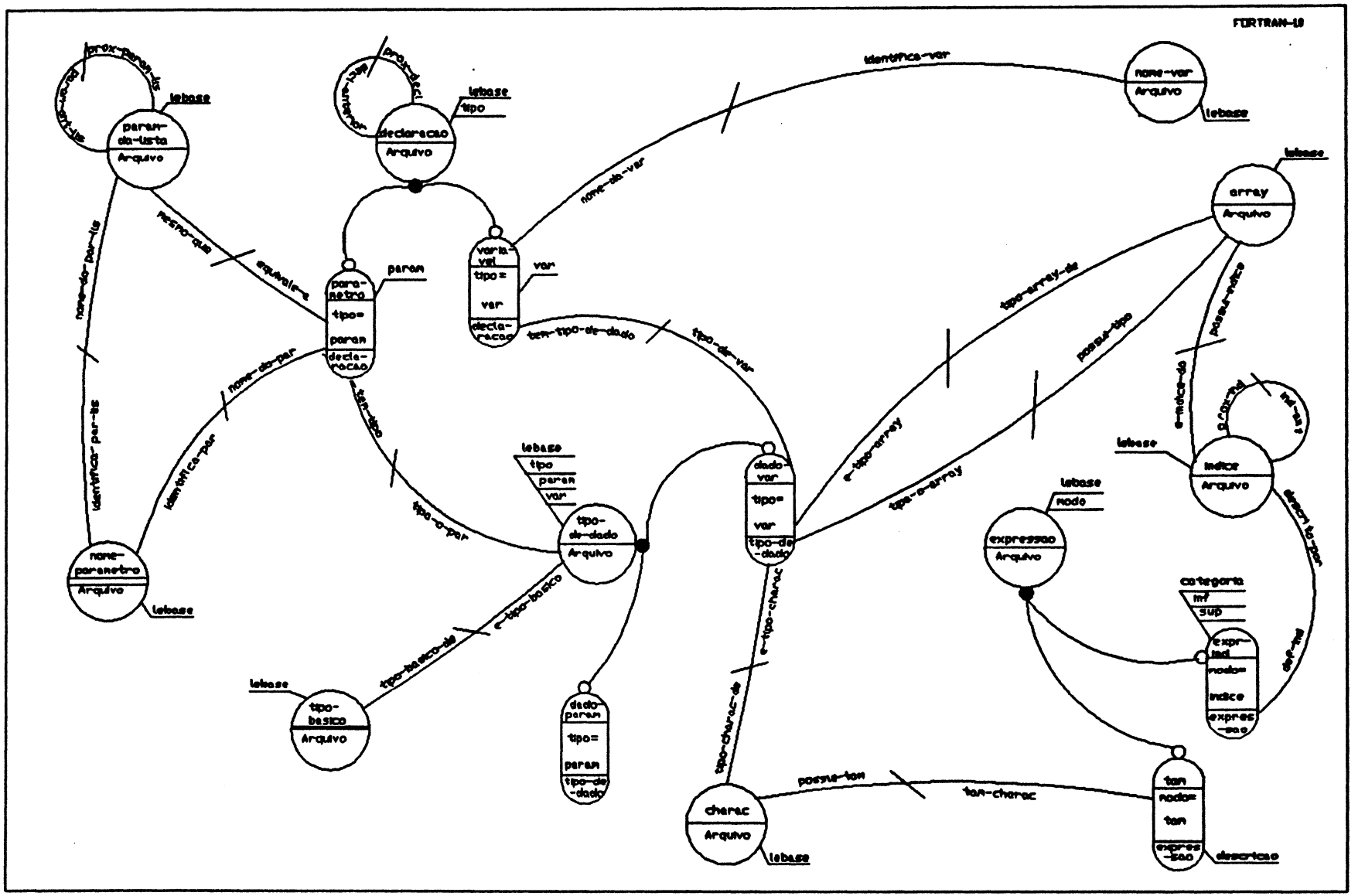

FORTRAN-5
comando
Arquivo
lebase
tipo
e-seguido_por
segue
ifte
tipo=
ifte
comando
ift
tipo=
ift
comando
goto
tipo=
goto
comando
atrib
tipo=
atrib
comando
chan_sub
tipo=
chan_sub
comando
compoe-bloco
e-composto-por
bloco
Arquivo
executa-caso-v
executa-caso-f
verifica-cond
executado-caso-v
executado-caso-f
cond-verificada
1
cate-
goria=
1
bloco
2
cate-
goria=
2
com-
posto
3
cate-
goria=
3
com-
posto
com-
posto
tipo=
com-
posto
bloco
categoria
composto
inicio
tipo=
inicio
bloco
subrot
tipo=
subrot
bloco
con-
dicao
modo=
con-
dicao
expres-
sao
descricao
expressao
Arquivo
lebase
modo
desvio_para
define_desvio
rotulo
Arquivo
lebase
formada-por-expr
forma-atrib
atri-
buicao
modo=
atri-
buicao
expres-
sao
descricao
possui-lis-expr
forma-lis-por
lista-
expr
modo=
lista-
expr
expres-
sao
descricao
nome-chan-sub
identifica-chan
nome-
subrot
Arquivo
lebase
expressao
Arquivo

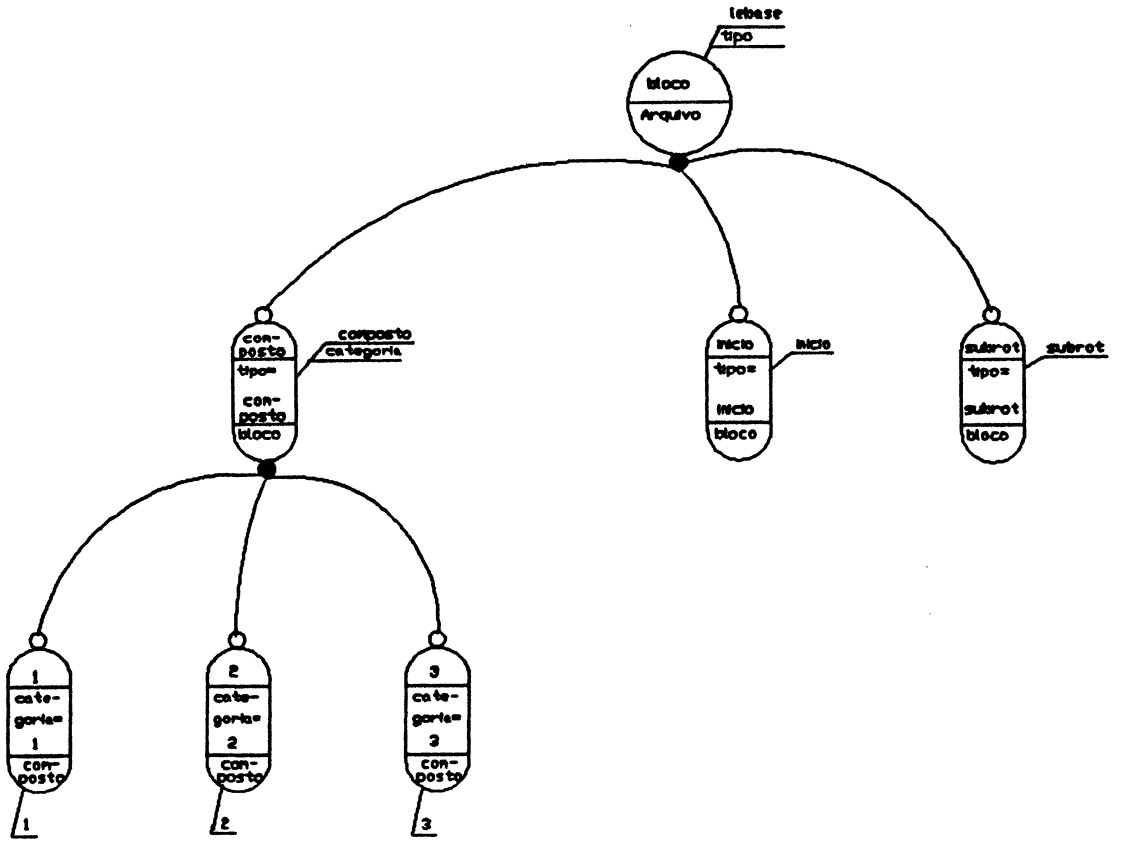

Segue-Por
comando
Arquivo
rotulo
tipo
while
while
tipo=
while
comando
do
do
tipo=
do
comando
repeat
repeat
tipo=
repeat
comando
bloco
Arquivo
rotulo
tipo
composto
tipo=
composto
bloco
categoria
composto
executa-caso-v
contem-rotv2
contem-rotv1
rotulo2-cnd-wh
rotulo1-cnd-wh
verifica-cond
cond-verificada
verifica-cond
rotulo-do-do
executa-blc
verifica-cond
rotulo-do-repeat
rotulo-cnd-repeat
executa-blc
expressao
Arquivo
rotulo
condicao
nodo=
condicao
expres-sao
cond-verificada
rotulo
Arquivo
rotulo
3
categoria=
3
composto
3
blc-executado
2
categoria=
2
composto
executado-caso-v
2

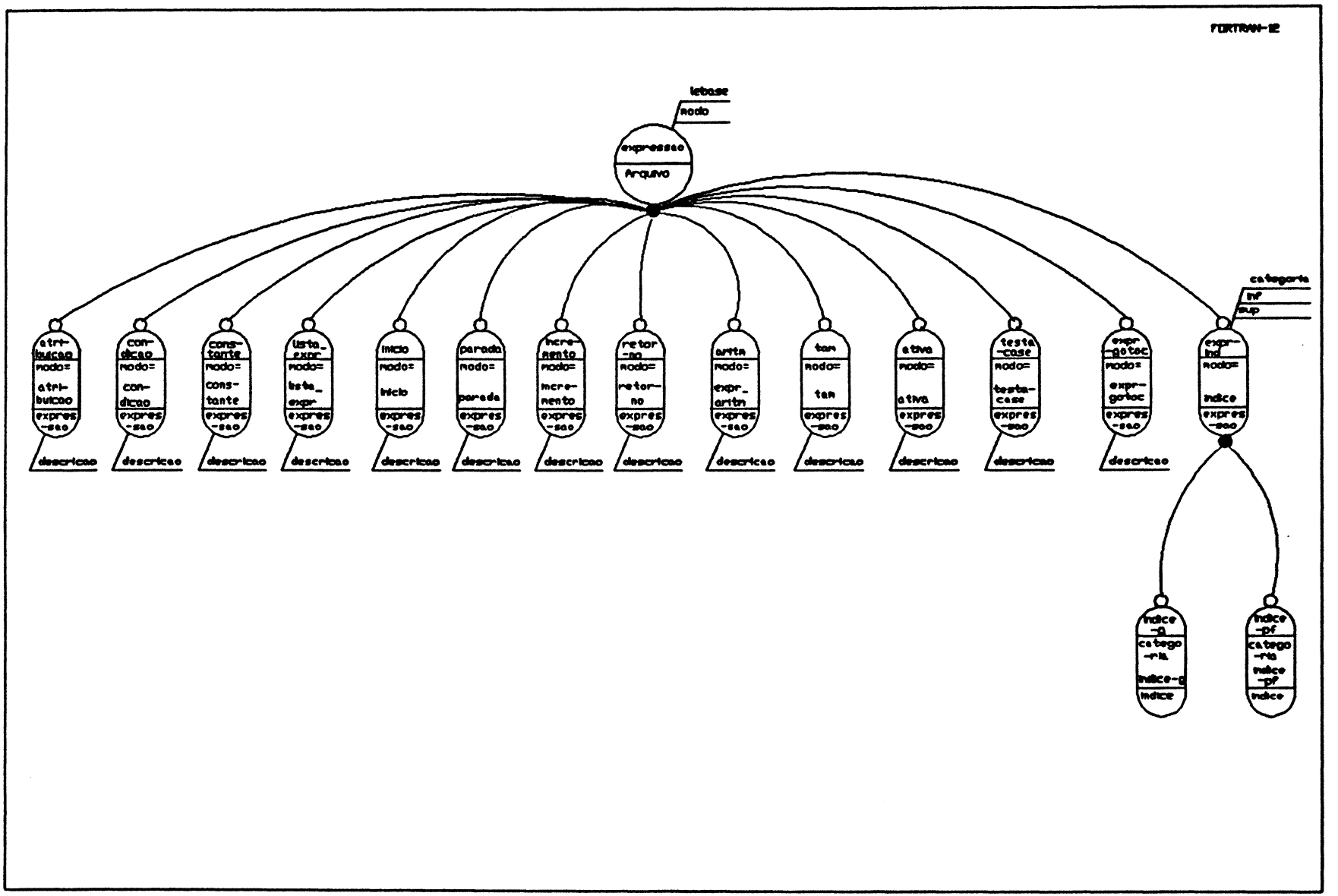

lebase
tipo
comando
Arquivo
for
categoria
case
for
tipo=
forf
comando
rotulo
Arquivo
rotulo-do-for
rotulo-for
forg
categoria=
forg
for
forfc
categoria=
forfc
for
forf
categoria=
forf
for
case
tipo=
case
comando
ativa-blc
verifica-case
verifica-expr
expr-inicial
expr-parada
expr-incremento
executa-blc
expr-verificada
blc-ativado
inicio-for
para-for
incrementa-for
composto
categoria
inicio
nodo=
inicio
expressao
descricao
parada
nodo=
parada
expressao
descricao
incremento
nodo=
incremento
expressao
descricao
ativa
nodo=
ativa
expressao
descricao
3
categoria=
3
composto
3
composto
tipo=
composto
bloco
bloco
Arquivo
lebase
tipo
blc-executado
case-verificado
2
categoria=
2
composto
2
subrot
subrot
tipo=
subrot
bloco
inicio
inicio
tipo=
inicio
bloco
expressao
Arquivo
lebase
nodo
teste-case
nodo=
teste-case
expressao
descricao


FORTRAN-8
comando
Arquivo
lebase
tipo
0-1
segue
seguido_por
gotoc
tipo=
gotoc
coman-
do
possui-cmd
cmd-do-ifl
rotulos-de-desvio
compoe-rot-desvio
avalia-expr
expr-e-avaliada
ifa
tipo=
ifa
coman-
do
ifl
tipo=
ifl
coman-
do
possui-cmd
cmd-do-ifl
verifica-cond
cond-verificada
desvia-rot1
desvia-rot2
desvia-rot3
rot1-de-ifa
rot2-de-ifa
rot3-de-ifa
chan_sub
tipo=
chan_
sub
coman-
do
atrib
tipo=
atrib
coman-
do
goto
tipo=
goto
coman-
do
rot-ant
prox-rot
rotulos
Arquivo
lebase
conten-rot
rot-de-rotulos
rotulo
Arquivo
lebase
expr
-gotoc
nodo=
expr-
gotoc
expres
-sao
descricao
expressao
Arquivo
lebase
nodo
con-
dicao
nodo=
con-
dicao
expres
-sao
descricao
expressao
Arquivo
lebase
nodo
expr-
aritm
nodo=
expr-
aritm
expres
-sao


sequencia-comandos
sequencia
comando
Arquivo
lebase
tipo
ifte-r
ifte-r
tipo=
ifte-r
comando
possui-rotulo
rotulo-comando
rotulo
Arquivo
lebase
verifica-cond
cond-verificada
executa-caso-v
executa-caso-f
executado-caso-v
executado-caso-f
bloco
Arquivo
lebase
tipo
composto
tipo=
composto
bloco
categoria
composto
subrot
tipo=
subrot
bloco
subrot
inicio
tipo=
inicio
bloco
inicio
1
categoria=
1
composto
1
2
categoria
2
composto
2
condicao
nodo=
condicao
expressao
expressao
Arquivo
lebase
nodo


FORTRAN-80
prox-parametro
lobase
parametro
Arquivo
prox-declaracao
declaracao
tipo
identifica-var
name-var
Arquivo
lobase
array
Arquivo
nome-do-parametro
mesmo-que
equivalente
parametro
tipo=
paran
declaracao
variavel
tipo=
var
declaracao
var
tem-tipo-de-dado
tipo-de-var
tipo-array-de
possui-indice
e-indice-de
proximo
ind-as?
indice
Arquivo
lobase
identificar-parametro
nome-do-par
ser-tipo
tipo-e-par
lobase
tipo
paran
var
tipo-de-dado
Arquivo
dado-var
tipo=
var
tipo-de-dado
e-tipo-array
possui-tipo
tipo-e-array
direita-por
def-ind
lobase
nodo
expressao
Arquivo
categoria
inf
sup
expr-ind
nodo=
indice
expressao
identificar-par
nome-parametro
Arquivo
lobase
e-tipo-basico
tipo-basico-de
lobase
tipo-basico
Arquivo
dado-paran
tipo=
paran
tipo-de-dado
e-tipo-charac
tipo-charac-de
possui-tan
tan-charac
charac
Arquivo
lobase
tan
nodo=
tan
expressao
descricao


tebase
tipo
bloco
Arquivo
com-
posto
tipo=
com-
posto
bloco
composto
categoria
inicio
tipo=
inicio
bloco
inicio
subrot
tipo=
subrot
bloco
subrot
1
cate-
goria=
1
com-
posto
2
cate-
goria=
2
com-
posto
3
cate-
goria=
3
com-
posto
1
2
3


lebase
nodo
expressao
Arquivo
atri-buicao
nodo=
atri-buicao
expres-sao
descricao
condicao
nodo=
condicao
expres-sao
descricao
cons-tante
nodo=
cons-tante
expres-sao
descricao
lista_ expr
nodo=
lista_ expr
expres-sao
descricao
inicio
nodo=
inicio
expres-sao
descricao
parada
nodo=
parada
expres-sao
descricao
incre-mento
nodo=
incre-mento
expres-sao
descricao
retor-no
nodo=
retor-no
expres-sao
descricao
artm
nodo=
expr_ artm
expres-sao
descricao
tam
nodo=
tam
expres-sao
descricao
ativa
nodo=
ativa
expres-sao
descricao
testa-case
nodo=
testa-case
expres-sao
descricao
expr-goiac
nodo=
expr-goiac
expres-sao
descricao
categoria
inf
sup
expr-ind
nodo=
indice
expres-sao
indice-a
catego-ria
indice-a
indice
indice-pf
catego-ria
indice-pf
indice

APÊNDICE C

Esquemas de Recuperação de Programas das Linguagens Pascal, C e FORTRAN

Considerações:

*Mostra-se as informações utilizadas pelo Editor para a recuperação e apresentação de programas na linguagem **Pascal C e FORTRAN**. Tais informações compõem o ERPPas, ERPC e ERPFOR.*

APÊNDICE C

ERPPas

tipo do objeto	tipo do atributo	atributos	Pascal
arquivo	lebase	RELA contém_prog FIM	
subrotina	lebase	TIPO FIM	
principal	principal	INSE program @; 0 INSE <nome> 1 RELA nome_da_subrot INSE label @; 0 INSE <decl.rótulo> 1 RELA decl_rótulos INSE const -1 INSE <define const.> 1 RELA define_const INSE type -1 INSE <define tipo> 1 RELA define_tipo INSE var -1 INSE <declara var.> 1 RELA decl_var_glob INSE <decl.subrot.> -1 RELA decl_global INSE <bloco> 0 RELA contém_blc_inic FIM	

tipo do objeto	tipo do atributo	atributos	Pascal
procedi-mento	procedi-mento	SUBS procedure @(@); 0 INSE <nome> 1 RELA nome_da_subrot INSE <decl.parâm.> 1 RELA declara_parâmetros INSE label @; -1 INSE <decl.rótulo> 1 RELA decl_rótulos INSE const -1 INSE <define const.> 1 RELA define_const INSE type -1 INSE <define tipo> 1 RELA define_tipo INSE var -1 INSE <declara var.> 1 RELA decl_var_loc INSE <decl.subrot.> -1 RELA decl_local INSE <bloco> 0 RELA proc_contém_blc FIM	
			2

tipo do objeto	tipo do atributo	atributos	Pascal
função	função	SUBS function @(@):@; 0 INSE <nome> 1 RELA nome_da_subrot INSE <decl.parâm.> 1 RELA declara_parâmetros RELA retorna_tipo INSE label @; -1 INSE <decl.rótulo> 1 RELA decl_rótulos INSE const -1 INSE <define const.> 1 RELA define_const INSE type -1 INSE <define tipo> 1 RELA define_tipo INSE var -1 INSE <declara var.> 1 RELA decl_var_loc INSE <decl.subrot.> -1 RELA decl_local INSE <bloco> 0 RELA func_contém_blc FIM	

tipo do objeto	tipo do atributo	atributos	Pascal
nome_ subrot	lebase	IDEN FIM	
decl_ rótulo	lebase	SUBS @,@ 0 INSE <rótulo> 1 RELA cria_rótulo INSE <decl.rótulo> 1 RELA prox_decl_rót. FIM	
def_const	lebase	SUBS @=@; 0 INSE <nome> 1 RELA nome_da_const INSE <constante> 1 RELA forma_const INSE <define const.>-2 RELA prox_def_const FIM	
def_tipo	lebase	SUBS @=@; 0 INSE <nome> 1 RELA nome_do_def_novo RELA contém_tipo INSE <define tipo> -1 RELA prox_def_tipo FIM	
rótulo	lebase	IDEN FIM	
ident_ tipo_ novo	lebase	IDEN FIM	
nome_var	lebase	IDEN FIM	

tipo do objeto	tipo do atributo	atributos	Pascal
declaração	lebase	TIPO FIM	
variável	var	SUBS @:@; 0 INSE <nome> 1 RELA nome_da_var RELA tem_tipo_de_dado INSE <declara var.>-1 RELA prox_decl FIM	
parâmetro	parâm	SUBS @:@; @ 0 INSE <nome> 1 RELA nome_do_par RELA tem_tipo INSE <decl.parâm.>100 RELA prox_decl FIM	
decl_ subrot_ glob	lebase	TIPO FIM	
proc_glob	procedi-mento	RELA decl_proc_glob RELA prox_decl_glob FIM	
func_glob	função	RELA decl_func_glob RELA prox_decl_glob FIM	
nome_const	lebase	IDEN FIM	

tipo do objeto	tipo do atributo	atributos	Pascal
bloco	lebase	TIPO FIM	
início	início	SUBS begin 0 INSE <comando> 1 RELA e_composto_por INSE end. -1 FIM	
subrot	subrot	SUBS begin 0 INSE <comando> 1 RELA e_composto_por INSE end; -1 FIM	
composto	composto	ATRI categoria FIM	
1	1	SUBS begin 0 INSE <comando> 1 RELA e_composto_por INSE end -1 FIM	
2	2	SUBS begin 0 INSE <comando> 1 RELA e_composto_por INSE end; -1 FIM	
3	3	INSE <comando> 1 RELA e_composto_por FIM	
			6

tipo do objeto	tipo do atributo	atributos	Pascal
decl_ subrot_ loc	lebase	TIPO FIM	
proc_loc	procedi- mento	RELA decl_proc_loc RELA prox_decl_loc FIM	
func_loc	função	RELA decl_func_loc RELA prox_decl_loc FIM	
tipo_de_ dado	lebase	TIPO FIM	
dado_var	var	INSE <tipo> 1 RELA é_tipo_registro RELA é_tipo_array RELA é_tipo_novo RELA é_tipo_básico FIM	
dado_parâm	parâm	INSE <tipo> 1 RELA é_tipo_novo RELA é_tipo_básico FIM	
nome_ parâmetro	lebase	IDEN FIM	
tipo_ básico	lebase	IDEN FIM	
			7

tipo do objeto	tipo do atributo	atributos	Pascal
registro	lebase	SUBS record 0 INSE <declara var.> 1 RELA declara_campos INSE end; 0 FIM	
array	lebase	SUBS array[@] of @ 0 INSE <índices> 1 RELA possui_índice RELA possui_tipo FIM	
índice	lebase	SUBS @, @ 0 INSE <índice> 1 RELA forma_índice INSE <índices> 1 RELA prox_ind FIM	
expressão	lebase	ATRI modo FIM	
atribuição	modo	ATRI descrição FIM	
condição	modo	ATRI descrição FIM	
constante	modo	ATRI descrição FIM	
lista_expr	modo	ATRI descrição FIM	
início	modo	ATRI descrição FIM	
parada	modo	ATRI descrição FIM	

tipo do objeto	tipo do atributo	atributos	Pascal
incremento	modo	ATRI descrição FIM	
retorno	modo	ATRI descrição FIM	
expr_aritm	modo	ATRI descrição FIM	
ativa	modo	ATRI descrição FIM	
expr_ind	modo	ATRI categoria ATRI inf ATRI sup FIM	
comando	lebase	TIPO FIM	
ift	ift	SUBS if(@) 0 INSE <condição> 1 RELA verifica_cond INSE then 0 INSE <bloco> 1 RELA executa_caso_v INSE <comando> -2 RELA segue FIM	
ifl	ifl	SUBS if(@) 0 INSE <condição> 1 RELA verifica_cond INSE then 0 INSE <comando> 1 RELA possui_cmd INSE <comando> -2 RELA segue FIM	
goto	goto	SUBS goto @; 0 INSE <rótulo> 1 RELA desvia_para INSE <comando> -1 RELA segue FIM	

tipo do objeto	tipo do atributo	atributos	Pascal
ifte	ifte	SUBS if(@) 0 INSE <condição> 1 RELA verifica_cond INSE then 0 INSE <bloco> 1 RELA executa_caso_v INSE else -1 INSE <bloco> 1 RELA executa_caso_f INSE <comando> -2 RELA segue FIM	
cham_sub	cham_sub	SUBS @(@); 0 INSE <nome> 1 RELA nome_cham_sub INSE <lista expr.> 1 RELA possui_lis_expr INSE <comando> -2 RELA segue FIM	
ifa	ifa	SUBS IF(@=0) 0 INSE <expr.> 1 RELA verifica_expr_arit INSE goto @; 0 INSE <rótulo> 1 RELA desvia_rot1 INSE else if(@>0) -1 INSE <expr.> 1 RELA verifica_expr_arit INSE goto @; 0 INSE <rótulo> 1 RELA desvia_rot2 INSE else -1 INSE goto @; 1 INSE <rótulo> 1 RELA desvia_rot3 INSE <comando> -4 RELA segue FIM	
			10

tipo do objeto	tipo do atributo	atributos	Pascal
for	for	SUBS for @ to @ do 0 INSE <expr.> 1 RELA expr_inicial INSE <expr.> 1 RELA expr_parada INSE <bloco> -1 RELA executa_bloco INSE <comando> -1 RELA segue FIM	
repeat	repeat	SUBS repeat 0 INSE <bloco> 1 RELA executa_blc INSE until(@); 0 INSE <condição> 1 RELA verifica_cond INSE <comando> -2 RELA segue FIM	
while	while	SUBS while(@)do 0 INSE <condição> 1 RELA verifica_cond INSE <bloco> 0 RELA executa_caso_v INSE <comando> -1 RELA segue FIM	
do	do	SUBS repeat 0 INSE <bloco> 1 RELA executa_blc INSE until(not @); 0 INSE <condição> 1 RELA verifica_cond INSE <comando> -2 RELA segue FIM	
atrib	atrib	SUBS @; 0 INSE <atribuição> 1 RELA formada_por_expr INSE <comando> -1 RELA segue FIM	

tipo do objeto	tipo do atributo	atributos	Pascal
ift_r	ift_r	SUBS @:if(@) 0 INSE <rótulo> 1 RELA possui_rótulo INSE <condição> 1 RELA verifica_cond INSE then 0 INSE <bloco> 1 RELA executa_caso_v INSE <comando> -3 RELA segue FIM	
ifte_r	ifte_r	SUBS @:if(@) 0 INSE <rótulo> 1 RELA possui_rótulo INSE <condição> 1 RELA verifica_cond INSE then 0 INSE <bloco> 1 RELA executa_caso_v INSE else -1 INSE <bloco> 1 RELA executa_caso_f INSE <comando> -2 RELA segue FIM	
goto_r	goto_r	SUBS @:goto @; 0 INSE <rótulo> 1 RELA possui_rótulo INSE <rótulo> 1 RELA desvia_para INSE <comando> -1 RELA segue FIM	
atrib_r	atrib_r	SUBS @:@; 0 INSE <rótulo> 1 RELA possui_rótulo INSE <atribuição> 1 RELA formada_por_expr INSE <comando> -1 RELA segue FIM	

tipo do objeto	tipo do atributo	atributos	Pascal
for_r	for_r	SUBS @:for @ to @ do 0 INSE <rótulo> 1 RELA possui_rótulo INSE <expr.> 1 RELA expr_inicial INSE <expr.> 1 RELA expr_parada INSE <bloco> -1 RELA executa_bloco INSE <comando> -1 RELA segue FIM	
repeat_r	repeat_r	SUBS @:repeat 0 INSE <rótulo> 1 RELA possui_rótulo INSE <bloco> 1 RELA executa_blc INSE until(@); 0 INSE <condição> 1 RELA verifica_cond INSE <comando> -2 RELA segue FIM	
while_r	while_r	SUBS @:while(@)do 0 INSE <rótulo> 1 RELA possui_rótulo INSE <condição> 1 RELA verifica_cond INSE <bloco> 0 RELA executa_caso_v INSE <comando> -1 RELA segue FIM	
do_r	do_r	SUBS @:repeat 0 INSE <rótulo> 1 RELA possui_rótulo INSE <bloco> 1 RELA executa_blc INSE until(not @); 0 INSE <condição> 1 RELA verifica_cond INSE <comando> -2 RELA segue FIM	

tipo do objeto	tipo do atributo	atributos	Pascal
cham_sub_r	cham_sub_r	SUBS @:@(@); 0 INSE <rótulo> 1 RELA possui_rótulo INSE <nome> 1 RELA nome_cham_sub INSE <lista expr.> 1 RELA possui_lis_expr INSE <comando> -2 RELA segue FIM	
ifl_r	ifl_r	SUBS @:if(@) 0 INSE <rótulo> 1 RELA possui_rótulo INSE <condição> 1 RELA verifica_cond INSE then 0 INSE <comando> 1 RELA possui_cmd INSE <comando> -3 RELA segue FIM	
ifa_r	ifa_r	SUBS @:if(@=0) 0 INSE <rótulo> 1 RELA possui_rótulo INSE <expr.> 1 RELA verifica_expr_arit INSE goto @; 0 INSE <rótulo> 1 RELA desvia_rot1 INSE else if(@>0) -1 INSE <expr.> 1 RELA verifica_expr_arit INSE goto @; 0 INSE <rótulo> 1 RELA desvia_rot2 INSE else -1 INSE goto @; 1 INSE <rótulo> 1 RELA desvia_rot3 INSE <comando> -5 RELA segue FIM	

APÊNDICE C

ERPC

tipo do objeto	tipo do atributo	atributos	C
arquivo	lebase	RELA contém_prog FIM	
subrotina	lebase	TIPO FIM	
principal	principal	INSE <define const.> 0 RELA define_const INSE <define tipo> 0 RELA define_tipo INSE <declara var.> 0 RELA decl_var_glob INSE <declara função> 0 RELA decl_glob INSE main() 0 INSE { 1 INSE <define const.> 1 RELA define_const INSE <define tipo> 0 RELA define_tipo_loc INSE <declara var.> 0 RELA decl_var_loc INSE <comando> 0 RELA contém_blc_inic INSE } -1 FIM	
			1

tipo do objeto	tipo do atributo	atributos	C
procedi-mento	procedi-mento	SUBS @ @(@) 0 INSE <tipo> 1 RELA retorna_tipo_nulo INSE <nome> 1 RELA nome_da_subrot INSE <lista parâm.> 1 RELA tem_lista_parâm INSE <decl.parâm> -2 RELA declara_parâmetros INSE { 0 INSE <define const.> 1 RELA define_const INSE <define tipo> 0 RELA define_tipo INSE <declara var.> 0 RELA decl_var_loc INSE <comando> 0 RELA proc_contém_blc INSE } -1 FIM	
			2

tipo do objeto	tipo do atributo	atributos	C
função	função	SUBS @ @(@) 0 RELA retorna_tipo INSE <nome> 1 RELA nome_da_subrot INSE <lista parâm.> 1 RELA tem_lista_parâm INSE <decl.parâm> -2 RELA declara_parâmetros INSE { 0 INSE <define const.> 1 RELA define_const INSE <define tipo> 0 RELA define_tipo INSE <declara var.> 0 RELA decl_var_loc INSE <comando> 0 RELA func_contém_blc INSE } -1 FIM	
			3

tipo do objeto	tipo do atributo	atributos	C
nome_ subrot	lebase	IDEN FIM	
def_const	lebase	SUBS #define @ @ 0 INSE <nome> 1 RELA nome_da_const INSE <constante> 1 RELA forma_const INSE <define const.>-2 RELA prox_def_const FIM	
def_tipo	lebase	SUBS typedef 0 INSE @ @; 1 RELA contém_tipo INSE <nome> 100 RELA nome_do_tipo_novo INSE <define tipo> 99 RELA prox_def_tipo FIM	
rótulo	lebase	IDEN FIM	
ident_ tipo_ novo	lebase	IDEN FIM	
nome_var	lebase	IDEN FIM	
nome_const	lebase	IDEN FIM	
nome_ parâmetro	lebase	IDEN FIM	
			4

tipo do objeto	tipo do atributo	atributos	C
declaração	lebase	TIPO FIM	
variável	var	SUBS @ @; 0 RELA tem_tipo_de_dado INSE <nome> 100 RELA nome_da_var INSE <declara var.> 99 RELA prox_decl FIM	
parâmetro	parâm	SUBS @ @; 0 RELA tem_tipo INSE <nome> 100 RELA nome_do_par INSE <decl.parâm.> 99 RELA prox_decl FIM	
decl_ subrot_ glob	lebase	TIPO FIM	
proc_glob	procedi- mento	RELA decl_proc_glob RELA prox_decl_glob	
func_glob	função	RELA decl_func_glob RELA prox_decl_glob	
nome_const	lebase	IDEN FIM	
			5

tipo do objeto	tipo do atributo	atributos	C
bloco	lebase	RELA é_composto_por FIM	
início	início	RELA é_composto_por FIM	
subrot	subrot	RELA é_composto_por FIM	
composto	composto	SUBS { 0 INSE <comando> 1 RELA é_composto_por INSE } 0 FIM	
			6

tipo do objeto	tipo do atributo	atributos	C
tipo_de_ dado	lebase	TIPO FIM	
dado_var	var	INSE <tipo> 1 RELA é_tipo_registro RELA é_tipo_array RELA é_tipo_novo RELA é_tipo_básico FIM	
dado_parâm	parâm	INSE <tipo> RELA é_tipo_novo RELA é_tipo_básico FIM	
tipo_ básico	lebase	IDEN FIM	
parâm_da_ lista	lebase	SUBS @,@ 0 INSE <parâmetro> 1 RELA identifica_par_lis INSE <lista parâm.> RELA próx_parâm_lis FIM	
			7

tipo do objeto	tipo do atributo	atributos	C
registro	lebase	SUBS struct{ 0 INSE <declara var.> 1 RELA declara_campos INSE }@; 0 FIM	
array	lebase	ELIM RELA possui_tipo AVAN RELA nome_do_tipo CONC @ RELA possui_índice FIM	
índice	lebase	SUBS [@]@ 0 INSE <índice> 1 RELA forma_índice INSE <índices> 1 RELA prox_ind FIM	
expressão	lebase	ATRI modo FIM	
atribuição	modo	ATRI descrição FIM	
condição	modo	ATRI descrição FIM	
constante	modo	ATRI descrição FIM	
lista_expr	modo	ATRI descrição FIM	
início	modo	ATRI descrição FIM	
			8

tipo do objeto	tipo do atributo	atributos	C
parada	modo	ATRI descrição FIM	
incremento	modo	ATRI descrição FIM	
retorno	modo	ATRI descrição FIM	
expr_aritm	modo	ATRI descrição FIM	
ativa	modo	ATRI descrição FIM	
expr_ind	modo	ATRI categoria ATRI inf ATRI sup FIM	
comando	lebase	TIPO FIM	
ift	ift	SUBS if(@) 0 INSE <condição> 1 RELA verifica_cond INSE <bloco> 0 RELA executa_caso_v INSE <comando> -1 RELA segue FIM	
return	return	SUBS return(@); 0 INSE <expr.> 1 RELA retorna_expr INSE <comando> -1 RELA segue FIM	
			9

tipo do objeto	tipo do atributo	atributos	C
ifte	ifte	SUBS if(@){ 0 INSE <condição> 1 RELA verifica_cond INSE <bloco> 0 RELA executa_caso_v INSE }else 0 INSE <bloco> 1 RELA executa_caso_f INSE <comando> -2 RELA segue FIM	
goto	goto	SUBS goto @; 0 INSE <rótulo> 1 RELA desvia_para INSE <comando> -1 RELA segue FIM	
atrib	atrib	SUBS @; 0 INSE <atribuição> 1 RELA formada_por_expr INSE <comando> -1 RELA segue FIM	
cham_sub	cham_sub	SUBS @(@); 0 INSE <nome> 1 RELA nome_cham_sub INSE <lista expr.> 1 RELA possui_lis_expr INSE <comando> -2 RELA segue FIM	
			10

tipo do objeto	tipo do atributo	atributos	c
ifta	ifta	SUBS if(@==0) 0 INSE <expr.> 1 RELA verifica_expr_arit INSE goto @; 0 INSE <rótulo> 1 RELA desvia_rot1 INSE else if(@>0) -1 INSE <expr.> 1 RELA verifica_expr_arit INSE goto @; 0 INSE <rótulo> 1 RELA desvia_rot2 INSE else 0 INSE goto @; 1 INSE <rótulo> 1 RELA desvia_rot3 INSE <comando> -4 RELA segue FIM	
if1	if1	SUBS if(@) 0 INSE <condição> 1 RELA verifica_cond INSE <comando> 0 RELA possui_cmd INSE <comando> -1 RELA segue FIM	
			11

tipo do objeto	tipo do atributo	atributos	C
for	for	SUBS for(@;@;@) 0 INSE <expr.> 1 RELA expr_inicial INSE <expr.> 1 RELA expr_parada INSE <expr.> 1 RELA expr_incremento INSE <bloco> -2 RELA executa_bloco INSE <comando> -1 RELA segue FIM	
repeat	repeat	SUBS do{ 0 INSE <bloco> 1 RELA executa_blc INSE }while(!@); 0 INSE <condição> 1 RELA verifica_cond INSE <comando> -2 RELA segue FIM	
while	while	SUBS while(@) 0 INSE <condição> 1 RELA verifica_cond INSE <bloco> 0 RELA executa_caso_v INSE <comando> -1 RELA segue FIM	
do	do	SUBS do 0 INSE <bloco> 1 RELA executa_blc INSE until(@); 0 INSE <condição> 1 RELA verifica_cond INSE <comando> -2 RELA segue FIM	
			12

tipo do objeto	tipo do atributo	atributos	c
ifte_r	ifte_r	SUBS @: if(@){ 0 INSE <rótulo> 1 RELA possui_rótulo INSE <condição> 1 RELA verifica_cond INSE <bloco> 0 RELA executa_caso_v INSE }else 0 INSE <bloco> 1 RELA executa_caso_f INSE <comando> -3 RELA segue FIM	
goto_r	goto_r	SUBS @: goto @; 0 INSE <rótulo> 1 RELA possui_rótulo INSE <rótulo> 1 RELA desvia_para INSE <comando> -2 RELA segue FIM	
atrib_r	atrib_r	SUBS @: @; 0 INSE <rótulo> 1 RELA possui_rótulo INSE <atribuição> 1 RELA formada_por_expr INSE <comando> -2 RELA segue FIM	
cham_sub_r	cham_sub_r	SUBS @: @(@); 0 INSE <rótulo> 1 RELA possui_rótulo INSE <nome> 1 RELA nome_cham_sub INSE <lista expr.> 1 RELA possui_lis_expr INSE <comando> -3 RELA segue FIM	
			13

tipo do objeto	tipo do atributo	atributos	C
ifta_r	ifta_r	SUBS @: if(@==0) 0 INSE <rótulo> 1 RELA possui_rótulo INSE <expr.> 1 RELA verifica_expr_arit INSE goto @; 0 INSE <rótulo> 1 RELA desvia_rot1 INSE else if(@>0) -1 INSE <expr.> 1 RELA verifica_expr_arit INSE goto @; 0 INSE <rótulo> 1 RELA desvia_rot2 INSE else 0 INSE goto @; 1 INSE <rótulo> 1 RELA desvia_rot3 INSE <comando> -5 RELA segue FIM	
ifl_r	ifl_r	SUBS @: if(@) 0 INSE <rótulo> 1 RELA possui_rótulo INSE <condição> 1 RELA verifica_cond INSE <comando> 0 RELA possui_cmd INSE <comando> -2 RELA segue FIM	
			14

tipo do objeto	tipo do atributo	atributos	C
for_r	for_r	SUBS @: for(@;@;@) 0 INSE <rótulo> 1 RELA possui_rótulo INSE <expr.> 1 RELA expr_inicial INSE <expr.> 1 RELA expr_parada INSE <expr.> 1 RELA expr_incremento INSE <bloco> -2 RELA executa_bloco INSE <comando> -2 RELA segue FIM	
repeat_r	repeat_r	SUBS @: do{ 0 INSE <rótulo> 1 RELA possui_rótulo INSE <bloco> 1 RELA executa_blc INSE }while(!@); 0 INSE <condição> 1 RELA verifica_cond INSE <comando> -3 RELA segue FIM	
while_r	while_r	SUBS @: while(@) 0 INSE <rótulo> 1 RELA possui_rótulo INSE <condição> 1 RELA verifica_cond INSE <bloco> 0 RELA executa_caso_v INSE <comando> -2 RELA segue FIM	
do_r	do_r	SUBS @: do 0 INSE <rótulo> 1 RELA possui_rótulo INSE <bloco> 1 RELA executa_blc INSE until(@); 0 INSE <condição> 1 RELA verifica_cond INSE <comando> -3 RELA segue FIM	

APÊNDICE C

ERPFOR

tipo do objeto	tipo do atributo	atributos	FOR
arquivo	lebase	RELA contém_prog FIM	
subrotina	lebase	TIPO FIM	
principal	principal	INSE program @; 0 INSE <nome> 1 RELA nome_da_subrot INSE <declara var.> 0 RELA decl_var_glob INSE <define const.> 0 RELA define_const INSE <bloco> 1 RELA contém_blc_inic INSE <decl.subrot.> -1 RELA decl_global FIM	
			1

tipo do objeto	tipo do atributo	atributos	FOR
procedi-mento	procedi-mento	SUBS subroutine @(@) 0 INSE <nome> 1 RELA nome_da_subrot INSE <lista parâm> 1 RELA tem_lista_parâm INSE <decl.parâm.> -1 RELA declara_parâmetros INSE <declara var.> 0 RELA decl_var_loc INSE <define const.> 0 RELA define_const INSE <bloco> 0 RELA proc_contém_blc INSE <decl.subrot.> -1 RELA prox_decl_glob FIM	
			2

tipo do objeto	tipo do atributo	atributos	FOR
função	função	SUBS @ function @(@) 0 RELA retorna_tipo INSE <nome> 1 RELA nome_da_subrot INSE <lista parâm.> 1 RELA tem_lista_parâm INSE <decl.parâm.> -1 RELA declara_parâmetros INSE <declara var.> 0 RELA decl_var_loc INSE <define const.> 0 RELA define_const INSE <bloco> 0 RELA func_contém_blc INSE <decl.subrot.> -1 RELA prox_decl_glob FIM	
			3

tipo do objeto	tipo do atributo	atributos	FOR
nome_ subrot	lebase	IDEN FIM	
def_const	lebase	SUBS parameter (@=@) 0 INSE <nome> 1 RELA nome_da_const INSE <constante> 1 RELA forma_const INSE <define const.>-2 RELA prox_def_const FIM	
nome_ parâmetro	lebase	IDEN FIM	
nome_const	lebase	IDEN FIM	
nome_var	lebase	IDEN FIM	
rótulo	lebase	IDEN FIM	
			4

tipo do objeto	tipo do atributo	atributos	FOR
declaração	lebase	TIPO FIM	
variável	var	SUBS @ @ 0 RELA tem_tipo_de_dado INSE <nome> 100 RELA nome_da_var INSE <declara var.> 99 RELA prox_decl FIM	
parâmetro	parâm	SUBS @ @ 0 RELA tem_tipo INSE <nome> 100 RELA nome_do_par INSE <decl.parâm.> 99 RELA prox_decl FIM	
decl_ subrot_ gllob	lebase	TIPO FIM	
proc_glob	procedi-mento	RELA decl_proc_glob RELA prox_decl_glob	
func_glob	função	RELA decl_func_glob RELA prox_decl_glob	
			5

tipo do objeto	tipo do atributo	atributos	FOR
bloco	lebase	TIPO FIM	
início	início	SUBS <comando> 0 RELA é_composto_por INSE stop 0 INSE end 0 FIM	
subrot	subrot	SUBS <comando> 0 RELA é_composto_por INSE return 0 INSE end 0 FIM	
composto	composto	SUBS <comando> 0 RELA é_composto_por FIM	
			6

tipo do objeto	tipo do atributo	atributos	FOR
tipo_de_ dado	lebase	TIPO FIM	
dado_var	var	INSE <tipo> 1 RELA é_tipo_array RELA é_tipo_charac RELA é_tipo_básico FIM	
dado_parâm	parâm	INSE <tipo> RELA é_tipo_básico FIM	
tipo_ básico	lebase	IDEN FIM	
charac	lebase	SUBS character*@ 0 INSE tam 1 ATRI tam 0 FIM	
			7

tipo do objeto	tipo do atributo	atributos	FOR
array	lebase	ELIM RELA possui_tipo AVAN RELA nome_da_var CONC (@) INSE <índices> 1 RELA possui_índice FIM	
índice	lebase	SUBS @, @ 0 INSE <índice> 1 RELA forma_índice INSE <índices> 1 RELA prox_ind FIM	
expressão	lebase	ATRI modo FIM	
atribuição	modo	ATRI descrição FIM	
condição	modo	ATRI descrição FIM	
constante	modo	ATRI descrição FIM	
lista_expr	modo	ATRI descrição FIM	
tam	modo	ATRI descrição FIM	
início	modo	ATRI descrição FIM	
			8

tipo do objeto	tipo do atributo	atributos	FOR
parada	modo	ATRI descrição FIM	
incremento	modo	ATRI descrição FIM	
retorno	modo	ATRI descrição FIM	
expr_aritm	modo	ATRI descrição FIM	
ativa	modo	ATRI descrição FIM	
gotoc	modo	ATRI descrição FIM	
expr_ind	modo	ATRI categoria ATRI inf ATRI sup FIM	
comando	lebase	TIPO FIM	
ift	ift	SUBS if(@) 0 INSE <condição> 1 RELA verifica_cond INSE $then 0 INSE <bloco> 1 RELA executa_caso_v INSE endif -1 INSE <comando> -1 RELA segue FIM	
			9

tipo do objeto	tipo do atributo	atributos	FOR
ifte	ifte	SUBS if(@) 0 INSE <condição> 1 RELA verifica_cond INSE $then 0 INSE <bloco> 1 RELA executa_caso_v INSE else -1 INSE <bloco> 1 RELA executa_caso_f INSE endif -1 INSE <comando> -2 RELA segue FIM	
goto	goto	SUBS goto @ 0 INSE <rótulo> 1 RELA desvia_para INSE <comando> -1 RELA segue FIM	
atrib	atrib	SUBS @ 0 INSE <atribuição> 1 RELA formada_por_expr INSE <comando> -1 RELA segue FIM	
cham_sub	cham_sub	SUBS call @(@) 0 INSE <nome> 1 RELA nome_cham_sub INSE <lista expr.> 1 RELA possui_lis_expr INSE <comando> -2 RELA segue FIM	
			10

tipo do objeto	tipo do atributo	atributos	FOR
for	for	SUBS do @,@,@,@ 0 INSE <rótulo> 1 RELA for_desvia_para INSE <expr.> 1 RELA expr_inicial INSE <expr.> 1 RELA expr_parada INSE <expr.> 1 RELA expr_incremento INSE <bloco> -3 RELA executa_bloco INSE @ continue 0 INSE <rótulo> 1 RELA rótulo_do_for INSE <comando> -2 RELA segue FIM	
repeat	repeat	SUBS @ continue 0 INSE <rótulo> 1 RELA rótulo_do_repeat INSE <bloco> 1 RELA executa_blc INSE if(.NOT. @)goto @ 0 INSE <condição> 1 RELA verifica_cond INSE <rótulo> 1 RELA repeat_desvia_para INSE endif -2 INSE <comando> -2 RELA segue FIM	
gotoc	gotoc	SUBS goto (@)@ 0 INSE <rótulos> 1 RELA rótulos_de_desvio INSE <expr.> 1 RELA avalia_expr INSE <comando> -2 RELA segue FIM	
			11

tipo do objeto	tipo do atributo	atributos	FOR
while	while	SUBS @ if(.NOT.)goto@ 0 INSE <rótulo> 1 RELA contém_rót1 INSE <condição> 1 RELA verifica_cond INSE <rótulo> 1 RELA desvia_para_rot1 INSE <bloco> -1 RELA executa_caso_v INSE goto @ 0 INSE <rótulo> 1 RELA desvia_para_rot2 INSE @ continue -1 INSE <rotulo> 1 RELA contém_rot2 INSE endif -1 INSE <comando> -2 RELA segue FIM	
do	do	SUBS @ continue 0 INSE <rótulo> 1 RELA rótulo_do_do INSE <bloco> 0 RELA executa_blc INSE if(@) goto @ 0 INSE <condição> 1 RELA verifica_cond INSE <rótulo> 1 RELA do_desvia_para INSE endif -2 INSE <comando> -1 RELA segue FIM	
rótulos	lebase	SUBS @,@ 0 INSE <rótulo> 1 RELA contém_rot INSE <rótulos> 1 RELA prox_rot FIM	

tipo do objeto	tipo do atributo	atributos	FOR
if1	if1	SUBS if(@)@ 0 INSE <condição> 1 RELA verifica_cond INSE <comando> 1 RELA cmd_do_if1 INSE <comando> -2 RELA segue FIM	
ifa	ifa	SUBS if(@)@,@,@ 0 INSE <expr.> 1 RELA verifica_expr_arit INSE <rótulo> 1 RELA desvia_rót1 INSE <rótulo> 1 RELA desvia_rót2 INSE <rótulo> 1 RELA desvia_rót3 INSE <comando> -4 RELA segue FIM	
			13

tipo do objeto	tipo do atributo	atributos	FOR
ifte_r	ifte_r	SUBS @ if(@) 0 INSE <rótulo> 1 RELA possui_rótulo INSE <condição> 1 RELA verifica_cond INSE $then 0 INSE <bloco> 1 RELA executa_caso_v INSE else -1 INSE <bloco> 1 RELA executa_caso_f INSE endif -1 INSE <comando> -2 RELA segue FIM	
goto_r	goto_r	SUBS @ goto @ 0 INSE <rótulo> 1 RELA possui_rótulo INSE <rótulo> 1 RELA desvia_para INSE <comando> -2 RELA segue FIM	
atrib_r	atrib_r	SUBS @ @ 0 INSE <atribuição> 1 INSE <rótulo> 1 RELA possui_rótulo RELA formada_por_expr INSE <comando> -2 RELA segue FIM	
cham_sub_r	cham_sub_r	SUBS @ call @(@) 0 INSE <rótulo> 1 RELA possui_rótulo INSE <nome> 1 RELA nome_cham_sub INSE <lista expr.> 1 RELA possui_lis_expr INSE <comando> -3 RELA segue FIM	
			14

tipo do objeto	tipo do atributo	atributos	FOR
for_r	for_r	SUBS @ do @,@,@,@ 0 INSE <rótulo> 1 RELA possui_rótulo INSE <rótulo> 1 RELA for_desvia_para INSE <expr.> 1 RELA expr_inicial INSE <expr.> 1 RELA expr_parada INSE <expr.> 1 RELA expr_incremento INSE <bloco> -3 RELA executa_bloco INSE @ continue 0 INSE <rótulo> 1 RELA rótulo_do_for INSE <comando> -3 RELA segue FIM	
repeat_r	repeat_r	SUBS @ continue 0 INSE <rótulo> 1 RELA possui_rótulo INSE @ continue 0 INSE <rótulo> 1 RELA rótulo_do_repeat INSE <bloco> 1 RELA executa_blc INSE if(.NOT. @)goto @ 0 INSE <condição> 1 RELA verifica_cond INSE <rótulo> 1 RELA repeat_desvia_para INSE endif -2 INSE <comando> -3 RELA segue FIM	
gotoc_c	gotoc_r	SUBS @ goto (@)@ 0 INSE <rótulo> 1 RELA possui_rot INSE <rótulos> 1 RELA rótulos_de_desvio INSE <expr.> 1 RELA avalia_expr INSE <comando> -3 RELA segue FIM	
			15

tipo do objeto	tipo do atributo	atributos	FOR
while_r	while_r	SUBS @ continue 0 INSE <rótulo> 1 RELA possui_rótulo INSE @ if(.NOT.)goto@ 0 INSE <rótulo> 1 RELA contém_rót1 INSE <condição> 1 RELA verifica_cond INSE <rótulo> 1 RELA desvia_para_rot1 INSE <bloco> -1 RELA executa_caso_v INSE goto @ 0 INSE <rótulo> 1 RELA desvia_para_rot2 INSE @ continue -1 INSE <rotulo> 1 RELA contém_rot2 INSE endif -1 INSE <comando> -3 RELA segue FIM	
do_r	do_r	SUBS @ continue 0 INSE <rótulo> 1 INSE @ continue 0 INSE <rótulo> 1 RELA rótulo_do_do INSE <bloco> 0 RELA executa_blc INSE if(@) goto @ 0 INSE <condição> 1 RELA verifica_cond INSE <rótulo> 1 RELA do_desvia_para INSE endif -2 INSE <comando> -2 RELA segue FIM	
			16

tipo do objeto	tipo do atributo	atributos	FOR
ifl_r	ifl_r	SUBS @ if(@)@ 0 INSE <rótulo> 1 RELA possui_rótulo INSE <condição> 1 RELA verifica_cond INSE <comando> 1 RELA cmd_do_ifl INSE <comando> -3 RELA segue FIM	
ifa_r	ifa_l	SUBS @ if(@)@,@,@ 0 INSE <rótulo> 1 RELA possui_rótulo INSE <expr.> 1 RELA verifica_expr_arit INSE <rótulo> 1 RELA desvia_rót1 INSE <rótulo> 1 RELA desvia_rót2 INSE <rótulo> 1 RELA desvia_rót3 INSE <comando> -4 RELA segue FIM	
			17