Tabela de Símbolos

Introdução

Fonte: http://www.icmc.sc.usp.br/~gracan/download/sce126/

Tabela de Símbolos

Um compilador necessita coletar e usar informações sobre os nomes (identificadores) que aparecem no programa-fonte. Estas informações são colocadas numa estrutura de dados chamada Tabela de Símbolos (T.S.). As informações coletadas sobre o nome incluem : a cadeia de caracteres pelo qual é denotado, isto é, o nome propriamente dito; seu tipo - inteiro, real, booleano etc; sua forma - variável simples, subscrita; seu tamanho; sua localização na memória; e outros atributos, dependendo da linguagem.

Cada entrada na T.S. é composta por nome e informação. Cada vez que um nome é encontrado, a T.S. é pesquisada para ver se aquele nome já havia aparecido antes. Se o nome é novo, ele é introduzido na tabela. Informações sobre este nome são introduzidas na tabela durante as análises léxica e sintática.

Tabela de Símbolos

As informações contidas na T.S. são usadas durante diversas fases no processo de compilação. São usadas na análise semântica, isto é, na verificação se o uso do nome é consistente com sua declaração implícita ou explícita. Por exemplo, numa expressão aritmética ou lógica, as variáveis devem ser compatíveis entre si. Também na fase de geração de código, usamos tais informações para saber quanto e que tipo de memória em tempo de execução deve se alocada para determinado identificador. Durante a otimização de código, o espaço na T.S. pode ser usado para as variáveis temporárias, por exemplo, que são usadas mais que uma vez. Praticamente, são 5 os tipos de acesso à T.S.:

1- determinar se um dado nome está na tabela,
2- adicionar um novo nome à tabela,
3- acessar as informações associadas com um dado nome,
4- adicionar novas informações para um dado nome,
5- deletar um nome ou um grupo de nomes da tabela.

Tabela de Símbolos

Num compilador, os nomes na T.S. denotam objetos de diversos tipos. Pode haver tabelas separadas para nomes de variáveis, "labels", nomes de subrotinas, nomes de constantes, nomes de "arrays" e outros tipos de nomes dependendo da linguagem. É comum se ter mais do que uma tabela devido o espaço requerido por cada nome poder variar consideravelmente, dependendo do uso que se faz do nome. Entretanto, se o formato das entradas de informação puder variar, uma única tabela pode bastar.

Dependendo de como a análise léxica é implementada, pode ser útil inicializar a T.S. com as palavras reservadas. Se a linguagem não reserva palavras-reservadas (permite o uso de tais palavras como identificadores), então é essencial que as palavras reservadas sejam introduzidas na T.S. e que elas tenham associadas a si uma informação de que podem ser usadas como palavras-reservadas.

Tabela de Símbolos

Uma vez que a T.S. será consultada toda vez que a rotina de reconhecimento de identificadores for acionada, ela deverá ser organizada de modo que a recuperação das informações seja rápida e eficiente. Uma das técnicas de procura mais usadas para a T.S. é a que usa as "funções de espalhamento" ("hash").

Tabela de Símbolos

Características da tabela de símbolos:

- Pode ser grande;
- Dinâmica;
- Contém nomes e informações sobre variáveis, funções, tipos, etc;
- Inicialmente vazia, é *construída* pelo Parser e Gerador de Código;
- É *consultada* pelo Parser, por ações semânticas e pelo Gerador de Código.

Tabela de Símbolos

Descritores

Os descritores são os registros que formam a T.S. de um compilador. São neles que armazenaremos as informações correspondentes a cada identificador. Para cada tipo de identificador teremos uma estrutura de descritores diferentes, por exemplo, para o descritor de um identificador de variável simples teremos:

- tipo de variável (inteira, real, complexa, booleana, etc);
- endereço na memória;
- nome da variável;
- contexto em que foi definida (programa principal, subrotina etc);
- campo de informação, etc.

Para o descritor de um vetor deveríamos ter ainda o tamanho do vetor, o valor de seus limites, etc.

Tabela de Símbolos

Estruturas de Dados

Fonte: http://www.icmc.sc.usp.br/~gracan/download/sce126/

Tabela de Símbolos

Operações sobre T. S.

- verificar se um dado nome está na T. S.
- inserir um nome na T. S.
- acessar a info. associada a um nome
- adicionar info. associada a nome
- eliminar 1 ou grupo de nomes.

Entrada da T.S.

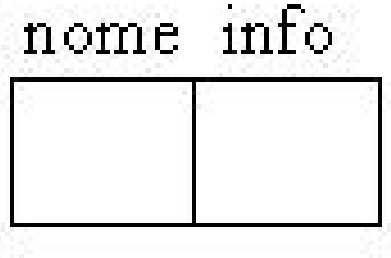


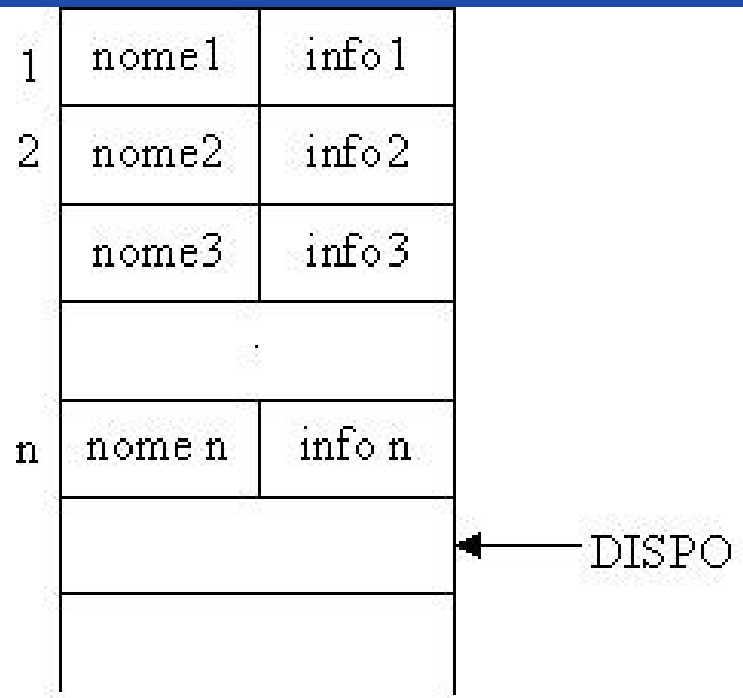
Tabela de Símbolos

Estruturas de Dados para a T.S.

1) Listas Lineares Seqüenciais:

- simples; fácil de implementar
- custo p/ inserção: ~n
- busca média: n/2
- custo p/ n inserções e m buscas: $cn(n+m)=O(n^2)$

Tabela de Símbolos



Listas encadeadas

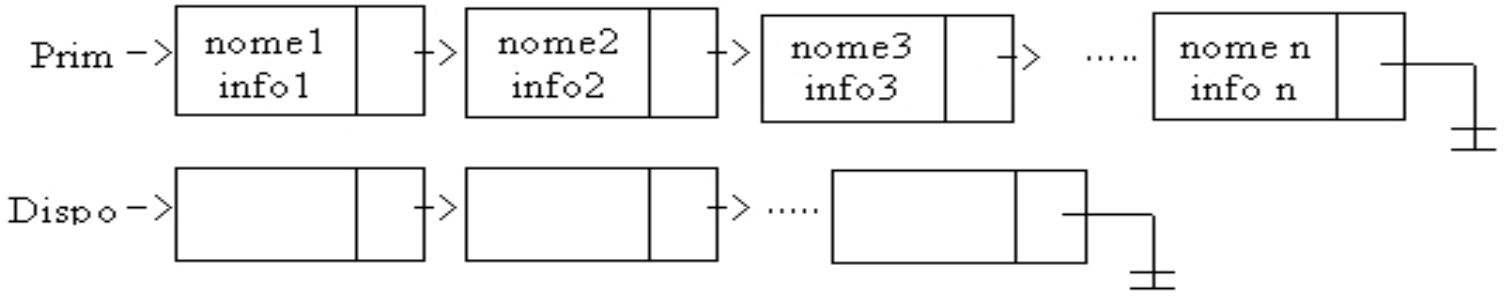


Tabela de Símbolos

2) Árvores de Busca:

" while P (diferente) nil do
if P.nome = nome then...
else if p.nome < nome then P: = P.esq
else P: = P.Dir "

Nomes em ordem aleatória:

Comprimento médio de busca: ~ log(n)

Custo p/ n inserções e m buscas: ~ (n + m) log (n)

n > 50 ====>árvore busca melhor listas.

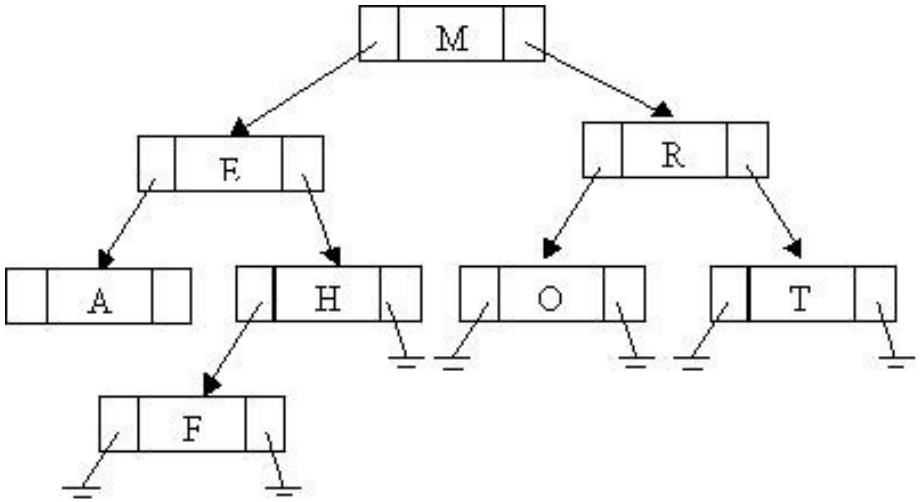


Tabela de Símbolos

3) Tabelas Hash:

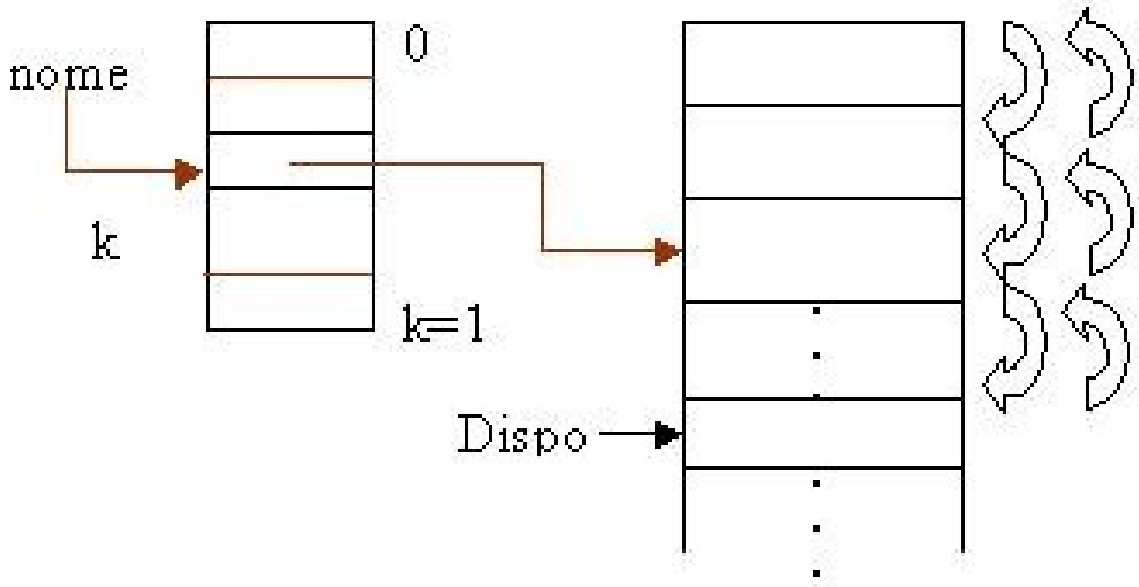
As funções "hash", E(x), são na verdade relações que levam, de uma certa característica da chave em que se aplica a relação, a um endereço válido para cada caso específico. Em particular, numa T.S. que comporta M identificadores, devemos construir uma relação de modo que, aplicando-a a um dado identificador obteremos o endereço onde deveriam estar as informações referentes a ele.
Atenção: o endereço obtido deve sempre estar dentro da T.S.

Id ==========> 0<E<=M
E(x)

onde: Id = conjunto de todos os identificadores possíveis de serem formados de acordo com as regras impostas.
E = conjunto de endereços válidos para a T.S. 0<E<=M

Tabela de Símbolos

Obviamente, o número de identificadores possíveis é muito maior do que o número de posições na T.S., e isto significa que haverá casos de conflitos; isto é, mais do que um identificador produzirá o mesmo endereço na T.S. A eficiência do método de funções de "hash" está intimamente relacionada com a solução adotada para resolver os casos de conflito.



busca média:

n/k

Tabela de Símbolos

h: nome ==> o .. k-1

Requisitos sobe h: - distribuição uniforme entre as k listas
- computação simples

Em geral : k = nº primo
h (nome) = (nome como um inteiro) mod k

Seja:

h1(nome) = (nome como inteiro) mod (k-2) + 1
então h1: nome ==> o .. k-2

No caso de colisão entre h, fazemos
h = (h + i h1) mod k; i = 1, 2, ...

Tabela de Símbolos

Hashing na T.S.R.

Na fase de construção da T.S.R. (que é fixa), armazenamos o nº de colisões sofridas de cada chave para ser inserida.

Ao verificar se uma cadeia é palavra ou símbolo reservado, saberemos quantas tentativas devemos fazer, no máximo, para decidir que a cadeia não pertence à T.S.R.

Tamanho da T.S.R. = nº primo >= m + 65%m

onde m = nº de símbolos e palavras reservadas

Tabela de Símbolos

T.S. para LALG

Fonte: http://www.icmc.sc.usp.br/~gracan/download/sce126/

Tabela de Símbolos

Tabela de Símbolos

Escopo de procab

	cadeia	token	categoria	tipo	Valor
1.	'a'	ident	var	integer	-
2.	'3'	num	-	integer	3
3.	'procab'	ident	proc	-	-
4.	'x'	ident	parâmetro	real	-
5.	'y'	ident	var	real	-
6.	'4'	num	-	integer	4

Tabela de Símbolos

Acessos à T.S.:

- busca(s) - retorna o índice da TS onde está a cadeia s
- insere(s,t) - retorna índice da nova entrada na TS para s, token t.

Fase de Inserção: declarações locais e globais
Busca: comandos
Eliminação: término de procedimento

Tabela de Símbolos

Esquemas das operações sobre a T.S. nos Grafos Sintáticos da LALG

dc

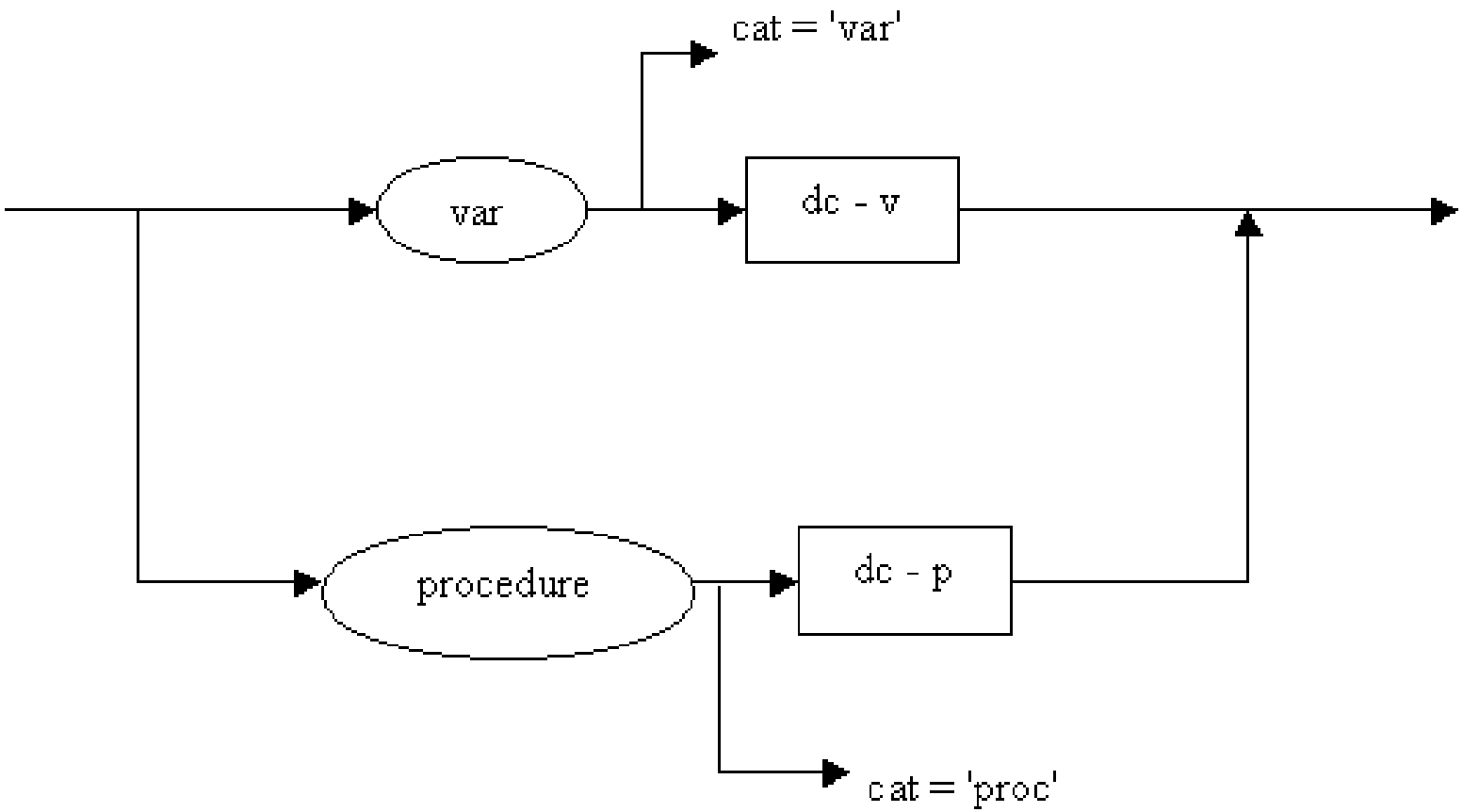


Tabela de Símbolos

variáveis

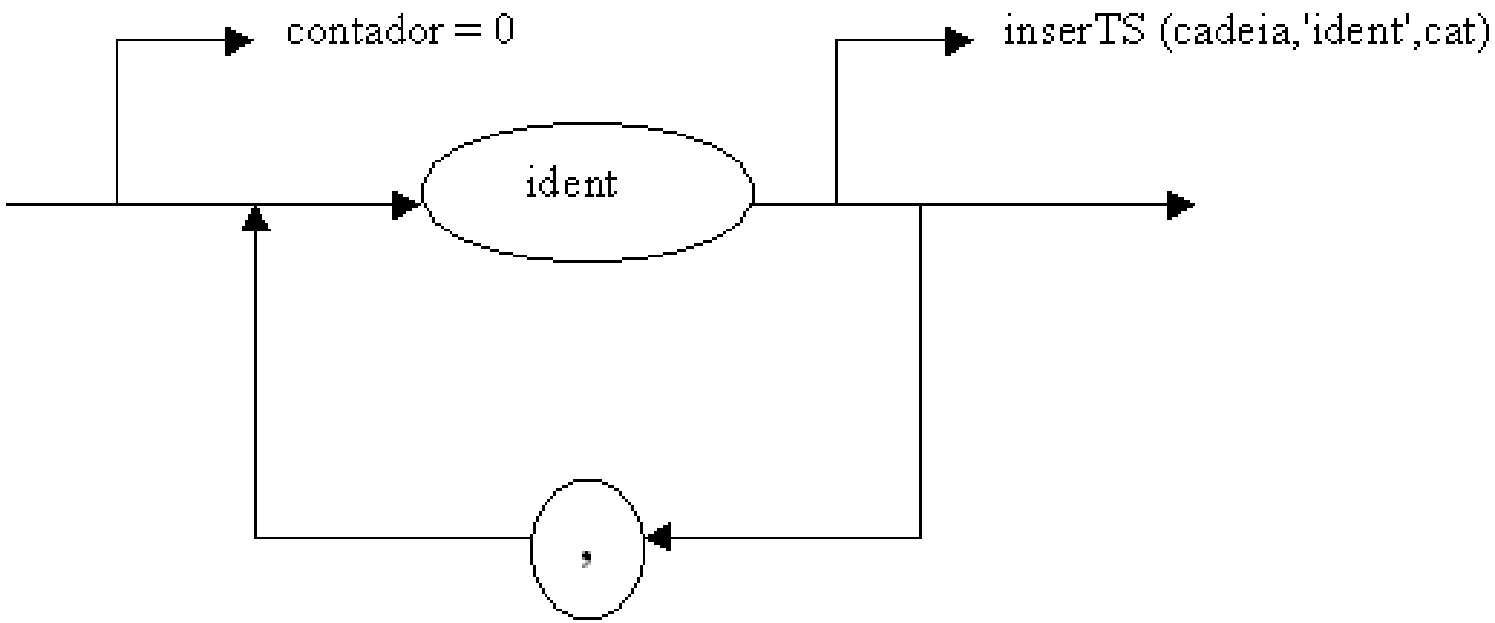


Tabela de Símbolos

dc_p

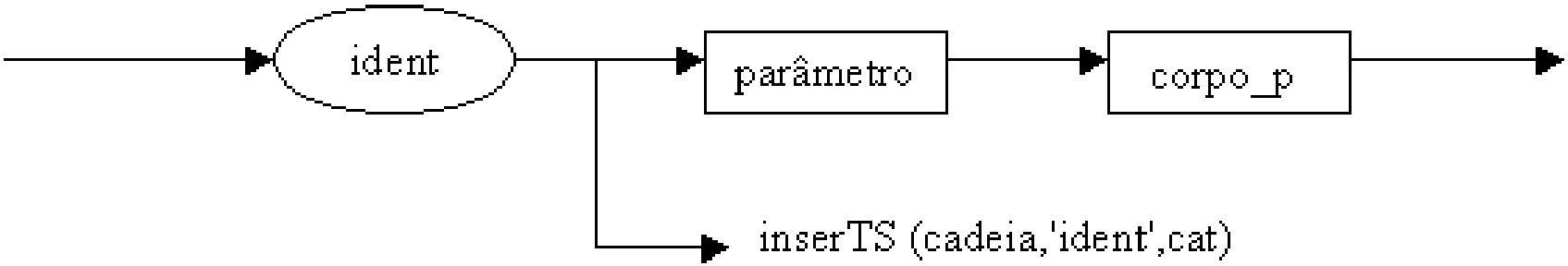


Tabela de Símbolos

parâmetros

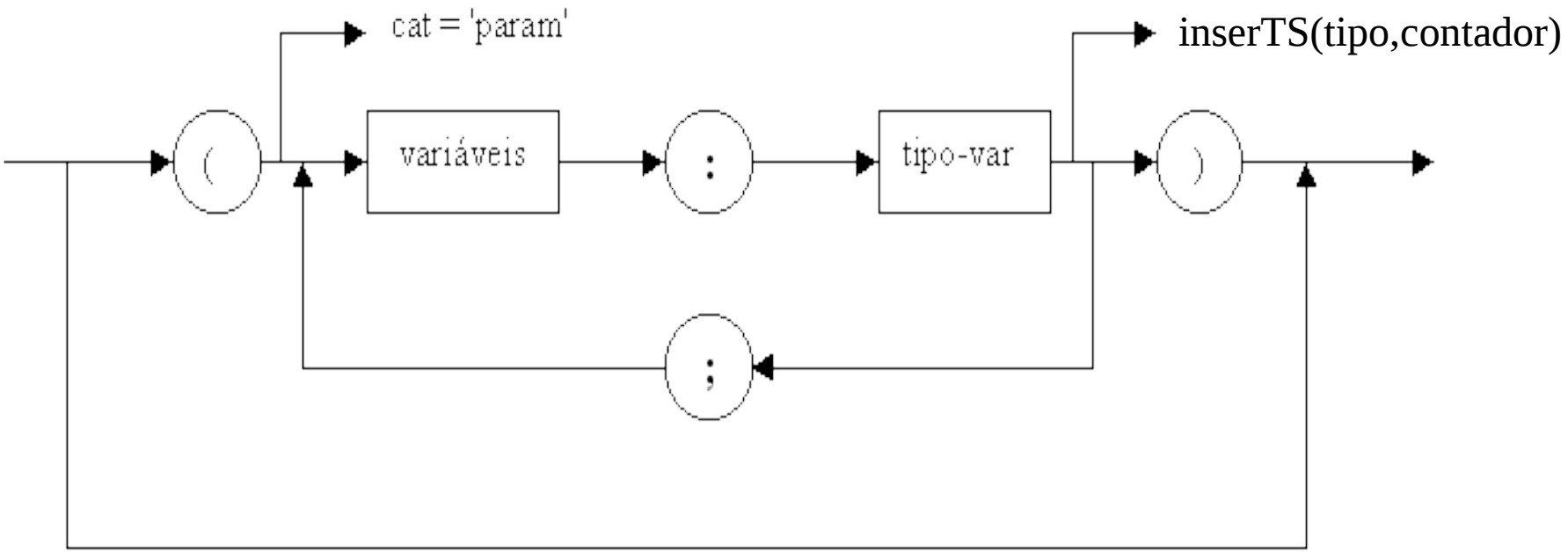


Tabela de Símbolos

fator

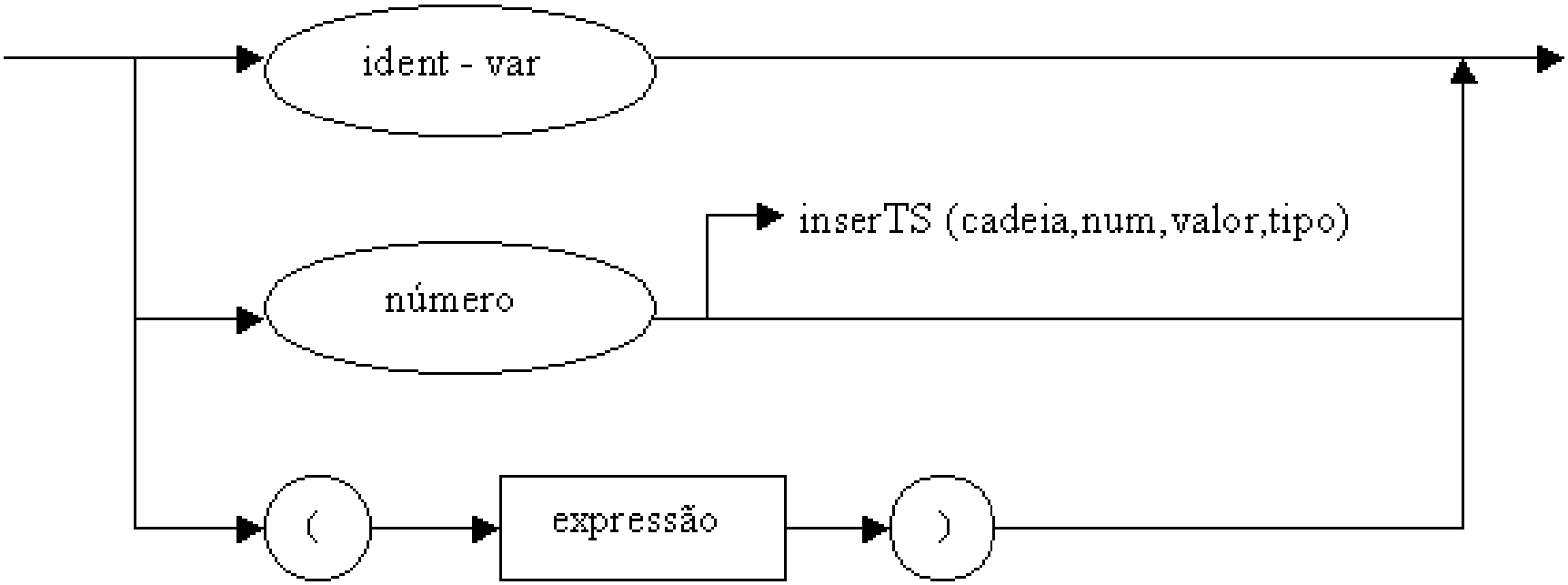


Tabela de Símbolos

corpo_p

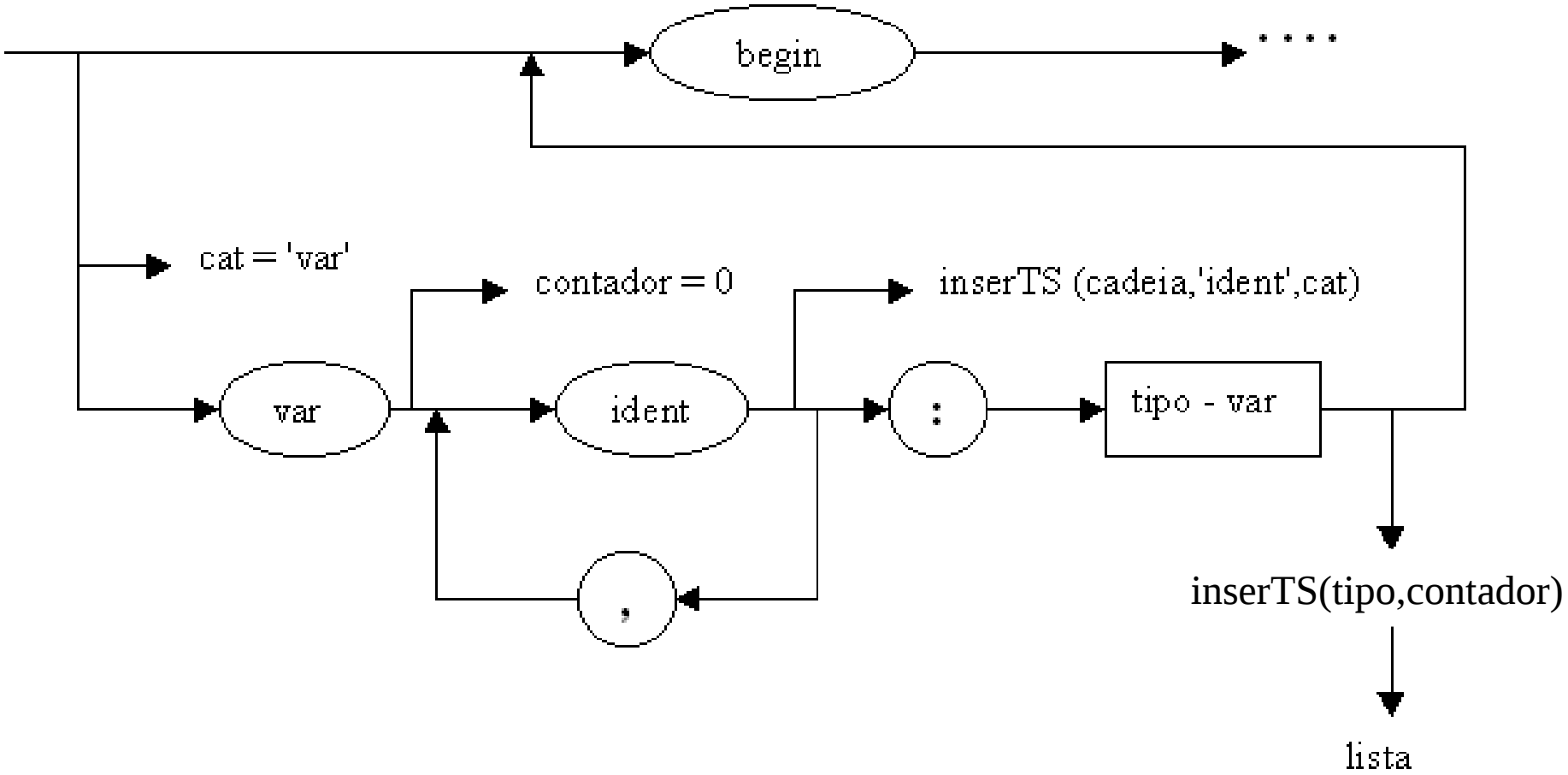
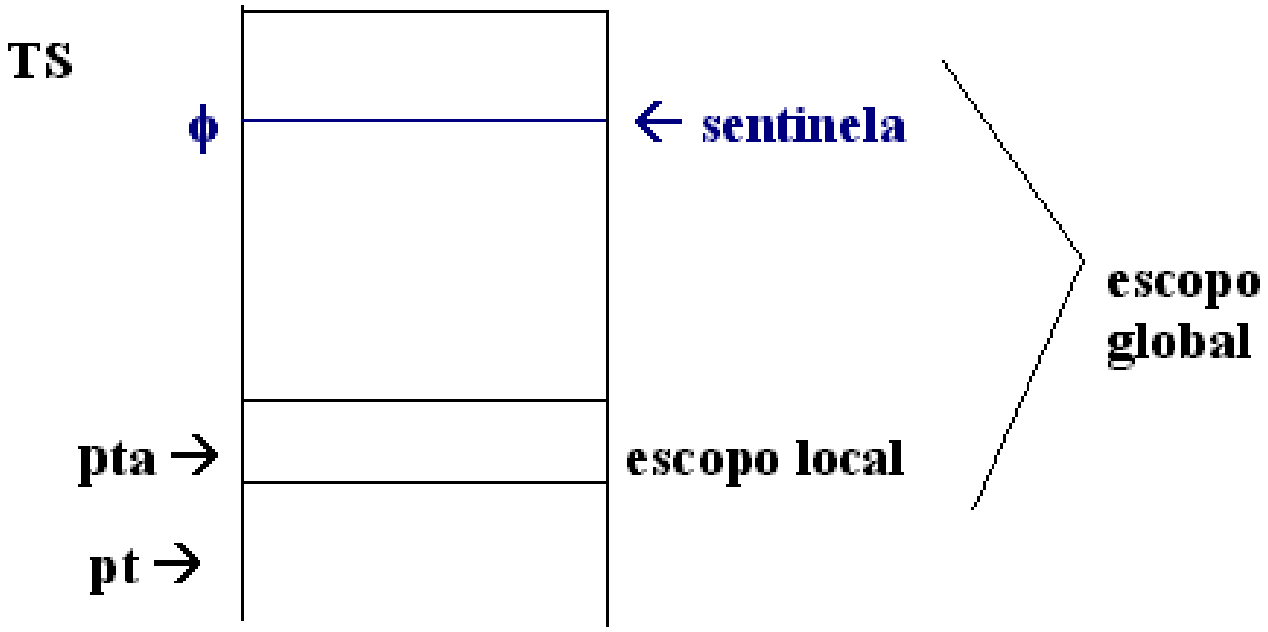


Tabela de Símbolos

ATENÇÃO: sempre que for inserir na TS, deve-se verificar se já não existe um identificador com o mesmo nome naquele escopo.

Dicas:

1. Delimitar o escopo por 2 ponteiros



2. e/ou usar um atributo de escopo para cada identificador na TS

Tabela de Símbolos

Na declaração global:

inserção com busca entre pt e pta = 0
→ se encontrar: declaração repetida
senão, inserir em pt

Para inicializar escopo local:

em <dc_p>

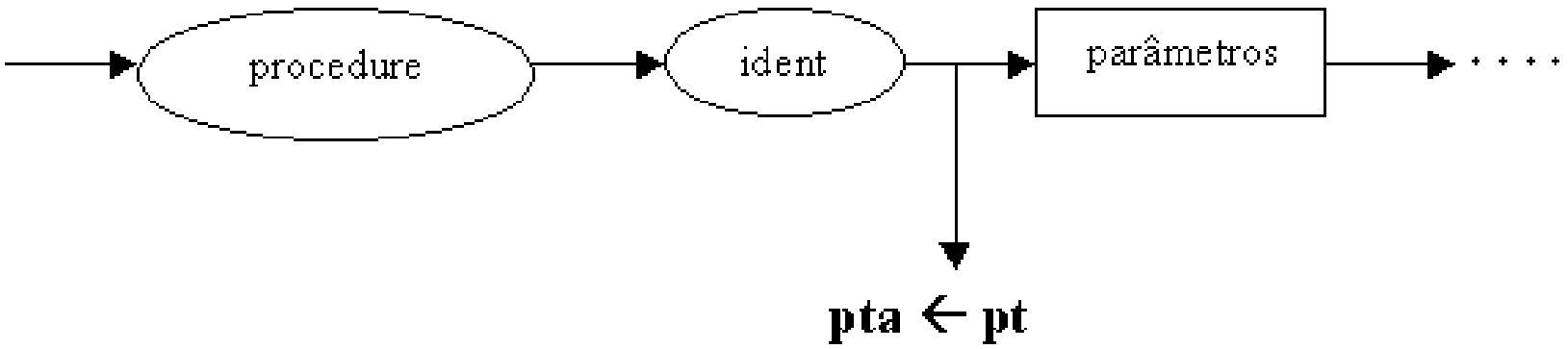


Tabela de Símbolos

Declaração local:

em <dc_loc>

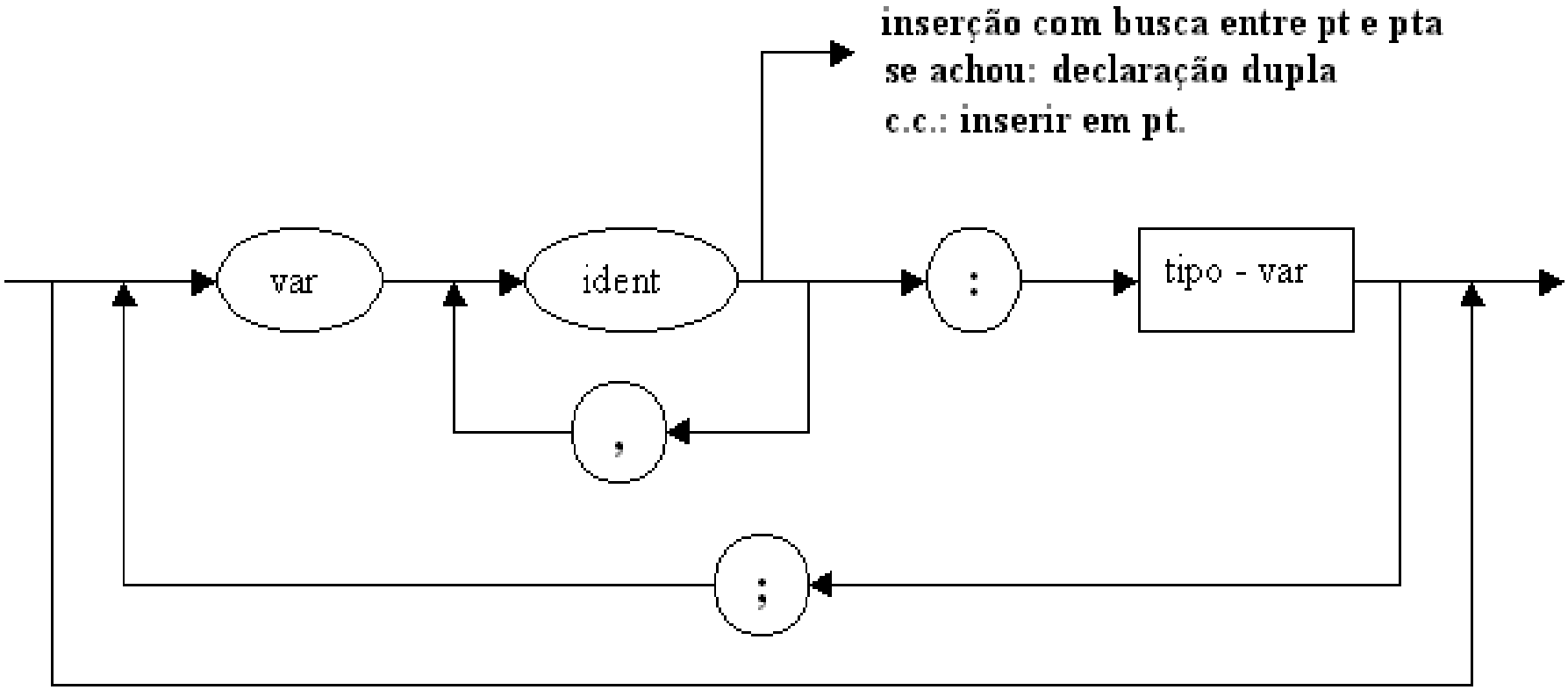


Tabela de Símbolos

Busca na fase de Comandos:

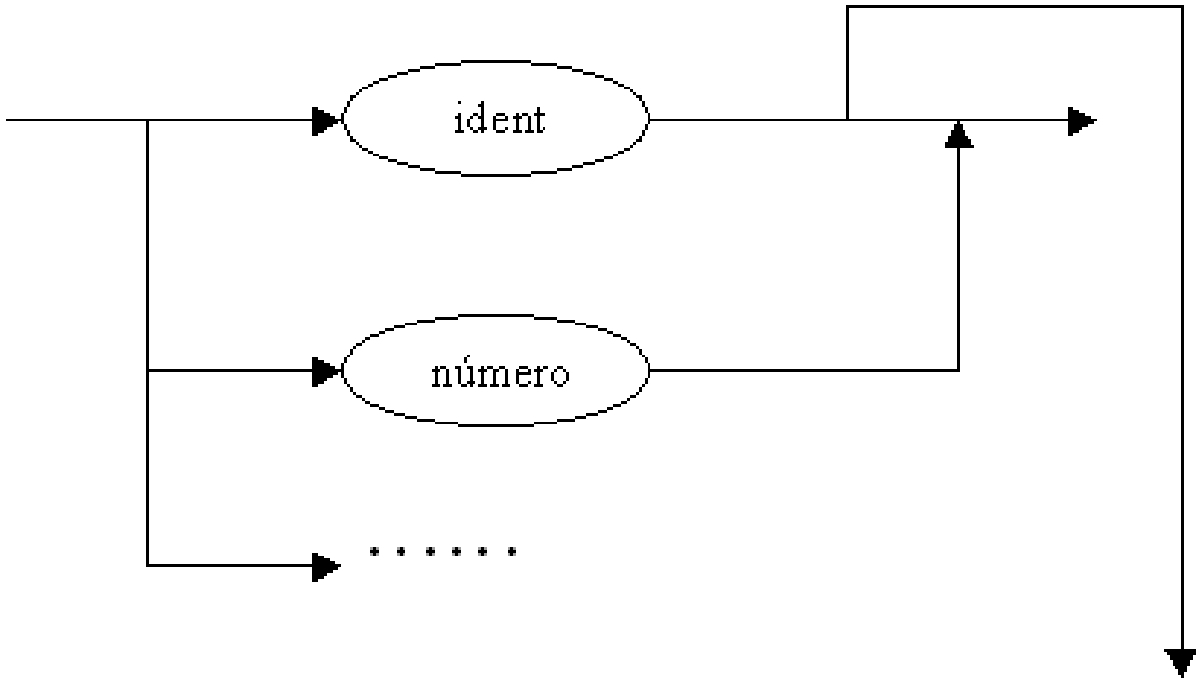
Verificar se identificador foi declarado, seu tipo, etc.

Escopo Global: busca de pt a pta = 0
Escopo Local: idem, considerando a 1ª ocorrência (escopo menor)

Quando se faz a busca?
Sempre que aparecer um identificador na área de comandos.
Portanto em comando e fator.

Tabela de Símbolos

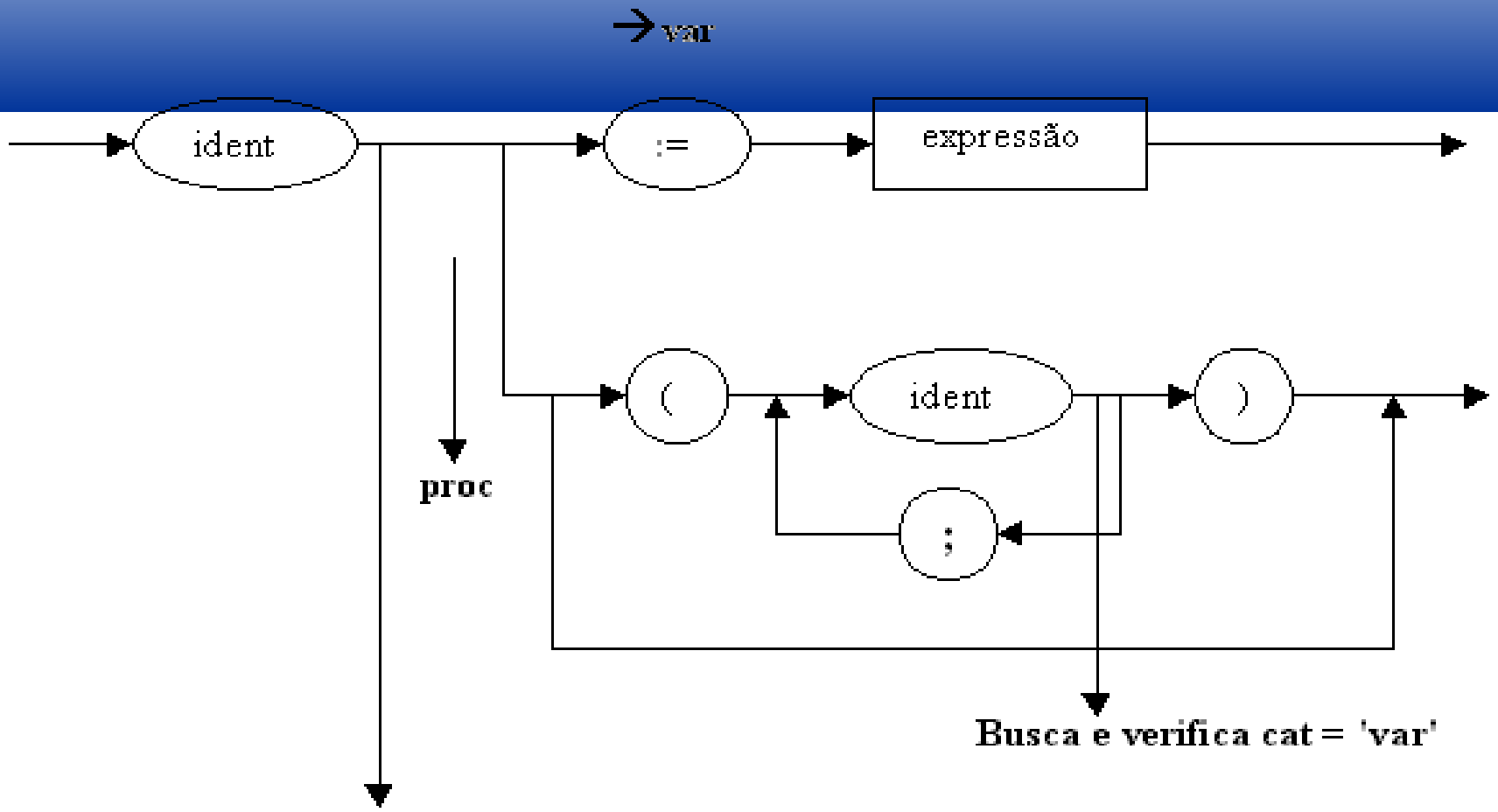
fator:



BUSCA - se não achou: ident. não declarado
se achou: verificar se categoria é 'var'

Tabela de Símbolos

em comando:



Busca de pt a pta = 0:

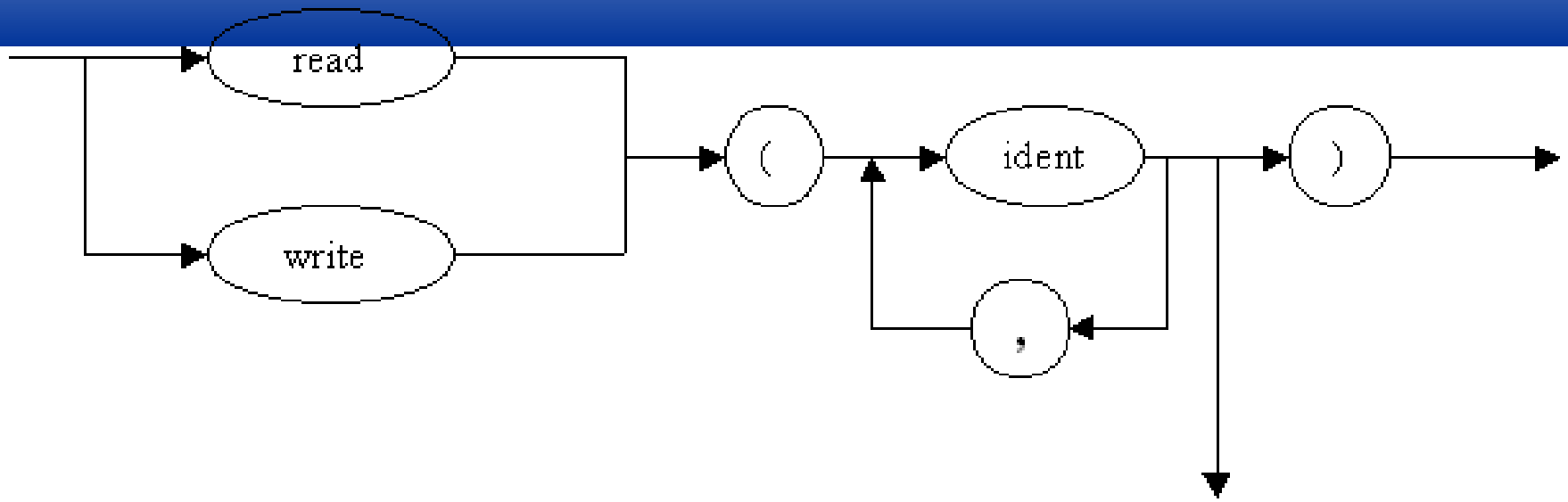
se não achou: ident. não declarado

se achou, verifica categoria:

se 'var' . . . . :=

se 'proc' . . . . (

Tabela de Símbolos



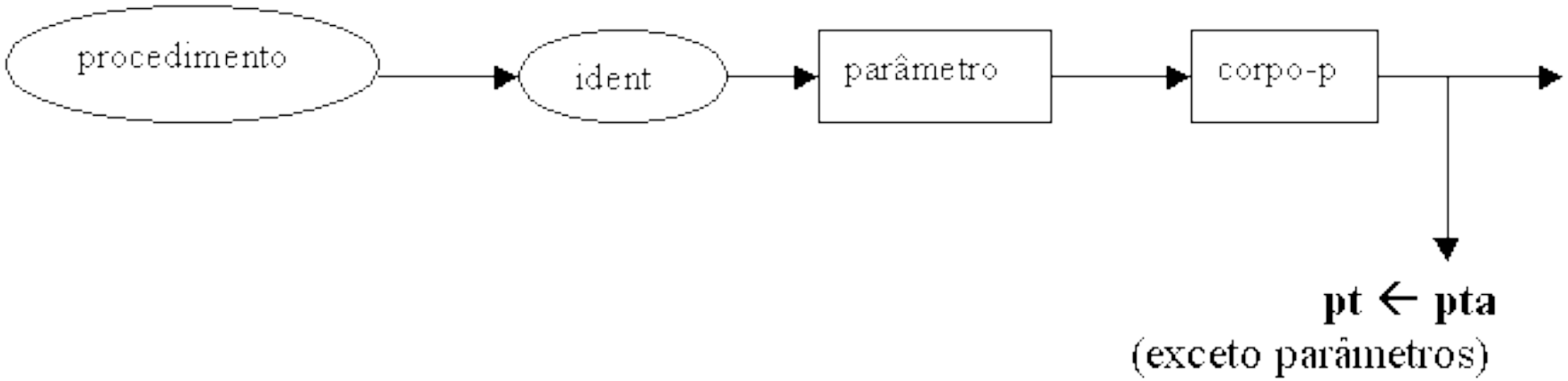
Busca e verifica se categoria é 'var'

Tabela de Símbolos

Eliminação na TS:

no final do procedimento, elimina símbolos locais, mantendo os parâmetros.

em dc_p:



A categoria 'param' é equivalente a 'var', uma vez que LALG não admite passagem de procedimentos como argumentos.