CAP. VI – ANÁLISE SEMÂNTICA

VI.1 – Introdução

- **Semântica**
  - **SIGNIFICADO, SENTIDO LÓGICO, COERÊNCIA, ...**
  - **Diferença entre SINTAXE e SEMÂNTICA**
    - **Sintaxe : descreve as estruturas de uma linguagem;**
    - **Semântica : descreve o significado dessas estruturas**
  - **Objetivo:**
    - **Estabelecer o significado de um programa a partir de sua estrutura sintática.**
  - **Tipos de semântica**
    - **Semântica estática: Estabelece as restrições que um programa sintaticamente correto deve satisfazer para que seja possível estabelecer um significado para ele.**
    - **Semântica dinâmica: Estabelece o significado de um programa; ou seja, seu comportamento e os resultados da execução desse programa.**

VI.2 – Especificação semântica de Linguagens

- **Consiste na definição de um conjunto de REGRAS SEMÂNTICAS que estabelecem o significado ou o comportamento dos programas.**

- **Regras Semânticas:**

  - **Estabelecem restrições / condições relativas ao uso dos elementos sintáticos da linguagem.**

  - **Devem ser satisfeitas para que o programa tenha sentido – seja passível de execução.**

  - **Visam o estabelecimento de um significado para o programa fonte em termos de um programa objeto.**

- **Regras de Semântica Estáticas X Dinâmicas**

  As regras semânticas são classificadas em Estáticas e Dinâmicas, dependendo do momento em que são verificadas (tempo de compilação ou tempo de execução)

  Exemplos:

  - **Regras verificadas Estaticamente:**

    - **Todo Identificador deve ser declarado antes de ser usado**

    - **Em um comando de atribuição o tipo da expr deve ser compatível com o tipo da variável do lado esquerdo**

- **Regras verificadas Dinamicamente**
  - **Índice inválido**
  - **Divisão por zero**

- **Especificação Formal x Informal da Semântica**
  - **A semântica das linguagens de programação deve(ria) ser descrita usando métodos formais (tal como se faz para os aspectos léxicos e sintáticos)**
  - **Ao contrário do que ocorre na sintaxe não existe um método universalmente aceito para descrição formal da semântica**
  - **Vantagens da Especificação Formal**
    - Definições claras e não ambíguas
    - Padronização de linguagens de programação;
    - Referência para implementadores e usuários;
    - Demonstração da correção de programas;
    - Geração automática de implementações;
    - Auxílio no projeto de novas linguagens.

  - **Dificuldades relativas ao uso de Especificações Formais**
    - Inexistência de métodos universais
    - Complexidade dos métodos formais existentes
    - Exige noções matemáticas não triviais
    - Implementações complexas e ineficientes

- **Formas usuais de Especificação Semântica**
  - **Informal**
    - **Especificação textual das regras (ling. natural)**
  - **Semi-Formal**
    - **Esquema de Tradução Dirigida pela Sintaxe**
      - **Criação de Ações Semânticas para verificação das regras semânticas**
      - **Inserção das Ações na GLC**
  - **Formal**
    - **Gramáticas de Atributos**
    - **Semântica de Ações**
    - **Semântica Denotacional (Funções Recursivas)**
    - **Semântica Operacional (Máquinas VHDL)**
    - **Semântica Axiomática (Lógica MTM)**

VI.3 – Analisador Semântico

- **Objetivo**
  - **Verificar se o programa fonte satisfaz as Regras Semânticas da Linguagem**
- **Verificações usuais**
  - **Coerência no uso de Identificadores**
    - **Quanto à declaração:**
      - **Múltiplas declarações de um mesmo identificador**
      - **Ausência de declaração de identificadores;**
    - **Quanto à visibilidade:**
      - **Análise de escopo**
        - **Alcançabilidade dos identificadores – contextos em que é válido o uso de um identificador**
    - **Quanto à coerência entre uso e declaração**
      - **Os identificadores estão sento utilizados de acordo com as regras intrínsecas à categoria de sua declaração?**
      - **Exemplos de uso indevido:**
        - **Uso de constante no lado esquerdo de uma atribuição,**
        - **Indexação de variáveis simples,**
        - **Uso de id de procedimento como operando,**
        - **Erro no número de dimensões de um array**
        - **Uso não qualificado de um campo de registro**
        - **...**

- **Verificação da Compatibilidade de Tipos**
  - **Aplicabilidade dos operadores**
  - **Compatibilidade entre os operandos**
  - **Correspondência entre parâmetros formais / atuais**
  - **Indexação de array's com tipos incompatíveis**
  - **...**
- **Verificação de Referências não resolvidas**
  - **Chamadas de funções / procedures /métodos inexistentes**
  - **Ponteiros para registros inexistentes**
  - **Desvios para label's inexistentes**

- **Outras funções do Analisador Semântico**
  - **Criação e Manutenção da TABELA DE SÍMBOLOS**
    - **funções para inserção, remoção , consulta e atualização dos identificadores e de seus atributos**
  - **Detecção e recuperação de erros semânticos**
    - **erro semântico – violação de uma regra semântica**
    - **recuperação – necessária? métodos?**
  - **Geração de código para verificação de aspectos semânticos em tempo de execução**
    - **Divisão por zero, índices inválidos, etc...**

- **Estrutura do Analisador Semântico**
  - **Autônomo**
    - **Passo independente no processo de compilação**
    - **Entrada:**
      - **Árvore Sintática resultante do parser**
    - **Saída:**
      - **Árvore Sintática Abstrata com Atributos ou código gerado (se integrado com ger. de código)**
  - **Subordinado ao Analisador Sintático**
    - **Ativado após o reconhecimento de determinados elementos sintáticos**

VI.4 - Esquemas de Análise Semântica e de Tradução Dirigidos pela Sintaxe

- **Baseados em Ações Semânticas**
  - **Ações Semânticas**
    - **Ações de Verificação**
      - **Validação Semântica do Programa Fonte**
        - **Verificação da satisfação das regras semânticas**
      - **Montagem da Tabela de Símbolos**
        - **Atualização dos Atributos dos identificadores**
          - **Visando a Análise e a Geração de Código**
    - **Ações de Geração de Código**
      - **Tradução do P.F. p/ P.O (Intermed. / Baixo Nível)**

- **Passos principais**
  - **Identificação das Ações a partir das Regras**
  - **Programação das Ações Semânticas**
  - **Integração das Ações com a GLC**
  - **Adequação do Algoritmo de A. Sintática**
    - **exemplo: No algoritmo LL (1), teriamos:**
      - **Se topo da pilha é ação semântica**

        Então ativa semântico (para execução da ação)

        desempilha ação
  - **Implementação da Tabela de Símbolos**
    - **Definição das Categorias (classes) de identificadores e de seus atributos**
    - **Definição da Estrutura física e lógica (com base nas classes e atributos dos identificadores)**
    - **Definição das Funções de Manipulação**

- **Exemplo de A. Semântica Dirigida pela Sintaxe**
  - **Sintaxe:**

$P \rightarrow D\ C$
$D \rightarrow int\ L\ ;\ D \mid bool\ L\ ;\ D \mid \varepsilon$
$L \rightarrow id\ L1$
$L1 \rightarrow ,\ id\ L1 \mid \varepsilon$
$C \rightarrow id := E\ ;\ C \mid \varepsilon$
$E \rightarrow id\ E1$
$E1 \rightarrow +\ id\ E1 \mid ou\ id\ E1 \mid \varepsilon$

  - **Regras Semânticas:**
    - **Variáveis devem ser declaradas apenas 1 vez;**
    - **Variáveis utilizadas devem ser previamente declaradas**
    - **O operador "+" só se aplica a id do tipo "int" e o resultado de sua aplicação também é "int"**
    - **O operador "ou" só se aplica a id do tipo "bool" e o resultado de sua aplicação também é "bool"**
    - **Na atribuição, o tipo da expressão (E) deve ser igual ao tipo da variável do lado esquerdo**

- **Ações Semânticas**

 # 1 - (* Guarda TIPO da declaração – "int" *)
TipoDecl ← "int"

 # 2 - (* Guarda TIPO da declaração – "bool" *)
TipoDecl ← "bool"

 # 3 - (* Verifica se "id" ainda não foi declarado *)
Se "id" ∈ TS
Entao ERRO SEMÂNTICO ('"id" já declarado')
Senão INCLUIR "id" na TS, juntamente com seu tipo (TipoDecl)

 # 4 – (* Verifica se "id" do lado esq. foi declarado e guarda seu tipo *)
Se "id" ∈ TS
Entao TipoLadoEsq ← TS[id].tipo
Senão ERRO SEMÂNTICO ('"id" não declarado')

 # 5 – (* Verifica se tipo da Expressão = tipo "id" do lado esquerdo *)
Se TipoExpr = TipoLadoEsq
Então "Gera código para atribuição"
Senão ERRO SEMÂNTICO ("Tipos incompatíveis")

 # 6 – (*Verifica se "id" foi declarado e define tipo da expressão (E) *)
Se "id" ∈ TS
Entao TipoExp ← TS[id].tipo
Senão ERRO SEMÂNTICO ('"id" não declarado')

 # 7 – (*Verifica compatibilidade entre operador e operando *)
Se TipoExp ≠ "int"
Entao ERRO SEM. (' Operador "+" exige operando "int"')
Senão "Guarda operador para futura geração de código"

 # 8 – (*Verifica compatibilidade entre operandos *)
Se TS[id].tipo = TipoExp
Entao "Gera código para operação, de acordo com operador")
Senão ERRO SEM. ('Operandos Incompatíveis')

 # 9 – (*Verifica compatibilidade entre operador e operando *)
Se TipoExp ≠ "bool"
Entao ERRO SEM. ('Operador "ou" exige operando "bool"')
Senão "Guarda operador para futura geração de código"

- **GLC + AÇÕES SEMÂNTICAS:**

0: P → D C
1,2,3: D → int #1 L ; D | bool #2 L ; D | ε
4: L → id #3 L1
5,6: L1 → , id #3 L1 | ε
7,8: C → id #4 := E #5 ; C | ε
9: E → id #6 E1
10,11,12: E1 → + #7 id #8 E1 | ou #9 id #8 E1 | ε

- **TABELA DE PARSING LL (1)**

	$	int	bool	ou	id	";"	":="	"+"	","
<P>	0	0	0	-	0	-	-	-	-
<D>	3	1	2	-	3	-	-	-	-
<L>	-	-	-	-	4	-	-	-	-
<L1>	-	-	-	-	-	6	-	-	5
<C>	8	-	-	-	7	-	-	-	-
<E>	-	-	-	-	9	-	-	-	-
<E1>	-	-	-	11	-	12	-	10	-

- **Exemplo de A. Sintática + A Semântica:**

x = int A,B,C;
bool D,E;
A := B + C;
C := D ou A;

VI.5 - TABELA DE SÍMBOLOS (TS)

- **Objetivo : Manter informações sobre identificadores usados no programa fonte**
- **Geralmente construída durante a análise semântica**
- **Usada nas fases de análise semântica e geração de código**
- **Além de identificadores, opcionalmente, a TS poderá conter também palavras reservadas, identificadores especiais e outros símbolos.**
- **Deve ser tão eficiente quanto possível (influi diretamente na performance do compilador)**
- **Que informações devem ser mantidas na TS ???**

  NOME dos identificadores

  CATEGORIA dos identificadores

  ATRIBUTOS dos identificadores

  Quais atributos?

  Depende da CATEGORIA !!!!

  Depende da Semântica da Linguagem !!!

- **Exemplos de categorias de Identificadores:**
  - **Identificador de Programa**
  - **Variável**
  - **Constante**
  - **Tipo**
  - **Elemento de Conjunto, elemento Enumerado**
  - **Parâmetro, Campo de Registro**
  - **Procedimento (Função, Procedure, Método,....)**
  - **Unit, Classe, Módulo, Objeto, Pacote, ...**

- **Exemplo de atributos de identificadores**
  - **Categoria "Variável"**

    Endereço relativo

    Nível em que foi declarada (escopo)
    Deslocamento (posição) dentro do nível

    Tipo (sub-categoria)

    Tipo Simples

    - **pré-definido**
      - **integer, real, char, boolean ...**
    - **definido pelo usuário**
      - **enumerado, intervalo, cadeia**

    Tipo Estruturado (array, registro, conjunto, ...)

    - **array**
      - **dimensões, tipo índice, tipo elementos**
    - **registro (struct)**
      - **qtidade campos, info sobre os campos**
    - **conjunto (set)**
      - **tipo base, info sobre elementos**

- **Implementação da Tabela de Símbolos**
  - **Definição da Estrutura**
    - **Array bi-dimensional**
    - **Multi-listas encadeadas**
    - **Tabela de espalhamento (Hash)**
  - **Operações de Acesso**
    - **Inserção, Exclusão, Consulta e Atualização**
  - **Exemplo de estrutura de uma TS**

```
type atributos = record
                    nome: string [ 30];
                    atributo_1 : <tipo> ;
                        ...
                    atributo_n : <tipo>;
                end;

type tabela_de_símbolos = array [ 1 .. 300 ] of atributos

var TS : tabela_de_simbolos
```

1 – Array bi-dimensional

Nome	Categoria	ATRIB. 1	ATRIB. 2	.....	ATRIB. N
EX	Id-Programa	-	-	-	-
A	Id-Constante	Boolean	true	-	-
X	Id-Variável	Integer	-	-	-
PROC	Id-Função	"tipo do resultado"	num. de parâmetros	...	info M do parametro N

Vantagens:

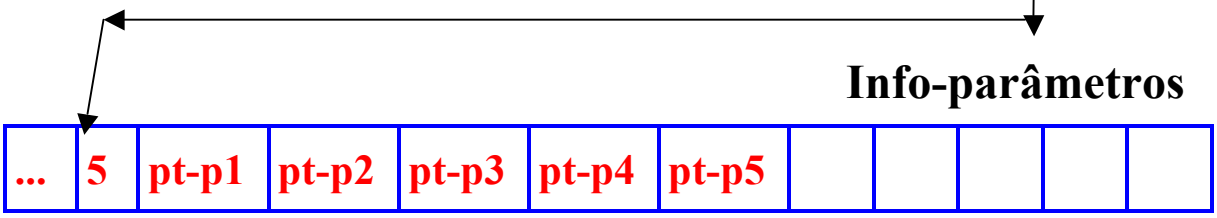
- Simplicidade
- Boa performance na inserção/remoção de identificadores

Desvantagens:

- Baixa performance na busca de identificadores e na atualização de seus atributos (Busca Seqüencial)
- Desperdício de Espaço
  - Categorias com Quantidade de atributos variável
  - diferentes categorias possuem quantidade diferente de atributos

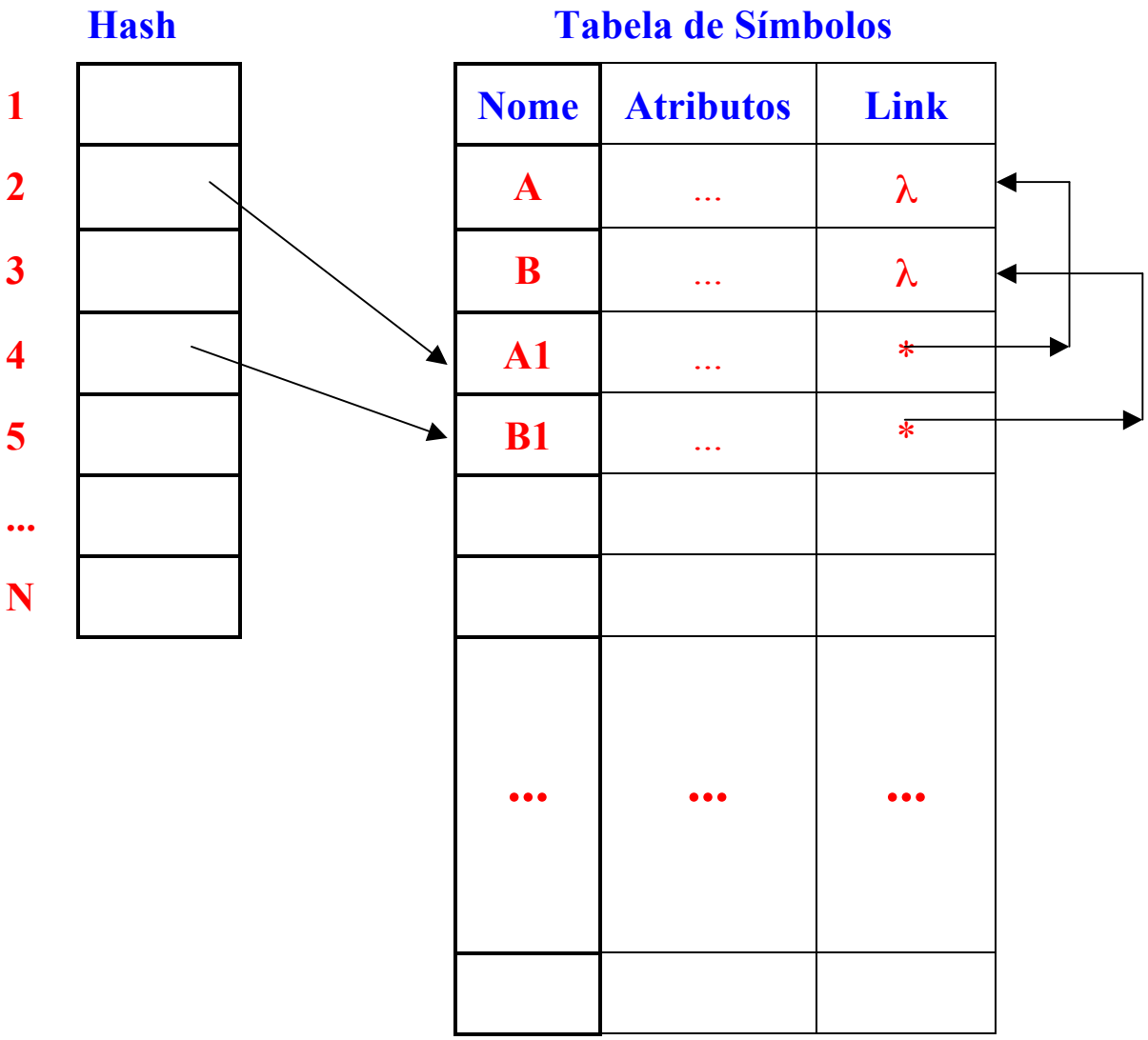
2 – Multi-Listas

Nome	Categoria	Atrib. 1	Atrib. 2	Link
PROG1	id-programa	-	-	-
N	id-constante	int	10	-
X	id-variavel	“nivel”	“desloc”	char
PROC	id-procedimento	função	int	*



- O uso de multi-listas:
  - elimina o desperdício de espaço;
  - diminui a simplicidade;
  - aumenta o tempo de consulta e atualização.

3 - HASH



O uso de HASH :

- Melhora significativamente o tempo de acesso
- Não resolve, por si só, outros problemas
- Piora a performance na inclusão e na exclusão