6 – Análise Semântica

Linguagens e Programação

Ana Madureira
Engenharia Informática
Ano Letivo: 2021/2022

Fontes:

1. Compiladores Princípios e práticas, Keneth C.Louden, Thomson, 2004.
Cap. 6 Análise Semântica

2. Processadores de Linguagens – da concepção à implementação, Rui Gustavo Crespo. IST Press.1998.
Cap. 6 Análise Semântica

3. Compiladores Princípios,Técnicas e Ferramentas AlfredV.Aho,R.Sethi e Jeffrey D.Ullman, 2007.
Cap. 5 Tradução dirigida pela Sintaxe

1

O compilador até à análise semântica

- Análise léxica
  - Separa o texto de entrada numa sequência de tokens
  - Deteta textos de entrada com tokens ilegais (erros léxicos)
- Análise sintática
  - Aplica regras gramaticais
  - Constrói (por vezes implicitamente) uma árvore de parse com os tokens de entrada
  - Deteta textos com sequências ilegais de tokens (erros sintáticos)
- Cada uma destas fases deteta certos erros de programação mas não todos
- A última fase envolvida na análise é a análise semântica
  - Destinada a verificar se as regras semânticas da linguagem são satisfeitas e a calcular os valores associados aos símbolos de modo a poder conhecer-se o significado completo da frase

Existem situações em que não é possível ao compilador efectuar alguns tipos de verificação. Por exemplo:

```
tabela: array[0..255] of char;
i: integer;
```

Não é possível garantir durante a compilação que **tabela[i]** é uma expressão válida, pois **i** pode vir a tomar valores fora do intervalo [0..255]

2

Análise Semântica

- A análise semântica é a última das fases do processo de análise de um compilador.
- O analisador semântico deve verificar se as construções estão corretas do ponto de vista semântico da linguagem. Assim as verificações realizadas pelo analisador semântico podem ser classificadas em:
  1. **verificação de tipos**
  2. **verificação do fluxo de controle**
  3. **verificação de unicidade**
  4. **verificação relacionada aos nomes**

3

Funções do analisador semântico

- Executar uma grande quantidade de verificações
  - todos os identificadores foram declarados
  - não existência de declarações múltiplas e incompatíveis para um mesmo identificador
  - determinar os tipos de expressões e variáveis e a sua compatibilidade quando aparecem numa mesma expressão
  - relações de herança (em linguagens OO)
  - boa utilização de palavras e identificadores reservados
  - não existência de métodos com definição múltipla numa classe (em linguagens OO)
- Os requisitos exatos dependem da linguagem que se está a analisar
- Praticamente todas as linguagens requerem as três primeiras verificações (exceto as linguagens que não requerem a declaração de identificadores)

4

Implementação da análise semântica

- A análise semântica é implementada utilizando extensões das gramáticas livres de contexto utilizadas pelos analisadores sintáticos
- Essas extensões designam-se por definições dirigidas pela sintaxe (*syntax-directed definitions*)
- Estas definições são compostas por dois tipos de extensões das gramáticas independentes do contexto (GIC)
  - **Atributos** - valores associados a cada símbolo (terminais e não-terminais) da gramática
    - Os atributos podem representar uma multiplicidade de propriedades dos símbolos gramaticais, tais como:
      - Tipos das variáveis
      - Valores de constantes e expressões
      - Endereços das variáveis
      - Endereços dos procedimentos e funções
  - **Regras ou acções semânticas** - cálculos ou outras ações executadas quando se aplica uma regra gramatical (produção) na análise sintática
    - Estas ações deverão ser executadas no final do reconhecimento das regras ou a meio destas. Calculam atributos ou executam outras ações

5

Atributos e ações

- Considere-se uma produção de uma GIC da forma:
  $Y \rightarrow X_1 X_2 \ldots X_n$
  - Os atributos são valores associados aos símbolos da gramática (nós da árvore de parse). Por exemplo poderão existir os atributos Y.type, X1.val, X2.type, etc, associados aos símbolos da regra anterior. Cada símbolo pode ter vários atributos.
  - Os símbolos terminais obtêm os seus atributos do analisador léxico (valor de uma constante, nome de uma variável, etc.)
  - As ações são pedaços de código associados à regra (geralmente no fim)
- Exemplo:
  - Considere-se a seguinte regra de uma gramática: N → N digit em que o não-terminal N e o terminal digit possuem um atributo chamado val.

    Esta regra e a respetiva ação semântica poderia ser escrita como:
    - N1 → N2 digit { N1.val = N2.val × digit.val; }
    - Os índices distinguem apenas as várias instâncias de N dentro da regra
  - Outro exemplo : Duas regras de uma gramática e respetivas ações embebidas
    - R1 → + { print('+'); } T R2
    - T → num { print (num.val); }

6

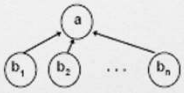
Gramáticas de atributos

- Definições **L-atribuídas** – cujos atributos podem sempre ser avaliados numa ordem de pesquisa em profundidade
- Definições **S-atribuídas** – definição dirigida pela sintaxe que usa exclusivamente atributos sintetizados.
- **Gramáticas de atributos** – como o conjunto de atributos e regras semânticas que expressam como a computação dos atributos se relacionam com as regras gramaticais da linguagem.
- A avaliação das regras semânticas define os valores dos atributos nos nós da árvore gramatical, tem apenas que produzir a mesma saída para cada cadeia de entrada.

7

Atributos sintetizados e herdados

- As ações semânticas definem dependências entre os atributos
  - Se, por exemplo, existir uma ação do tipo a = f(b_1, b_2, …, b_n), onde a e b_1 a b_n são atributos associados aos símbolos de uma regra, diz-se que a depende dos atributos b_1 a b_n
  - Essa dependência pode exprimir-se através de um grafo (de dependências) onde os atributos são nós, e há ramos dirigidos dos atributos usados no cálculo para os atributos dependentes:



  - Se na totalidade do conjunto de ações de uma gramática apenas houver atributos que dependam dos atributos dos filhos numa árvore de parse, diz-se que essa gramática (estendida com atributos e ações) é **S-attributed** e esses atributos dizem-se **sintetizados**
  - Se houver dependências entre atributos de símbolos do lado direito das regras (filhos), ou estes dependerem de atributos do não-terminal do lado esquerdo (pai), estes atributos dizem-se **herdados**.
  - Se, para os atributos herdados, estes dependerem apenas de atributos à sua esquerda ou do pai, diz-se que essa gramática é **L-attributed**

8

Cálculo dos atributos

- Os atributos são calculados nas ações semânticas. As acções de cálculo só podem ser executadas quando todos os atributos de que dependem já estiverem disponíveis
- As ações de todas as gramáticas *S-attributed* e de muitas outras que sejam *L-attributed* podem ser executadas durante a análise sintática pelo *parser* e sem necessidade da construção explicita da árvore de *parse*
- No caso geral poderá ser necessário construir explicitamente a árvore de *parse* e o grafo de dependências
  - Cálculo dos atributos no caso geral:
    - Construir a árvore de *parse*
    - Construir o grafo de dependências
    - Fazer uma ordenação topológica do grafo de dependências
    - Seguir essa ordenação
  - Para que seja possível efetuar a ordenação topológica o grafo de dependências terá que ser acíclico

9

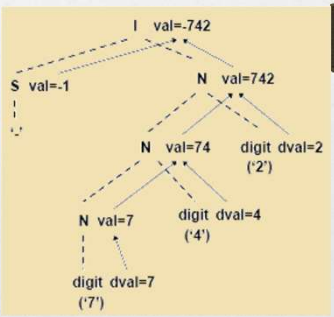
Exemplo 1

- Considere-se a seguinte gramática para números inteiros em que os *tokens* são os dígitos ('0' a '9') e os sinais '+' e '-'. O objetivo é a determinação do valor de cada inteiro. Para isso associa-se aos não-terminais I, S e N um atributo inteiro *val* e pressupõe-se que o analisador léxico retorna para o *token* 'digit' um atributo *dval* (digit value) entre 0 e 9.

Gramática	Acções semânticas
$I \rightarrow S\ N$	I.val = S.val × N.val
$S \rightarrow +$	S.val = 1
$S \rightarrow -$	S.val = -1
$S \rightarrow \varepsilon$	S.val = 1
$N_1 \rightarrow N_2$ digit	N_1.val = N_2.val × 10 + digit.dval
$N \rightarrow$ digit	N.val = digit.dval

As acções semânticas estabelecem dependências entre os atributos, indicadas pelas setas, nesta árvore de *parse*

Seguindo esta gramática, uma árvore de *parse* para o texto -742 seria:



10

Exemplo 2

- Considere-se a seguinte gramática para declarações de variáveis (numa linguagem tipo C). O objetivo é adicionar a uma estrutura de dados externa – a tabela de símbolos – os nomes das variáveis declaradas, juntamente com o seu tipo. Para isso associa-se aos não-terminais T e L um atributo **type** que pode ter os valores real e **integer**. Pressupõe-se que o analisador léxico retorna o atributo **name** para os **tokens** id.
- A função **insert_var(name, type)** insere uma nova variável na tabela de símbolos.

Gramática	Acções semânticas
D → T L	L.type = T.type
T → int	T.type = integer
T → float	T.type = real
$L_1 \rightarrow L_2$, id	L_2.type = L_1.type insert_var(id.name,L1.type)
L → id	insert_var(id.name,L.type)

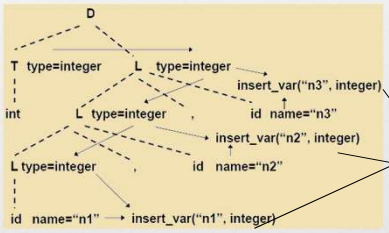


Tabela de símbolos:

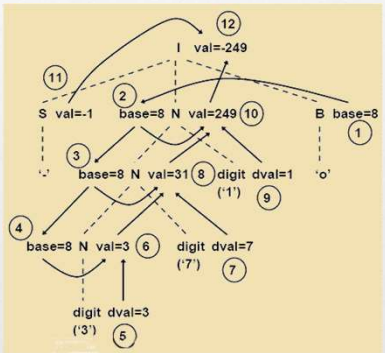
n1	integer
n2	integer
n3	integer

11

Exemplo 3

- Considere-se novamente a gramática do exemplo anterior mas agora estendida de modo a indicar a base do número inteiro (octal ou decimal):

Gramática	Acções semânticas
I → S N B	I.val = S.val × N.val N.base = B.base
B → o	B.base = 8
B → d	B.base = 10
S → +	S.val = 1
S→ -	S.val = -1
S → ε	S.val = 1
$N_1 \rightarrow N_2$ digit	N_1.val = N_2.val × N_1.base + digit.dval N_2.base = N_1.base if (N_1.base== 8 && digit.dval>=8) N_1.val = 0; error()
N → digit	N.val = digit.dval if (N.base==8 && digit.val>=8) N.val = 0; error()



Pode ver-se uma ordenação topológica do grafo de dependências. Se chamarmos a_k ao atributo do nó k, temos os seguintes cálculos:

a2 = a1 (8)
a3 = a2 (8)
a4 = a3 (8)
a6 = a5 (3)
a8 = a6xa3 + a7 (31)
a10 = a8xa2 + a9 (249)
a12 = a11xa10 (-249)

12

Ações e atributos durante a análise sintática top-down

- Praticamente todas as gramáticas S-attributed ou L-attributted são de fácil avaliação durante a análise sintática num parser de descida recursiva (top-down)
- A técnica a usar é a seguinte:
  - Nestes parsers há um procedimento para expandir cada símbolo gramatical não-terminal
  - Estes procedimentos são substituídos por funções onde:
    - os atributos herdados do símbolo a expandir são passados como argumentos da função
    - os atributos sintetizados são calculados na função e retornados por esta
  - Como os atributos herdados numa gramática L-attributed dependem apenas de atributos à esquerda numa regra, quando se chama a função de expansão já deverão estar disponíveis para serem passados como parâmetros
  - Os atributos sintetizados dependem apenas dos filhos, que vão retornando os respetivos atributos, permitindo o cálculo dos atributos atuais e o seu retorno a quem chamou a função

13

Cálculo de atributos sintetizados em parsers bottom-up

- Os parsers bottom-up (SLR/LALR/LR) mantêm uma stack onde vão sendo armazenados os terminais e não-terminais. Quando os n símbolos do topo da stack coincidem com a parte direita de alguma regra gramatical, podem ser substituídos pelo símbolo da parte esquerda (efetua-se uma redução)
- É na altura das reduções que se devem executar as ações semânticas
- Para isso a stack do parser é estendida por forma a incorporar os atributos associados a cada símbolo; (cada elemento da stack pode ser uma estrutura com vários campos)
- Os atributos sintetizados dependem apenas dos atributos dos filhos, numa árvore de parse
- Os filhos, na aplicação de uma regra sintática, estão no topo da stack do parser

Exemplo: Regra E1 → E2 + T {E1.val = E2.val + T.val}

Símb.	Val
T	10
+	
E	35

→

Símb.	Val
E	45

14

Exemplo

Cálculo de expressões de dígitos:

Gramática	Acções semânticas
L → E	print(val[top])
$E_1 \rightarrow E_2 + T$	val[top-2]=val[top-2] + val[top]
E → T	
$T_1 \rightarrow T_2 * F$	val[top-2]=val[top-2] * val[top]
T → F	
F → (E)	val[top-2]=val[top-1]
F → digito	

Configuração do *parser* durante a análise de:
3 * 5 + 4 $

Entrada	Símbolo	Val	Regra
3*5+4$			
*5+4$	3	3	
*5+4$	F	3	F → digito
*5+4$	T	3	T → F
5+4$	T *	3 -	
+4$	T * 5	3 - 5	
+4$	T * F	3 - 5	F → digito
+4$	T	15	T → T * F
+4$	E	15	E → T
4$	E +	15 -	
$	E + 4	15 - 4	
$	E + F	15 - 4	F → digito
$	E + T	15 - 4	T → F
$	E	19	E → E + T
$	L	19	L → E (print(19))

15

Implementação de ações a meio em *parsers bottom-up*

- Nos esquemas de tradução aparecem ações semânticas no meio das produções
- Num *parser bottom-up* só é possível executar as ações semânticas quando se aplica a redução de uma regra gramatical

  Como se implementam então as execuções destas ações semânticas ?
- Com a introdução de novos não-terminais marcadores

Gramática e semântica para a escrita pós fixa de expressões (tradução de uma expressão em notação infixa para notação pós fixa):

```
E → T R
R → + T {print('+')} R |
    - T {print('-')} R |
    ε
T → num {print(num.val)}
```

```
E → T R
R → + T M R |- T N R |ε
T → num {print(num.val)}
M → ε {print('+')}
N → ε {print('-')}
```

Os novos não-terminais M e N servem apenas para a execução das acções semânticas que aparecem no seu lugar

16

Atributos herdados em parsers bottom-up

- Certos atributos herdados podem facilmente ser implementados num parser bottom-up
- Num parser bottom-up uma regra do tipo A → XY é aplicada quando na stack do parser já existem os símbolos X e Y (este no topo); eles são então substituídos pelo símbolo A
- No entanto quando o símbolo Y é construído (através de reduções na sua sub-árvore), já se encontram disponíveis atributos sintetizados de X (na stack do parser)
- Estes atributos podem então ser herdados por um ou mais atributos de Y: por exemplo pela regra Y.i = X.s, em que i é um atributo herdado de Y e s um atributo sintetizado de X

17

Exemplo

Considere-se novamente a gramática para declarações, agora num esquema de tradução:

Gramática	Acções semânticas
D → T L	L.type = T.type
T → int	T.type = integer
T → float	T.type = real
$L_1 \rightarrow L_2$, id	L_2.type = L_1.type insert_var(id.name,L1.type)
L → id	insert_var(id.name,L.type)

A sequência de acções que um *parser bottom-up* executaria para a entrada "float p, q, r" pode ver-se a seguir:

Entrada	Símbolo	Regra
float p, q, r		
p, q, r	float	
p, q, r	T	T → float
, q, r	T p	
, q, r	T L	L → id
q, r	T L ,	
, r	T L , q	
, r	T L	L → L , id
r	T L ,	
	T L , r	
	T L	L → L , id
	D	D → T L

18

Exemplo

- Note-se que sempre que se aplica uma regra gramatical relativa ao símbolo L, por baixo, na stack do *parser*, está sempre o símbolo T (e o seu atributo *type*). Logo a semântica pode ser simplificada, para a que se mostra em baixo:

Gramática	Acções semânticas
$D \rightarrow T\ L$	
$T \rightarrow int$	type[top]=integer
$T \rightarrow float$	type[top]=float
$L_1 \rightarrow L_2$, id	insert_var(name[top], type[top-3])
$L \rightarrow id$	insert_var(name[top], type[top-1])

- Neste exemplo os atributos herdados *type* do não-terminal L vão buscar o seu valor ao atributo *type* de T. Este está na stack do parser sempre numa posição bem definida.

19