Análise Sintática

Tipos de Analisadores Gramáticais: PARSERS

* Analisadores TOP-DOWN:
  Árvore de derivação começa pela raiz indo para as folhas => Análise Descendente
  Tipo LL(1): Left to right / Leftmost derivation / 1 symbol each time - lookahead
  - Recursivo com Retrocesso (Backtracking)
  - Recursivo Preditivo
  - Preditivo Tabular (não recursivo - pilha + tabela)

* Analisadores BOTTON-UP:
  Árvore de derivação começa pelas folhas indo para a raiz => Análise Ascendente
  Shift / Reduce <= Análise Redutiva
  - LR(k) => i) SLR (simple)
  ii) LR canônicos
  iii) LALR (lookahead LR)
  Tipo LALR(1) => Yacc / Bison - Left to right / Rightmost derivation / 1 each time

* Recuperação de Erros
* Tradução dirigida pela sintaxe

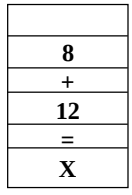
1

Análise Sintática

* Analisadores Gramaticais **BOTTON-UP**:
  Árvore de derivação começa pelas folhas indo para a raiz => Análise Ascendente
  Shift / Reduce <= Análise Redutiva
  Tipo LALR(1) => Yacc / Bison - Left to right / Rightmost derivation / 1 each time

  Exemplo: Shift (empilha) / Reduce (Busca chegar no S - símbolo inicial)

Sentença:
X = 12 + 8



Empilha: shifting tokens on to the internal stack

P = { statement => name = expr |
expr
expr => number + number |
number - number }

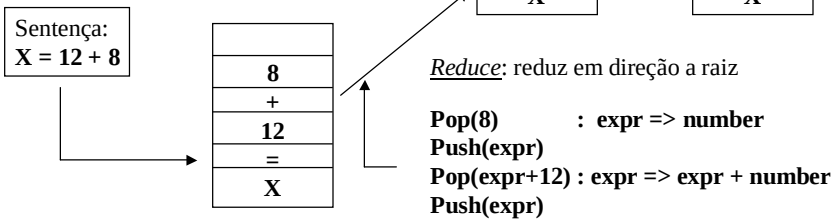
P = { statement => name = expr |
expr
expr => number
expr => expr + number |
expr - number }

2

Análise Sintática

* Analisadores Gramaticais **BOTTON-UP**:

P = { statement => name = expr |
expr
expr => number
expr => expr + number |
expr - number }



Sentença:
X = 12 + 8

Empilha: shifting tokens on to the internal stack

Reduce: reduz em direção a raiz

Pop(8) : expr => number
Push(expr)
Pop(expr+12) : expr => expr + number
Push(expr)
Pop(X=expr) : statement => name = expr
Push(statement)
Finish => Initial Symbol

3

Análise Sintática

* Analisadores Gramaticais **BOTTON-UP**:
Implementação do reconhecimento de uma sentença

PILHA	ENTRADA	AÇÃO
(1) $	$id_1 + id_2 * id_3$	shift
(2) $id1	$+ id_2 * id_3$	reduce por E → id
(3) $E	$+ id_2 * id_3$	shift
(4) $E +	$id_2 * id_3$	shift
(5) $E + id2	$* id_3$	reduce por E → id
(6) $E + E	$* id_3$	shift
(7) $E + E *	id_3	shift
(8) $E + E * id3	$	reduce por E → id
(9) $E + E * E	$	reduce por E → E * E
(10) $E + E	$	reduce por E → E + E
(11) $E	$	accept

Sentença

$id_1+id_2*id_3$

Gramática

$E \rightarrow E + E$
$E \rightarrow E * E$
$E \rightarrow (E)$
$E \rightarrow id$

Tipos de Ações

1. *shift*: o próximo símbolo de entrada é colocado no topo da pilha.
2. *reduce*: o analisador reconhece o lado direito do *handle* que está no topo da pilha, devendo então pesquisar o lado esquerdo e decidir que não-terminal será utilizado para substituí-lo.
3. *accept*: o analisador identifica um estado de final de análise, com sucesso.
4. *error*: o analisador identifica um erro de sintaxe e chama (se houver) uma rotina de recuperação de erro.

4

Análise Sintática – Botton Up Parsing

* Analisadores Gramaticais **BOTTON-UP**:

Implementação de um reconhecedor...

OCW MIT – OpenCourseWare
Web: http://www.ocw.mit.edu/

6.035 - Computer Language Engineering Fall 2002
Lecture 4: Shift-Reduce Parsing

Web: http://ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-035Computer-Language-EngineeringFall2002/CourseHome/

Material de apoio...

- Implementação de Linguagens de Programação: Compiladores
  Ana Price e Simão Toscani – Pags. 53 à 73
- Compiladores : Princípios, Técnicas e Ferramentas
  Aho, Sethi, Ullman – Pags. 86 à 111

5

Análise Sintática – Botton Up Parsing

* Analisadores Gramaticais **BOTTON-UP**:

Conflito...

- conflito *shift-reduce*, , no qual o analisador não sabe se deve realizar uma ação *shift* ou *reduce*;
- conflito *reduce-reduce*, no qual o analisador não consegue decidir que redução deve ser feita, já que várias possibilidades são adequadas.

Conflito shift-reduce

comando → *if expr then comando*
| *if expr then comando else comando*
| *outro*

tem-se o conflito *shift-reduce* na configuração:

Pilha: ... *if expr then comando*
Entrada: *else ...$*

Conflito reduce-reduce

comando → *id(listaParâmetros)*
| *expr := expr*

listaParâmetros → *listaParâmetros, parâmetro*
| *parâmetro*

parâmetro → *id*

expr → *id(listaExpr)*
| *id*

listaExpr → *listaExpr, expr*
| *expr*

Normalmente, em um conflito *shift-reduce*, o analisador opta pela ação *shift*.

6

That´s all foks!

Análise Sintática – Botton Up Parsing

COMPILERS:
May the Force be with you...