Primeiros(α) :

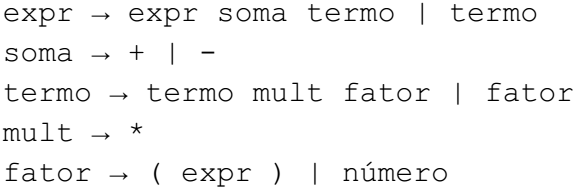
* α é uma cadeia de símbolos gramaticais.
  1. Obs: Não necessariamente uma cadeia de terminais!!
* É o conjunto de terminais que começam as cadeias derivadas de α

Seguidores(A)

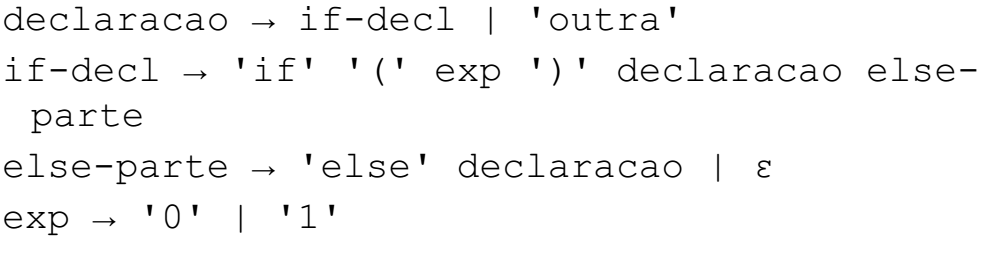
* A é um não-terminal
* É o conjunto de terminais que podem aparecer imediatamente após A em alguma forma sentencial.
  1. Em outras palavras, é o conjunto de terminais a tal que existe uma derivação na forma S => αAaβ
  2. { a | S => αAaβ }
  3. Símbolo especial $ (fim de cadeia)
  4. Se A pode ser o símbolo mais à direita em alguma forma sentencial, $ está em seguidores (A)

Para calcular o conjunto seguidores:

1. Adicione $ a seguidores(S)
   1. S é o símbolo inicial
2. Se existir uma produção A 🡪 αBβ, adicione os primeiros(β) a seguidores(B) (exceto ε)
3. Se existir uma produção A 🡪 αB, ou uma produção A 🡪 αBβ onde primeiros(β) contém ε, adicione os seguidores(A) a seguidores(B)



|  |  |  |
| --- | --- | --- |
|  | Primeiros | Seguidores |
| expr | “(“, número | $, +, -, ) |
| Soma | +, - | “(“, número |
| Termo | “(“, número | $, \*, +, -, ) |
| Mult | \* | “(“, número |
| fator | “(“, número | $, \*, +, -, ) |

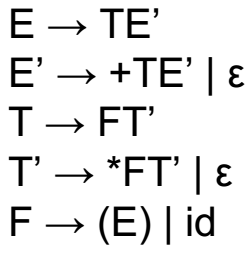


|  |  |  |
| --- | --- | --- |
|  | Primeiros | Seguidores |
| Declaração | ‘if’, ‘outra’ | $, ‘else’ |
| If-decl | ‘if’ | $, ‘else’ |
| Else-parte | ‘else’, ε | $, ‘else’ |
| Exp | ‘0’, ‘1’ | ‘)’ |

Uma gramática G é LL(1) se: para duas produções distintas G

* A 🡪 α | β :
  1. Para um terminal “a”, tanto α quanto β não derivam cadeias começando com “a”;
  2. No máximo um dos dois, α ou β, pode derivar a cadeia vazia;
  3. Se β => \* ε, então α não deriva nenhuma cadeia começando com um terminal em seguidores(A).
     + Se α => \* ε, então β não deriva ... ... terminal em seguidores(A).

Tabela LL(1)

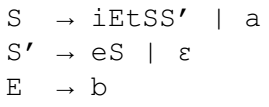


Primeiros(E) = { ‘(‘, ‘id’ } Primeiros(E’) = { ‘+’, ε} Primeiros(T’) = { ‘\*’, ε } Primeiros(T) = { ‘(‘, ‘id’ } Primeiros(F) = { ‘(‘, ‘id’ } Seguidores(E) = { $, ‘)’ } Seguidores(E’) = { $, ‘)’ } Seguidores(T) = { ‘+’, $, ‘)’ }

Seguidores(T’) = { ‘+’, $, ‘)’ }

Seguidores(F) = { ‘\*’, ‘+’, $, ‘)’ }

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | + | \* | ( | ) | id | $ |
| E |  |  | E🡪TE’ |  | E🡪TE’ |  |
| E’ | E’🡪+TE’ |  |  | E’🡪 ε |  | E’🡪 ε |
| T |  |  | T🡪FT’ |  | T🡪FT’ |  |
| T’ | T’🡪ε | T’🡪\*FT’ |  | T’🡪ε |  | T’🡪ε |
| F |  |  | F🡪(E) |  | F🡪id |  |



Primeiros(S) = { ‘i’, ‘a’ } Primeiros(S’) = { ‘e’, ε } Primeiros(E) = { b }

Seguidores(S) = { $, ‘e’ } Seguidores(S’) = { $, ‘e’ } Seguidores(E) = { t }

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | i | t | a | e | b | $ |
| S | S🡪iEtSS’ |  | S🡪a |  |  |  |
| S’ |  |  |  | S’🡪eS  S’🡪 ε |  | S’ 🡪 ε |
| E |  |  |  |  | E🡪b |  |

Não-determinismo causado pela ambiguidade da gramática

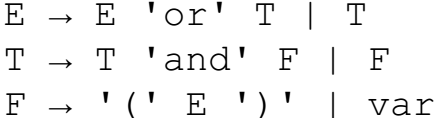
Análise Sintática Preditiva Sem Recursão

Entrada = id + id \* id $

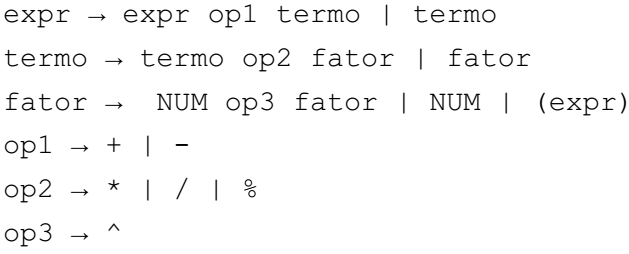
|  |  |  |  |
| --- | --- | --- | --- |
| Casamento | Pilha | Entrada | Ação |
|  | **E**$ | **id** + id \* id $ | E🡪TE’ |
|  | **T**E’$ | **id** + id \* id $ | T🡪FT’ |
|  | **F**T’E’$ | **id** + id \* id $ | F🡪id |
| Id | **id**T’E’$ | **id** + id \* id $ | match |
| id | T’E’$ | **+** id \* id $ | T’🡪 ε |
|  | **E’**$ | **+** id \* id $ | E’🡪+TE’ |
|  | **+**TE’$ | **+** id \* id $ | match |

...

Tabela de Precedência



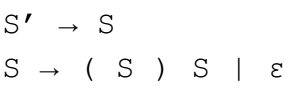
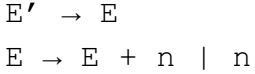
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | var | ‘or’ | ‘and’ | ‘(‘ | ‘)’ | $ |
| var | - | > | > |  | > | > |
| ‘or’ | < | > | < | < | > | > |
| ‘and’ | < | > | > | < | > | > |
| ‘(‘ | < | < | < | < | = |  |
| ‘)’ |  | > | > |  | > | > |
| $ | < | < | < | < |  | ok |



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | NUM | + | - | \* | / | % | ^ | ( | ) | $ |
| NUM |  | > | > | > | > | > | > |  | > | > |
| + | < | > | > | < | < | < | < | < | > | > |
| - | < | > | > | < | < | < | < | < | > | > |
| \* | < | > | > | > | > | > | < | < | > | > |
| / | < | > | > | > | > | > | < | < | > | > |
| % | < | > | > | > | > | > | < | < | > | > |
| ^ | < | > | > | > | > | > | < | < | > | > |
| ( | < | < | < | < | < | < | < | < | = |  |
| ) |  | > | > | > | > | > | > |  | > | > |
| $ | < | < | < | < | < | < | < | < |  | OK |

Itens LR(0):

G1:

S’ 🡪 .S E’ 🡪 .E

S’ 🡪 S. E’ 🡪 E.

S 🡪 .( S ) S E 🡪 . E + n

S 🡪 (. S ) S E 🡪 E . + n

S 🡪 ( S . ) S E 🡪 E + . n

S 🡪 ( S ) . S E 🡪 E + n .

S 🡪 ( S ) S . E 🡪 . n

S 🡪 . E 🡪 n.

S : bAb

A : CB | a

B : Aa

C : c | ε

Primeiros(S): { b }

Primeiros(A): { c, a }

Primeiros(B): { c, a }

Primeiros(C): { c, ε }

Seguidores(S): { $ }

Seguidores(A): { a, b }

Seguidores(B): { c, a }

Seguidores(C): { a, b }

S : A | B | ε

A : A + B | A – B | 1 | 2 | 3 | ε

B : A | C

C : ( A )

Primeiros(S): { ‘(‘, 1, 2, 3, +, -, ε }

Primeiros(A): { 1, 2, 3, +, -, ε }

Primeiros(B): {‘(‘, 1, 2, 3, +, -, ε }

Primeiros(C): { ‘(‘ }

Seguidores(S): { $ }

Seguidores(A): { +, -, ‘)’, $ }

Seguidores(B): { $, +, -, ‘)’ }

Seguidores(C): { $, +, -, ‘)’ }

S : ε | abA | abB | abC

A : aSaa | b

B : bSbb | c

C : cScc | d

Primeiros(S): { ε, a }

Primeiros(A): { a, b }

Primeiros(B): { b, c }

Primeiros(C): { c, d }

Seguidores(S): { $, a, b, c }

Seguidores(A): { $, a, b, c }

Seguidores(B): {$, a, b, c }

Seguidores(C): {$, a, b, c}

Considere a gramática:

Lexp : átomo | lista

Átomo : numero | identificador

Lista : ( lexpseq )

Lexpseq: lexpseq lexp | lexp

1. Remova recursão à esquerda:

Lexp: átomo | lista

Átomo: numero | identificador

Lista: ( lexpseq )

Lexpseq: lexp lexpseq2

Lexpseq2: lexp lexpseq2 | ε

b)

primeiros(lexp) = { numero, identificador, ( }

primeiros(átomo) = { numero, identificador }

primeiros(lista) = { ( }

primeiros(lexpseq) = { numero, identificador, ( }

primeiros(lexpseq2) = { numero, identificador, (, ε }

seguidores(lexp)= { $, numero, identificador, (, ) }

seguidores(átomo)= { $, numero, identificador, (, ) }

seguidores(lista) = { $, numero, identificador, (, ) }

seguidores(lexpseq) = { ) }

seguidores(lexpseq2) = { ) }

c ) tabela de análise sintática

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | numero | identificador | ( | ) | $ |
| Lexp | 🡪atomo | 🡪atomo | 🡪lista |  |  |
| Átomo | 🡪numero | 🡪ident. |  |  |  |
| Lista |  |  | 🡪(lexpseq) |  |  |
| Lexpseq | 🡪lexp lexpsq2 | 🡪lexp lexpsq2 | 🡪lexp lex2 |  |  |
| Lexpseq2 | 🡪lexp lexpsq2 | 🡪lexp lexpsq2 | 🡪lexp lex2 | 🡪 ε |  |

Tabela de precedência:

S 🡪 ( S + S ) | ( S \* S ) | a | b

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | ( | ) | + | \* | a | b | $ |
| ( | < |  | = | = | < | < |  |
| ) |  | > | > | > |  |  | > |
| + | < | = |  |  | < | < |  |
| \* | < | = |  |  | < | < |  |
| a |  | > | > | > |  |  | > |
| b |  | > | > | > |  |  | > |
| $ | < |  |  |  | < | < | OK |

2)

|  |  |  |
| --- | --- | --- |
| pilha | entrada | Ação |
| $ | (a\*b)$ | Empilha ( |
| $( | A\*b)$ | Empilha a |
| $(a | \*b)$ | Reduz, gancho = a |
| $(S | \*b)$ | Empilha \* |
| $(S\* | b)$ | Empilha b |
| $(S\*b | )$ | Reduz, gancho = b |
| $(S\*S | )$ | Empilha ) |
| $(S\*S) | $ | Reduz, gancho = (S\*S) |
| $S | $ | aceita |

3) Construa a tabela de prec. p/ gramatica abaixo:

S 🡪 ( L ) | a

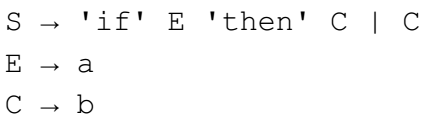
L 🡪 L , S | S

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ( | ) | a | , | $ |
| ( | < | = | < | < |  |
| ) |  | > |  | > | > |
| a |  | > |  | > | > |
| , | < | > | < | > |  |
| $ | < |  | < |  | OK |

4) i. (a, a)

|  |  |  |
| --- | --- | --- |
| pilha | entrada | Ação |
| $ | (a,a)$ | Empilha ( |
| $( | A,a)$ | Empilha a |
| $(a | ,a)$ | Reduz, gancho = a |
| $(S | ,a)$ | Empilha , |
| $(S, | a)$ | Empilha a |
| $(S,a | )$ | Reduz, gancho = a |
| $(S,S | )$ | Reduz, gancho = S,S |
| $(L | )$ | Empilha ) |
| $(L) | $ | Reduz, gancho = (L) |
| $S | $ | aceita |

1. Dada a gramatica



1. Construa a tabela SLR

S’ 🡪 S

S 🡪 ‘if’ E ‘then’ C | C

E 🡪 a

C 🡪 b

C = {

I0 = { [S’ 🡪 .S], [S 🡪. ‘if’E ‘then’ C], [S 🡪 .C], [C🡪.b] },

I1 = { [S’ 🡪 S.] },

I2 = { [S 🡪 ‘if’ . E ‘then’ C], [E 🡪 .a] },

I3 = { [S 🡪 C.] },

I4 = { [C 🡪 b.] },

I5 = { [S 🡪 ‘if’ E. ‘then’ C] },

I6 = { [E🡪a.]},

I7 = { [S 🡪 ‘if’ E ‘then’ . C], [C 🡪 .b]},

I8 = { [S 🡪 ‘if’ E ‘then’ C .] }

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | ‘if’ | ‘then’ | a | b | S | E | C |
| I0 | I2 |  |  | I4 | I1 |  | I3 |
| I1 |  |  |  |  |  |  |  |
| I2 |  |  | I6 |  |  | I5 |  |
| I3 |  |  |  |  |  |  |  |
| I4 |  |  |  |  |  |  |  |
| I5 |  | I7 |  |  |  |  |  |
| I6 |  |  |  |  |  |  |  |
| I7 |  |  |  | I4 |  |  | I8 |
| I8 |  |  |  |  |  |  |  |

Primeiros(S’) = { ‘if’, b }

Primeiros(S) = { ‘if’, b }

Primeiros(E) = { a }

Primeiros(C) = { b }

Seguidores(S’) = { $ }

Seguidores(S) = { $ }

Seguidores(E) = { ‘then’ }

Seguidores(C) = { $ }

1. S 🡪 ‘if’ E ‘then’ C
2. S 🡪 C
3. E 🡪 a
4. C 🡪 b

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Ação | | | | | Transição | | |
| Estados | ‘if’ | ‘then’ | a | b | $ | S | E | C |
| 0 | S2 |  |  | S4 |  | 1 |  | 3 |
| 1 |  |  |  |  | OK |  | 5 |  |
| 2 |  |  | S6 |  |  |  |  |  |
| 3 |  |  |  |  | R2 |  |  |  |
| 4 |  |  |  |  | R4 |  |  |  |
| 5 |  | S7 |  |  |  |  |  |  |
| 6 |  | R3 |  |  |  |  |  |  |
| 7 |  |  |  | S4 |  |  |  | 8 |
| 8 |  |  |  |  | R1 |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| pilha | símbolos | entrada | Ação |
| 0 |  | If a then b $ | S2 |
| 0 2 | if | A then b $ | S6 |
| 0 2 6 | If a | Then b $ | R3 |
| 0 2 5 | If E | Then b $ | S7 |
| 0 2 5 7 | If E then | b $ | S4 |
| 0 2 5 7 4 | If E then b | $ | R4 |
| 0 2 5 7 8 | If E then C | $ | R1 |
| 0 1 | S | $ | OK |