LL Parser (TOP-DOWN)

...

CT-200

Thiago Silva de Oliveira Duarte
Marcus Kimura Lopes

TOP DOWN x BOTTON UP PARSER

TOP DOWN

Algoritmo começa do símbolo de início aplicando produções até alcançar a string desejada

BOTTON UP

Algoritmo começa da string e reduz até o símbolo inicial identificando as produções que ele gera.

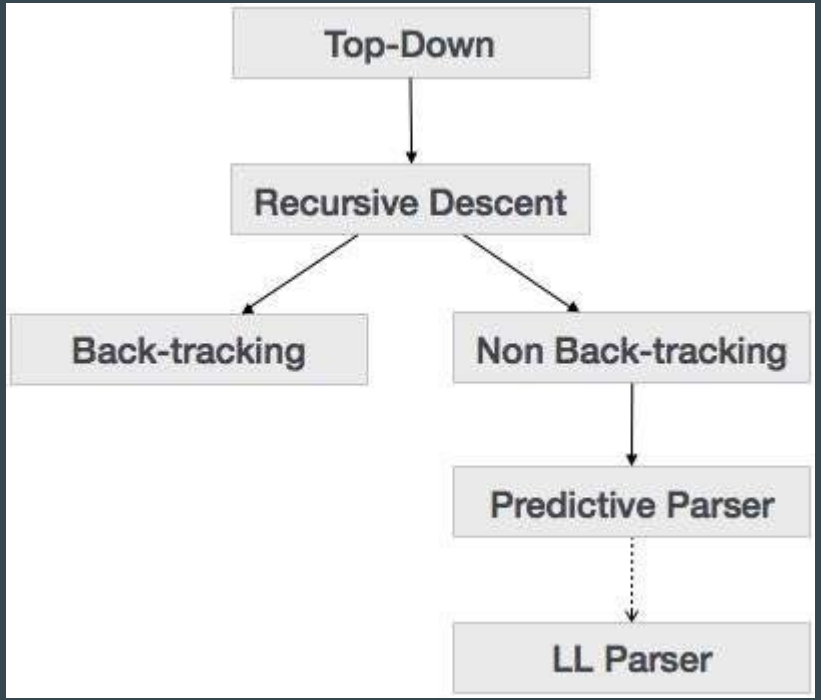
S	–>	AB
A	–>	aA | ε
B	–>	b | bB

S	
AB	S –> AB
aAB	A –> aA
aaAB	A –> aA
aaaAB	A –> aA
aaaεB	A –> ε
aaab	B –> b

aaab	
aaaεb	(insert ε)
aaaAb	A –> ε
aaAb	A –> aA
aAb	A –> aA
Ab	A –> aA
AB	B –> b
S	S –> AB

Técnicas TOP DOWN Parsing

Técnicas de Top Down parsing fazem o “parseamento” da entrada e construindo uma árvore de “parse” da raiz , movendo gradualmente para as folhas.

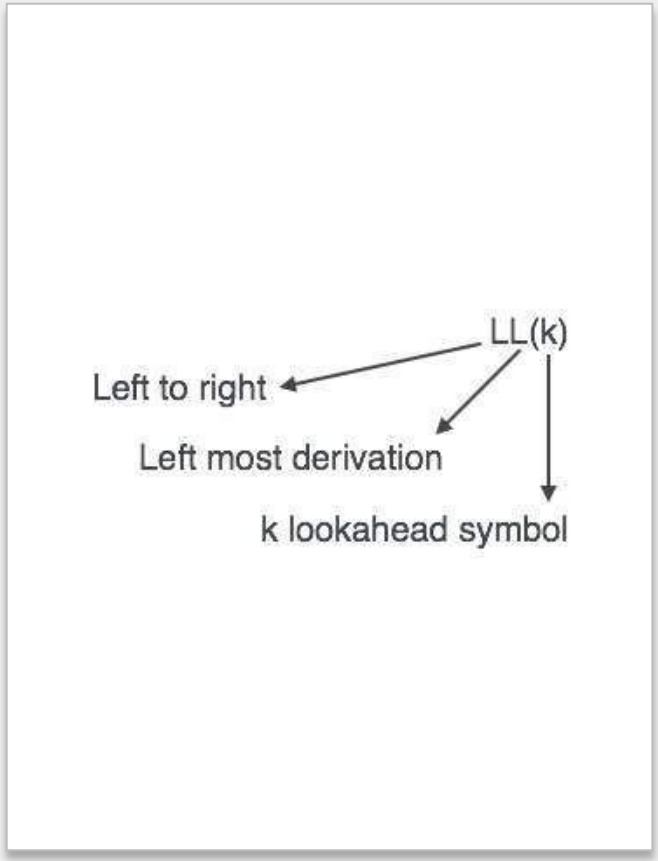


LL Parser

LL Parser aceita gramáticas do tipo LL que são subsets de gramáticas livres de contexto mas com algumas restrições a fim de termos uma implementação mais simples.

O parser LL é denotado como LL(k).

O Primeiro L significa a entrada da esquerda para a direita e o segundo L significa a derivação mais a esquerda e k representa o numero de “look aheads”.



Duas escolhas do parser

Predição

Tentar encontrar a formação que pode gerar o terminal mais à esquerda

Match

Identificar o terminal mais à esquerda da string

Exemplo

- S → E
- E → T + E
- E → T
- T → `int`

`int + int + int`

Production	Input	Action
S	int + int + int	Predict S -> E
E	int + int + int	Predict E -> T + E
T + E	int + int + int	Predict T -> int
int + E	int + int + int	Match int
+ E	+ int + int	Match +
E	int + int	Predict E -> T + E
T + E	int + int	Predict T -> int
int + E	int + int	Match int
+ E	+ int	Match +
E	int	Predict E -> T
T	int	Predict T -> int
int	int	Match int
		Accept

Principio do algoritmo

Basicamente é encontrar as formações que derivam nos terminais da sentença.

Humanamente é facil por tentativa e erro, mas a implementação do algoritmo pode tornar a busca das formações de forma mais rápidas.

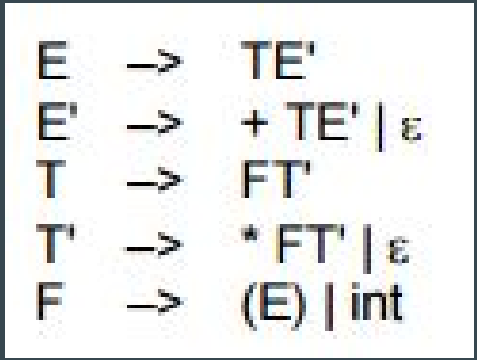
O algoritmo se baseia na criação de uma tabela que auxilie, baseado no terminal, identificar qual formação o gera usando uma pilha para guardar as variáveis da gramática.

Para formar a tabela é necessario identificar os FIRST and FOLLOW para cada derivação.

Construção da Tabela do Parse

FIRST

Quais os primeiros terminais ou epsilon alcançados pela formação



FOLLOW

Para um não terminal A, FOLLOW(A) é o conjuntos de terminais que podem aparecer a direita de A em alguma forma sentencial seguindo as regras:

- SE A é um simbolo inicial ,
  - coloque $ no FOLLOW(A)
- Produções $B \rightarrow \alpha A \beta$,
  - geram FOLLOW(A) = FIRST(β)
- Produções $B \rightarrow \alpha A \beta$ onde $\beta \rightarrow \varepsilon$
  - adiciona FOLLOW(A) = FOLLOW(B)

Identificando FIRST

FIRST(E) = FIRST{T} = FIRST(F) = { (, int }

FIRST(E´) = { + , ε }

FIRST(T´) = {*, ε }

E	–>	TE'
E'	–>	+ TE' | ε
T	–>	FT'
T'	–>	* FT' | ε
F	–>	(E) | int

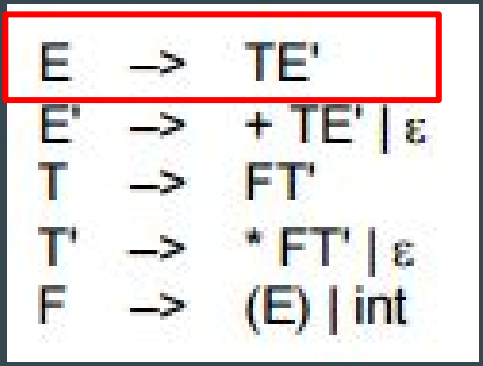
Identificando Follow

FOLLOW(E) = { }
FOLLOW(E´) = { }
FOLLOW(T) = { }
FOLLOW(T´) = { }
FOLLOW(F) = { }

E	–>	TE'
E'	–>	+ TE' | ε
T	–>	FT'
T'	–>	* FT' | ε
F	–>	(E) | int

Identificando Follow

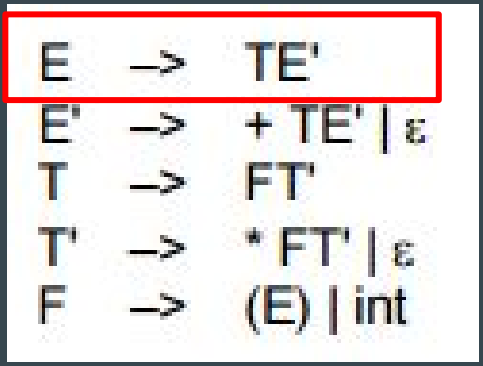
FOLLOW(E) = { $ }
FOLLOW(E´) = { }
FOLLOW(T) = { }
FOLLOW(T´) = { }
FOLLOW(F) = { }



Adiciona $ ao simbolo inicial E

Identificando Follow

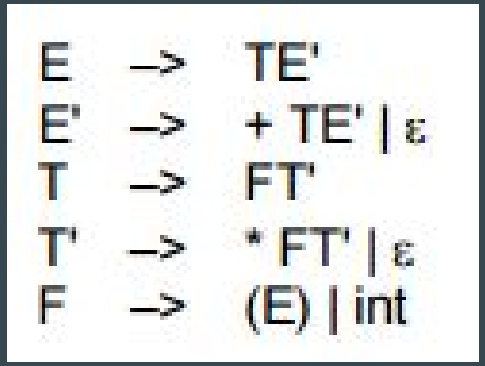
FOLLOW(E) = { $ }
FOLLOW(E´) = {FOLLOW(E) }
FOLLOW(T) = { }
FOLLOW(T´) = { }
FOLLOW(F) = { }



- Produções B → αAβ onde β → ε , adiciona FOLLOW(A) = FOLLOW(B)
  - E → TE´ onde
    - ↄ = T
    - A = E´
    - B = E
    - Portanto FOLLOW(E´) = FOLLOW(E)

Identificando Follow

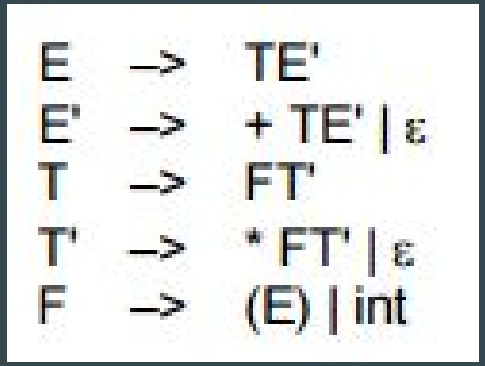
FOLLOW(E) = { $ }
FOLLOW(E´) = {FOLLOW(E) }
FOLLOW(T) = {+ }
FOLLOW(T´) = { }
FOLLOW(F) = { }



- Produções B → αAβ, geram FOLLOW(A) = FIRST(β)
  - E´ →+ TE´| ε , onde
    - Ↄ = +
    - A = T
    - β = E´
    - Portanto FOLLOWN(T) = FIRST(E´)

Identificando Follow

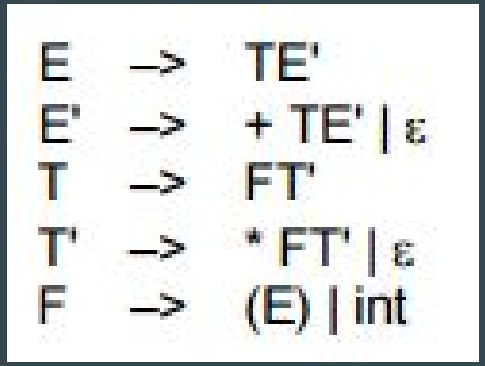
FOLLOW(E) = { $ }
FOLLOW(E´) = {FOLLOW(E) }
FOLLOW(T) = {+ }
FOLLOW(T´) = {FOLLOW(T) }
FOLLOW(F) = { }



- Produções B → αAβ onde β → ε , adiciona FOLLOW(A) = FOLLOW(B)
  - T →F T´ , onde
    - ↄ = F
    - A = T
    - Portanto FOLLOW(T) = FOLLWO(T´)

Identificando Follow

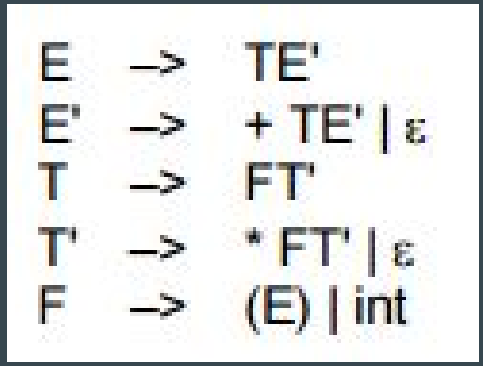
FOLLOW(E) = { $ }
FOLLOW(E′) = {FOLLOW(E) }
FOLLOW(T) = {+ }
FOLLOW(T′) = {FOLLOW(T) }
FOLLOW(F) = {* }



- Produções B → αAβ, geram FOLLOW(A) = FIRST(β)
  - T′ →*F T′| ε , onde
    - Ↄ = *
    - A = F
    - β = T′
    - Portanto FOLLOW(F) = FIRST(T′)

Identificando Follow

FOLLOW(E) = { $ } U {) }
FOLLOW(E´) = {FOLLOW(E) }
FOLLOW(T) = {+ }
FOLLOW(T´) = {FOLLOW(T) }
FOLLOW(F) = {* }



- Produções B → αAβ, geram FOLLOW(A) = FIRST(β)
  - F →(E) | int , onde
    - α = (
    - A = E
    - β =)
    - Portanto FOLLOW(E) = FIRST())

Identificando Follow

FOLLOW(E) = { $ } U {) }
FOLLOW(E´) = {FOLLOW(E) }
FOLLOW(T) = {+ } U FOLLOW(E´)
FOLLOW(T´) = {FOLLOW(T) }
FOLLOW(F) = {* }

E	–>	TE'
E'	–>	+ TE' | ε
T	–>	FT'
T'	–>	* FT' | ε
F	–>	(E) | int

- **Pensando na formação E´ → +TE´| ε ela gera a formação E´ → +T**
- Produções B → αAβ onde β → ε , adiciona FOLLOW(A) = FOLLOW(B)
  - E´ → +T , onde
    - ↄ = +
    - A = T
    - Portanto FOLLOW(T) = FOLLOW(E')

Identificando Follow

FOLLOW(E) = { $ } U {) }
FOLLOW(E´) = {FOLLOW(E) }
FOLLOW(T) = {+ } U FOLLOW(E´)
FOLLOW(T´) = {FOLLOW(T) }
FOLLOW(F) = {* } U FOLLOW(T´)

E	–>	TE'
E'	–>	+ TE' | ε
T	–>	FT'
T'	–>	* FT' | ε
F	–>	(E) | int

- **Pensando na formação T´ → *FT´| ε ela gera a formação T´ → *F**
- Produções B → αAβ onde β → ε , adiciona FOLLOW(A) = FOLLOW(B)
  - T´ → *F , onde
    - ↄ = *
    - A = F
    - Portanto FOLLOW(F) = FOLLOW(T')

Identificando Follow

FOLLOW(E) = { $,) }
FOLLOW(E´) = { $,) }
FOLLOW(T) = { +, $,) }
FOLLOW(T´) = { +, $,) }
FOLLOW(F) = { *,+, $,) }

E	–>	TE'
E'	–>	+ TE' | ε
T	–>	FT'
T'	–>	* FT' | ε
F	–>	(E) | int

FIRST E FOLLOWS

FIRST(E) = { (, int }
FIRST(E′) = { + , ε }
FIRST(T) = { (, int }
FIRST(T) = { * , ε }
FIRST(F) = { (, int }

FOLLOW(E) = { $,) }
FOLLOW(E′) = { $,) }
FOLLOW(T) = { +, $,) }
FOLLOW(T′) = { +, $,) }
FOLLOW(F) = { *,+, $,) }

Regras da Tabela

Para cada produção A->U faça os passos a seguir

Para cada terminal em First(U) adicione a formação de U na coluna do terminal

Se temos ε in First, adicionamos o arco A->ε nos terminais Follow

Tabela LL(1) construída

E	–>	TE'
E'	–>	+ TE' | ε
T	–>	FT'
T'	–>	* FT' | ε
F	–>	(E) | int

Input/ Top of parse stack	int	+	*	(	)	$
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–> int			F–> (E)		

Algoritmo

1. Se X=a e a é o fim da string , fim do parse com sucesso.
2. SE X=a e a é diferente do fim da string, pop X na pilha e avança para a próxima entrada.
3. Se X != a e não é um terminal, pop X e consulte a tabela para ver qual produção se aplica, empilha o lado direito da produção na pilha (predição).
4. Se nenhum dos casos acima se aplica ou não há uma entrada valida na tabela, tem um erro no parser

Aplicando o algoritmo com a tabela

PARSE STACK	REMAINING INPUT	PARSER ACTION
E$	int + int * int$	Predict E –> TE', pop E from stack, push TE', no change in input
TE'$	int + int * int$	Predict T–> FT'

Input/ Top of parse stack	int	+	*	(	)	$
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–> int			F–> (E)		

Aplicando o algoritmo com a tabela

FT'E'$	int + int * int$	Predict F–> int
intT'E'$	int + int * int$	Match int, pop from stack, move ahead in input
T'E'$	+ int * int$	Predict T'–>ε
E'$	+ int * int$	Predict E'–> +TE'
+TE'$	+ int * int$	Match +, pop
TE'$	int * int$	Predict T–>FT'
FT'E'$	int * int$	Predict F–> int

Input/ Top of parse stack	int	+	*	(	)	$
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–> int			F–> (E)		

Aplicando o algoritmo com a tabela

intT'E'$	int * int$	Match int, pop
T'E'$	* int$	Predict T'–> *FT'
*FT'E'$	* int$	Match *, pop
FT'E'$	int$	Predict F–> int
intT'E'$	int$	Match int, pop
T'E'$	$	Predict T'–> ε
E'$	$	Predict E'–> ε
$	$	Match $, pop, success!

Input/ Top of parse stack	int	+	*	(	)	$
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–> int			F–> (E)		

Referencias

http://dragonbook.stanford.edu/lecture-notes/Stanford-CS143/07-Top-Down-Parsing.pdf

http://blog.reverberate.org/2013/07/ll-and-lr-parsing-demystified.html

https://en.wikipedia.org/wiki/LL_parser

http://www.inf.ufrgs.br/~nicolas/pdf/Compiladores06-first-follow.pdf

https://www.youtube.com/watch?v=SBnjVW8dUqo

https://www.tutorialspoint.com/compiler_design/compiler_design_types_of_parsing.htm