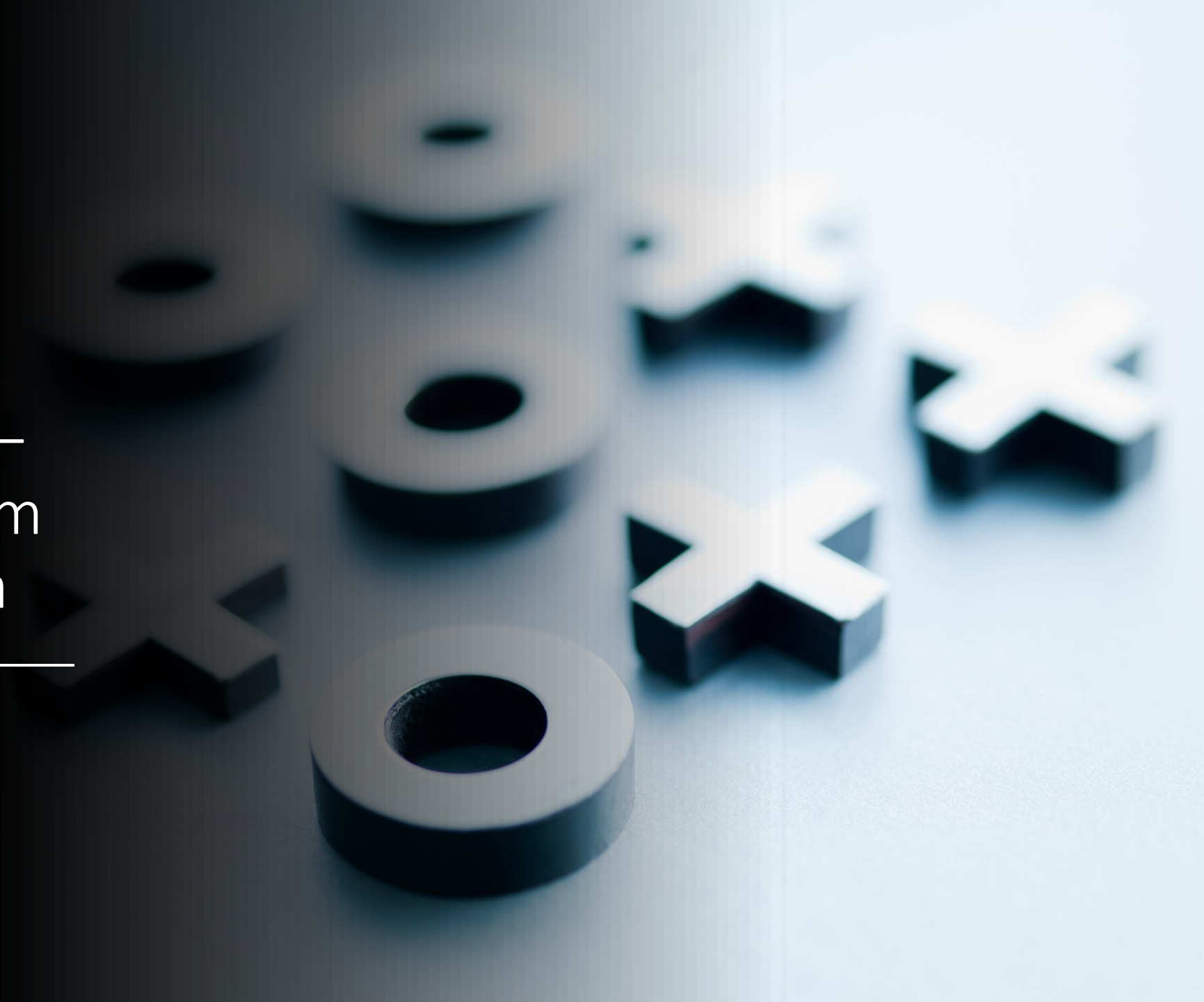
Análise sintática descendente preditiva. Parte II – análise baseada em autômato de pilha



Professor Eraldo Pereira Marinho

Compiladores 2º semestre 2020

UNESP/IGCE Rio Claro

Análise sintática LL(1) – implementação por autômato de pilha

Vimos até agora um tipo muito peculiar de análise LL(1), a denominada análise descendente preditiva recursiva, onde utilizamos procedimentos C para emular o processo de derivação esquerda. Vimos também que havia um conjunto de restrições sobre a gramática utilizada para este propósito. Ainda aqui estão valendo tais restrições:

1. Todas as produções não possuem recursão esquerda, ou seja, $\forall A \in N, A \not\Rightarrow^* A\alpha$;
2. Produções do tipo $A \rightarrow \alpha_1|\cdots|\alpha_n$ são unicamente determinadas por um só token, daí o número 1 no argumento do anacronismo LL(1), que decide qual lado direito será escolhido – a saber, LL(1) vem de ***L****eft-to-right scan,* ***L****eftmost parsing by* ***1*** *token* – ou seja, o primeiro L designa o sentido de varredura léxica, enquanto o segundo L designa a derivação esquerda;
3. Se houver produção do tipo $A \rightarrow \varepsilon$, esta será invocada na forma sentencial $\alpha.A\beta$ da seguinte forma, $\alpha.A\beta \Rightarrow \alpha\varepsilon.\beta$, se e somente se o operador $lookahead$ não pertencer a $\mathrm{FIRST}(A)$ mas pertencer a $\mathrm{FOLLOW}(A)$;

Análise sintática LL(1) – autômatos LL(1)

Nós vimos em Teoria da Computação uma maneira bastante genérica de transformar uma gramática livre do contexto, $G = (N, \Sigma, S, P)$, num autômato finito não determinístico de pilha, AFNP, que era um autômato de três estados,

$M = (\{q_0, q_1, q_2\}, \Gamma = \{\$\} \cup N \cup \Sigma, \Sigma, q_0, \$, \{q_2\}, \Delta)$, onde

$\Delta = \{(q_0, \varepsilon, \$, q_1, \$S), (q_1, \varepsilon, \$, q_2, \$)\} \cup \{(q_1, \varepsilon, A, q_1, \alpha^R) \mid A \in N, A \to \alpha\} \cup \{ (q_1, \boldsymbol{\sigma}, \boldsymbol{\sigma}, q_1, \varepsilon)\}$

O problema desse autômato finito de pilha (AFP) é que a escolha da produção $A \to \alpha$, que corresponde a substituir o topo da pilha, quando este é um não terminal A, pela reversa, α^R, da forma sentencial à direita α. Como saber qual é a forma a ser escolhida, uma vez que não determinismo de escolha do lado direito pode ocorrer?

A resposta é muito simples e já sabemos o porquê da aula assíncrona anterior. A gramática deve estar na forma LL(1). Assim, existe uma única transição pilha, $(q_1, \varepsilon, A, q_1, \alpha^R)$, uma vez que a escolha α^R em substituição do topo A é unicamente definido pelo token oculto no operador $lookahead$. Neste caso, o AFP torna-se um autômato semideterminístico de pilha.

Daqui em diante, chamaremos o AFP, para o caso de gramática LL(1), de autômato LL(1) ou simplesmente LL(1) parser.

Estudo de caso – gramática de expressões LL(1)

Vamos começar estudando uma situação peculiar antes de generalizarmos o conceito de análise sintática LL(1). Podemos fazer aquilo que fizemos em Teoria da Computação, com o AFNP de três estados, utilizando a gramática

$$E \rightarrow TR$$
$$T \rightarrow FQ$$
$$R \rightarrow \oplus\, TR \mid \varepsilon$$
$$Q \rightarrow \otimes\, FQ \mid \varepsilon$$
$$F \rightarrow (E) \mid \boldsymbol{n}$$

onde $\oplus \in \{+, -, \mathrm{OR}\}$ e $\otimes \in \{*,\ /, \mathrm{AND}\}$ abstraem os operadores aditivos e multiplicativos, respectivamente.

Estudo de caso – gramática de expressões LL(1) – continuação

O AFNP é o usual de três estados, visto no início dessa aula, cujas transições com pilha são dadas por

$$\Delta = \{(q_0, \varepsilon, \$, q_1, \$E), (q_1, \varepsilon, \$, q_2, \$)\} \cup \{(q_1, \varepsilon, A, q_1, \alpha^R) | \ A \in N, A \to \alpha\} \cup \{ (q_1, \boldsymbol{\sigma}, \boldsymbol{\sigma}, q_1, \varepsilon)\}$$

$F = \{q_2\}$, $N = \{E, F, Q, R, T\}$, $\Sigma = \{\oplus, \otimes, (,), \boldsymbol{n}\}$, onde as produções recursivas em q_1, sem consumo de tokens, são

$$\{(q_1, \varepsilon, E, q_1, RT),$$
$$(q_1, \varepsilon, T, q_1, QF),$$
$$(q_1, \varepsilon, R, q_1, RT \oplus), (q_1, \varepsilon, R, q_1, \varepsilon),$$
$$(q_1, \varepsilon, Q, q_1, QF \otimes), (q_1, \varepsilon, Q, q_1, \varepsilon),$$
$$(q_1, \varepsilon, F, q_1, \text{"})E(\text{"}), (q_1, \varepsilon, F, q_1, \boldsymbol{n})\}$$

As operações de pilha que correspondem ao operador match são aquelas em que o topo da pilha bate (matches) com o token avistado em $lookahead$:

$$\{(q_1, \boldsymbol{n}, \boldsymbol{n}, q_1, \varepsilon), (q_1, (, (, q_1, \varepsilon), (q_1,),), q_1, \varepsilon),$$
$$(q_1, \oplus, \oplus, q_1, \varepsilon), (q_1, \otimes, \otimes, q_1, \varepsilon)\}$$

Observem que as transições não épsilon seguem o padrão $(q_1, lookahead, top, q_1, \varepsilon) | \ lookahead == top$, se a igualdade relacional não ocorrer, isso implica erro de sintaxe, ou simplesmente *token mismatch*.

Estudo de caso – gramática de expressões LL(1) – continuação

Uma forma prática de denotarmos as potenciais transições épsilon, ou seja, quando o topo da pilha é um não terminal, se vale da seguinte tabela, denominada tabela gramatical, para traduzir do inglês “parse table”:

Topo da pilha \ $lookahead$	$\boldsymbol{n}$	(	)	$\oplus$	$\otimes$	\$
E	RT	RT	ERR	ERR	ERR	ERR
T	QF	QF	ERR	ERR	ERR	ERR
R	ERR	ERR	ε	$RT\oplus$	ERR	ε
Q	ERR	ERR	ε	ε	$QF\otimes$	ε
F	$\boldsymbol{n}$	$)E($	ERR	ERR	ERR	ERR

Essa tabela será consultada a cada transição cujo topo da pilha é um não terminal, para escolher, em função de $lookahead$, apontando para alguma coluna da tabela, qual forma sentencial irá substituir esse não terminal.

Estudo de caso – gramática de expressões LL(1) – continuação

A tabela anterior foi preenchida com base nas funções FIRST e FOLLOW. As células com erro correspondem a situações proibidas, oriundas de um erro sintático. Para reforçar, a células com palavra nula, ε, somente se e somente ocorrem nas colunas correspondentes a FOLLOW, cruzando com a linha do não terminal que possui produção nula, $A \rightarrow \varepsilon$.

Algoritmo de preenchimento da tabela gramatical

Entrada: $G = (N, \Sigma, S, P)$ na forma LL(1)

Saída: Uma tabela gramatical, $TG \subseteq N \times \Sigma \cup \{\$\}$, correspondente às regras de produção válidas para não terminais no topo da pilha

Método:

01: Para cada não terminal $X \in N$, associe uma linha na tabela TG
02: Para cada terminal $\tau \in \Sigma \cup \{\$\}$, associe uma coluna de TG
03: Para cada linha $X \in N$, faça:
04: Para cada coluna $\tau \in \Sigma \cup \{\$\}$, faça:
05: Se $\tau \in \text{FIRST}(X)$, então
06: $TG[X][\tau] \leftarrow {\alpha_\tau}^R$, para $X \rightarrow \alpha_\tau$
07: Senão,
08: Se $\exists\, X \rightarrow \epsilon \in P$, então
09: Se $\tau \in \text{FOLLOW}(X)$, então
10: $TG[X][\tau] \leftarrow \varepsilon$
11: Senão,
12: $TG[X][\tau] \leftarrow ERR$
13: Fim de se
14: Senão,
15: $TG[X][\tau] \leftarrow ERR$
16: Fim de se
17: Fim de se
18: Fim de faça para cada τ
19: Fim de faça para cada X

Análise LL(1) – Exemplo 1

Tomemos como exemplo a entrada $\boldsymbol{n} \oplus \boldsymbol{n}$

Como ficaria a tabela para entrada $\boldsymbol{n} \oplus \boldsymbol{n} \otimes (\boldsymbol{n} \oplus \boldsymbol{n})$?

Que acontece para a entrada $\boldsymbol{n} \oplus$?

Passo	Entrada	Lookahead	Pilha	Topo	Produção
0	$.\boldsymbol{n} \oplus \boldsymbol{n}$	$\boldsymbol{n}$	$\$$	$\$$	$\$ \rightarrow \$E$
1	$.\boldsymbol{n} \oplus \boldsymbol{n}$	$\boldsymbol{n}$	$\$E$	E	$E \rightarrow TR$
2	$.\boldsymbol{n} \oplus \boldsymbol{n}$	$\boldsymbol{n}$	$\$RT$	T	$T \rightarrow FQ$
3	$.\boldsymbol{n} \oplus \boldsymbol{n}$	$\boldsymbol{n}$	$\$RQF$	F	$F \rightarrow \boldsymbol{n}$
4	$.\boldsymbol{n} \oplus \boldsymbol{n}$	$\boldsymbol{n}$	$\$RQ\boldsymbol{n}$	$\boldsymbol{n}$	**match**
5	$\boldsymbol{n}.\oplus \boldsymbol{n}$	$\oplus$	$\$RQ$	Q	$Q \rightarrow \varepsilon$
6	$\boldsymbol{n}.\oplus \boldsymbol{n}$	$\oplus$	$\$R$	R	$R \rightarrow \oplus TR$
7	$\boldsymbol{n}.\oplus \boldsymbol{n}$	$\oplus$	$\$RT \oplus$	$\oplus$	**match**
8	$\boldsymbol{n} \oplus.\boldsymbol{n}$	$\boldsymbol{n}$	$\$RT$	T	$T \rightarrow FQ$
9	$\boldsymbol{n} \oplus.\boldsymbol{n}$	$\boldsymbol{n}$	$\$RQF$	F	$F \rightarrow \boldsymbol{n}$
10	$\boldsymbol{n} \oplus.\boldsymbol{n}$	$\boldsymbol{n}$	$\$RQ\boldsymbol{n}$	$\boldsymbol{n}$	**match**
11	$\boldsymbol{n} \oplus \boldsymbol{n}.$	$\$$	$\$RQ$	Q	$Q \rightarrow \varepsilon$
12	$\boldsymbol{n} \oplus \boldsymbol{n}.$	$\$$	$\$R$	R	$R \rightarrow \varepsilon$
13	$\boldsymbol{n} \oplus \boldsymbol{n}.$	$\$$	$\$$	$\$$	**end**

Análise LL(1) – Exemplo 2

Seja a gramática $S \rightarrow \boldsymbol{a}SR \mid \boldsymbol{c},\ R \rightarrow \boldsymbol{b}S \mid \varepsilon$

Facilmente, temos $\mathrm{FIRST}(S) = \{\boldsymbol{a}, \boldsymbol{c}\}, \mathrm{FIRST}(R) = \{\boldsymbol{b}, \varepsilon\}$, $\mathrm{FOLLOW}(R) = \{\$\}$, donde se deriva a tabela

	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{c}$	$\$$
S	$RS\boldsymbol{a}$	Erro	$\boldsymbol{c}$	Erro
R	Erro	$S\boldsymbol{b}$	Erro	ε

Exercitemos para a entrada $\boldsymbol{aacbc}$

Passo	Entrada	Lookahead	Pilha	Topo	Produção
1	$.\boldsymbol{aacbc}$	$\boldsymbol{a}$	$\$S$	S	$S \rightarrow \boldsymbol{a}SR$
2	$.\boldsymbol{aacbc}$	$\boldsymbol{a}$	$\$RS\boldsymbol{a}$	$\boldsymbol{a}$	**match**
3	$\boldsymbol{a}.\boldsymbol{acbc}$	$\boldsymbol{a}$	$\$RS$	S	$S \rightarrow \boldsymbol{a}SR$
4	$\boldsymbol{a}.\boldsymbol{acbc}$	$\boldsymbol{a}$	$\$RRS\boldsymbol{a}$	$\boldsymbol{a}$	**match**
5	$\boldsymbol{aa}.\boldsymbol{cbc}$	$\boldsymbol{c}$	$\$RRS$	S	$S \rightarrow \boldsymbol{c}$
6	$\boldsymbol{aa}.\boldsymbol{cbc}$	$\boldsymbol{c}$	$\$RR\boldsymbol{c}$	$\boldsymbol{c}$	**match**
7	$\boldsymbol{aac}.\boldsymbol{bc}$	$\boldsymbol{b}$	$\$RR$	R	$R \rightarrow \boldsymbol{b}S$

Análise LL(1) – Exemplo 2 – continuação

Passo	Entrada	Lookahead	Pilha	Topo	Produção
7	$\boldsymbol{aac.bc}$	$\boldsymbol{b}$	$\$RR$	R	$R \rightarrow \boldsymbol{b}S$
8	$\boldsymbol{aac.bc}$	$\boldsymbol{b}$	$\$RS\boldsymbol{b}$	$\boldsymbol{b}$	**match**
9	$\boldsymbol{aacb.c}$	$\boldsymbol{c}$	$\$RS$	S	$S \rightarrow \boldsymbol{c}$
10	$\boldsymbol{aacb.c}$	$\boldsymbol{c}$	$\$R\boldsymbol{c}$	$\boldsymbol{c}$	**match**
11	$\boldsymbol{aacbc.}$	$\$$	$\$R$	R	$R \rightarrow \varepsilon$
12	$\boldsymbol{aacbc.}$	$\$$	$\$$	$\$$	**end**

Análise LL(1) – sumário

Vimos que a análise sintática LL(1) baseia-se na utilização do autômato de pilha para o caso específico de emular uma gramática LL(1), aquela que não possui recursão esquerda e existe uma forma única de decidir quando uma produção épsilon deve ser utilizada.

A análise em si consiste de duas etapas. A primeira é o preenchimento de uma tabela gramatical a partir das regras de produção necessárias para a substituição do topo da pilha, de forma reversa, quando este é um não terminal.

A segunda etapa é a utilização do autômato propriamente dito. Se o topo da pilha é um terminal, é porque esse terminal é esperado no operador $lookahead$. Neste caso, se $lookahead$ bate com o topo da pilha, é feita a operação pop e o autômato pede um próximo $lookahead$.

As operações do analisador que não consomem token são aquelas correspondentes ao topo da pilha sendo um não terminal. Neste caso, reitero, a transição consiste da substituição do topo pelo respectivo lado direito (único) da produção, em função do operador $lookahead$, que é associado ao índice da coluna da tabela gramatica.

Apesar de não ter sido discutido aqui, a análise LL(1) é equivalente à sua versão recursiva, o que decorre do mecanismo de ativação de funções C e processos Unix, e isso será discutido em aula síncrona.

Questões valendo 2 horas

1. Que vem a ser um analisador sintático LL(1)?
2. Qual a finalidade da tabela gramatical?
3. Para que transições o autômato LL(1) recorre à tabela?
4. Sob que circunstâncias a pilha do autômato LL(1) sofre efetivamente uma operação pop?