Revisão P2
Linguagens Formais e Autômatos

Erickson G. Müller

Conteúdos

- Minimização de AF
- Árvores de Derviação
- Máquinas Reconhecedoras

1 Minimização de Autômatos Finitos

Um autômato finito é mínimo se não possui **estados inacessíveis, mortos e equivalentes**.
Considerando o seguinte AFND, vamos reconhecer esses estados

	a	b
$\rightarrow *A$	G	B
B	F	E
C	C	G
$*D$	A	H
E	E	A
F	B	C
$*G$	G	F
H	H	D

$A \rightarrow \{G, B, F, E, C, A$ *acess*
$B \rightarrow \{F, E, C, A, B, G$
$C \rightarrow \{C, G, B, F, E, A$
$D \rightarrow \{A, H, G, B, F, E, C, D$
$E \rightarrow \{E, A, G, B, F, C$
$F \rightarrow \{B, C, F, E, A, G$
$G \rightarrow \{G, F, B, C, E, A$
$H \rightarrow \{H, D, A, C, G, B, F, E$

O algoritmo para encontrar os estados inacessíveis começa com uma lista-chave de todos os estados seguidos dos estados que são possíveis de ser alcançado através daquele. Exemplo $A \rightarrow \{G, B$; Todos os estados que estão à direita compartilham o estados que se podem acessar com o estado à esquerda.

A segunda parte do algoritmo consiste em adicionar esses estados que podem ser acessados em duas transições através dos estados que podem ser acessados diretamente. Exemplo $A \rightarrow \{G, B, F, E$, e assim em ciclo até completar todos os estados.

Por fim, temos que os estados inalcançáveis são D e H.

Os estados que são mortos são aqueles que não alcançam, por nenhum fecho transitivo, um estado final. Portanto são os estados

Os estados são equivalentes quando, para cada símbolo, levam ao mesmo tipo de estados. Por exemplo, o A e o G são equivalentes pois G e G são estados finais e B e F são estados não finais.

F	$K - F$
A, G	B, C, E, F
$[A, G]$	$[B, C], [C, E]$

	a	b
$\rightarrow *[AG]$	$[AG]$	$[BF]$
$[BF]$	$[BF]$	$[CF]$
$[CE]$	$[CE]$	$[AG]$

1.1 Exercícios de minimização

2 Árvores de Derivação

Gramática de operadores:

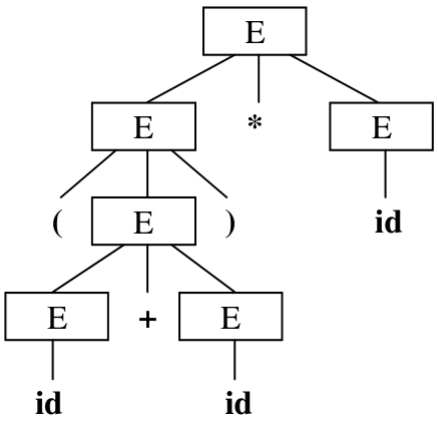
$$E ::= E + E | E * E | (E) | a$$

Árvores ilustram o processo de derivação a partir do símbolo inicial para construir uma gramática.

2.1 Reconhecimento Sintático

- Ascendente mais à esquerda
- Ascendente mais à direita
- Descendente mais à esquerda
- Descendente mais à direita

Segue exemplo de uma construção de árvore por abordagem descendente mais à direita para a construção de $a + a * a$. Perceba que a sentença $a + a * a$ pode ser formada pelos nós-folhas da árvore.



Em uma abordagem ascendente, a maquina reconhecedora vai identificar qual é a redução que pode ser feita, ao contrário da descendente que verifica a produção por derivação. Por exemplo, na ascendente mais à esquerda começaríamos com a sentença $a + a * a$. No primeiro símbolos podemos reduzir a produção a em uma produção E.

Se uma gramática G tem mais de uma árvore de derivação para uma mesma sentença, então G é chamada de gramática **ambígua**. Uma linguagem livre de contexto para a qual toda gramática livre de contexto é ambígua é denominada linguagem livre de contexto inerentemente ambígua.

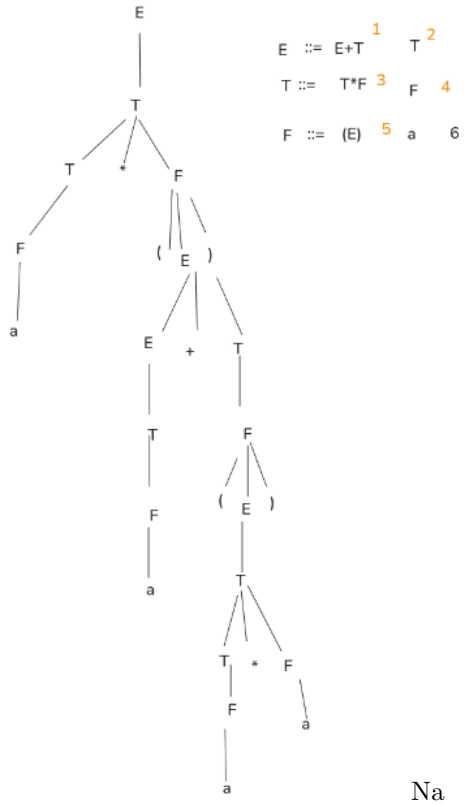
A ordem de produções por derivação mais à esquerda é o contrário da ordem de produções por redução mais à direita, e o mesmo se aplica ao contrário.

Exemplo:
Usando uma gramática de operadores que não é ambígua:
$E ::= E + T|T$
$T ::= T * F|F$
$F ::= (E)|a$

$$a * (a + (a * a))$$

Sequência de montagem da árvore.

1. **Descendente mais à esquerda:** 234651246452346
2. **Descendente mais à direita:** 2351452364624646
3. **Ascendente mais à esquerda:** 6464264632541532
4. **Ascendente mais à direita:** 643254642156432



Na ascendente, a árvore fica idêntica apenas com os nós-folhas alinhados no último nível da altura da árvore. Perceba como a sentença $a * (a + (a * a))$ está presente nos nós-folhas dessa árvore.

3 Eliminação de símbolos inúteis (improdutivos e inalcançáveis)

$S ::= ASB|BSA|SS|aS|\varepsilon$
$A ::= ABS|B$
$B ::= BSSA|A$

A primeira produção produtiva (que possui apenas símbolos terminais) é o ε, Logo o S é uma produção produtiva. Em seguida vamos marcar todos os símbolos terminais e o não-terminal S em todas as produções. As produções que estão totalmente marcadas (SS) são também produtivas.

As produções que não estão totalmente marcadas não são produções produtivas, portanto podem ser descartadas: $ASB, BSA, ABS, B, BSSA, A$. Assim a gramática reduzida é $S ::= SS|aS|\varepsilon$

$S ::= aS|SB|SS|\varepsilon$
$A ::= ASB|c$
$B ::= b$ Marcamos os terminais ε, a, c, b. Percebemos que S, A, B são produções produtivas, portanto podemos marcar todas as produções. Concluindo que não existem produções improdutivas.

Exemplo:
$S ::= aSa|FbD$
$D ::= Dd|fF|c$
$F ::= aA|CA|\varepsilon$
$E ::= BC|eE|EB$
$B ::= bB|FE$
$F ::= fF|Dd$
$C ::= cCb|AcA$

Primeiro passo, marcamos as produções produtivas (ε, c). Percebemos que D e F são produtivos. Em seguida vamos marcar todos os símbolos terminais e os não-terminais produtivos.

$S ::= \mathbf{a}S\mathbf{a}|\mathbf{FbD}$
$\mathbf{D} ::= \mathbf{Dd}|\mathbf{fF}|\mathbf{c}$
$\mathbf{A} ::= \mathbf{aA}|C\mathbf{A}| * \varepsilon *$
$E ::= BC|\mathbf{e}E|EB$
$B ::= \mathbf{b}B|\mathbf{F}E$
$\mathbf{F} ::= \mathbf{fF}|\mathbf{Dd}$
$C ::= \mathbf{cCb}|\mathbf{AcA}$

$\mathbf{S} ::= \mathbf{aSa}|\mathbf{FbD}$
$\mathbf{D} ::= \mathbf{Dd}|\mathbf{fF}|\mathbf{c}$
$\mathbf{A} ::= \mathbf{aA}|C\mathbf{A}| * \varepsilon *$
$E ::= BC|\mathbf{e}E|EB$
$B ::= \mathbf{b}B|\mathbf{F}E$
$\mathbf{F} ::= \mathbf{fF}|\mathbf{Dd}$
$\mathbf{C} ::= \mathbf{cCb}|\mathbf{AcA}$

Logo, E e B não são produtivos.
S ::= **aSa**|**FbD**
D ::= **Dd**|**fF**|**c**
A ::= **aA**|C**A**| $*\varepsilon*$
F ::= **fF**|**Dd**
C ::= **cCb**|**AcA**
Agora resta-nos encontrar os alcançáveis.
$S \rightarrow \{S, F, D$
$D \rightarrow \{D, F$
$A \rightarrow \{A, C$
$F \rightarrow \{F, D$
$C \rightarrow \{C, A$
Assim S, F e D são as únicas que têm produções alcancáveis e produtivas,
S ::= **aSa**|**FbD**
D ::= **Dd**|**fF**|**c**
F ::= **fF**|**Dd**

4 Fatoração

Eliminamos primeiro os indeterminismos diretos para depois eliminar os indeterminismos indiretos.

Uma GLC está fatorada se ela é determinística, ou seja, se ela não possui produções para um mesmo não-terminal no lado esquerdo cujo lado direito inicie com a mesma cadeia de símbolos ou com símbolos que derivam seqüências que iniciem com a mesma cadeia de símbolos. De acordo com esta definição, a seguinte gramática é não-determinística:
$S ::= AC|BC$
$A ::= aD|cC$
$B ::= aB|dD$
$C ::= eC|eA$
$D ::= fD|AB$
Fatorando o C:
$S ::= AC|BC$
$A ::= aD|cC$
$B ::= aB|dD$
C::= eC'
C'::= C—A
$D ::= fD|AB$
Dá pra fatorar também a produção S, pois começam com não terminal (não-determinismo indireto):
1º Passo:
$S ::=$ **aD**C|**cC**C|**aB**C|**dD**C
$C ::= eC'$
$C' ::= C|A$
$D ::= fD|AB$

2º passo:
$S ::= aS'|cCC|dDC$
$S' ::= DC|BC$
Exemplo:
$S := Abc|bBC|bCD$
$A ::= aBC|aDC$
$B ::= dCc|dc$
$C ::= Acd|dc$
$D ::= aBC|abc|aC$
Mudamos a produção S, A, B e D:
$S ::= Abc|bS'$
$S' ::= BC|CD$
$A ::= aA'$
$A' ::= BC|DC$
$B ::= dB'$
$B' ::= Cc|c$
$C ::= Acd|dc$
$D ::= aD'$
$D' ::= BC|bc|C$
Agora retiramos o não-determinismo indireto:
$S ::= Abc|bS'$
$S' ::= dB'C|AcdD|dcD$
$A ::= aA'$
$A' ::= BC|DC$
$B ::= dB'$
$B' ::= Cc|c$
$C ::= Acd|dc$
$D ::= aD'$
$D' ::= BC|bc|C$
2° passo:
$S ::= Abc|bS'$
$S' ::= dS''|AcdD$
$S'' ::= B'C|cD$
$A ::= aA'$
$A' ::= BC|DC$
$B ::= dB'$
$B' ::= Cc|c$
$C ::= Acd|dc$
$D ::= aD'$
Substituimos o D:
$S ::= Abc|bS'$
$S' ::= dS''|AcdD$
$S'' ::= B'C|cD$
$A ::= aA'$
$A' ::= BC|DC$
$B ::= dB'$

$B' ::= Cc|c$
$C ::= Acd|dc$
$D ::= aD'$
$D' ::= dB'C|bc|Acd|dc$
Novamente:
$S ::= Abc|bS'$
$S' ::= dS''|AcdD$
$S'' ::= B'C|cD$
$A ::= aA'$
$A' ::= BC|DC$
$B ::= dB'$
$B' ::= Cc|c$
$C ::= Acd|dc$
$D ::= aD'$
$D' ::= dD''|bc|Acd$
$D'' ::= B'C|c$
Produções B e S podem começar com c e podem ser fatoradas mais ainda.

5 Recursão à esquerda

Acontece quando uma produção possui uma regra que começa com um não-terminal que é o próprio símbolo da produção
Ex:

$$S ::= Aab|Bc|ScAB$$

$$A ::= SAc|BaA|ab$$

$$B ::= Ac|aBb|ab$$

Para eliminar essa recursão à esquerda, mantendo a gramática com a mesma capacidade gerativa. Começamos com a produção que não é recursiva (como Bc), e criamos uma nova produção S':

$$S ::= AabS'|BcS'$$

$$S' ::= cAbS'|\varepsilon$$

Substituímos as produções que possuem S pelas produções de S' (em A):

$$A ::= AabS'Ac|BcS'Ac|BaA|ab$$

Agora podemos eliminar a recursão à esquerda de A:

$$A ::= BcS'AcA'|BaAA'|abA'$$

$$A' :: -abS'AcA'|\varepsilon$$

Não sei o que é isso:
$B ::= \mathbf{B}cS'AcA'c|\mathbf{B}aA'Ac|abAc|aBb|ab$ *Professor riscou essa produção*

$B ::= abAcB'|aBbB'|abB'$
$B' ::= cS'AcA'cB'|aAA'cB'|\varepsilon$

Exemplo:

$S ::= SaB|Sbc|Acd$
$A ::= ABc|Acd|Bcd|Cd$
$B ::= Acd|Bcc|Cde$
$C ::= Ac|Cd|dc$

Primeiro em S:

$S ::= AcdS'$
$S' ::= aBS'|bcS'|AcdS'|\varepsilon$
$A ::= ABc|Acd|Bcd|Cd$
$B ::= Acd|Bcc|Cde$
$C ::= Ac|Cd|dc$

Agora em A:

$S ::= AcS'$
$S' ::= aBS'|bcS'|AcdS'|\varepsilon$
$A ::= BcdA'|CdA'$
$A' ::= BcA'|cdA'|\varepsilon$
$B ::= Acd|Bcc|Cde$
$C ::= Ac|Cd|dc$

Agora em B:

$S ::= AcS'$
$S' ::= aBS'|bcS'|AcdS'|\varepsilon$
$A ::= BcdA'|CdA'$
$A' ::= BcA'|cdA'|\varepsilon$
$B ::= AcdB'|CdeB'$
$B' ::= ccB'|\varepsilon$
$C ::= Ac|Cd|dc$

Agora em C:

$S ::= AcS'$
$S' ::= aBS'|bcS'|AcdS'|\varepsilon$
$A ::= BcdA'|CdA'$
$A' ::= BcA'|cdA'|\varepsilon$
$B ::= AcdB'|CdeB'$
$B' ::= ccB'|\varepsilon$
$C ::= AcC'|dcC'$
$C' ::= dC'|\varepsilon$