Linguagens Formais e Autômatos

Prof. Sergio D Zorzo
Departamento de Computação - UFSCar

1º semestre / 2017

Aula 10

Ambiguidade

Ambiguidade

- Considere as seguintes frases (verídicas), extraídas de um sistema de pedidos de um almoxarifado de um banco
  - “Armário para funcionário de aço”
  - “Cadeira para gerente sem braços”
- Quem é de aço? O armário ou funcionário?
- Quem não tem braços? A cadeira ou o gerente?
- O problema é a ambiguidade

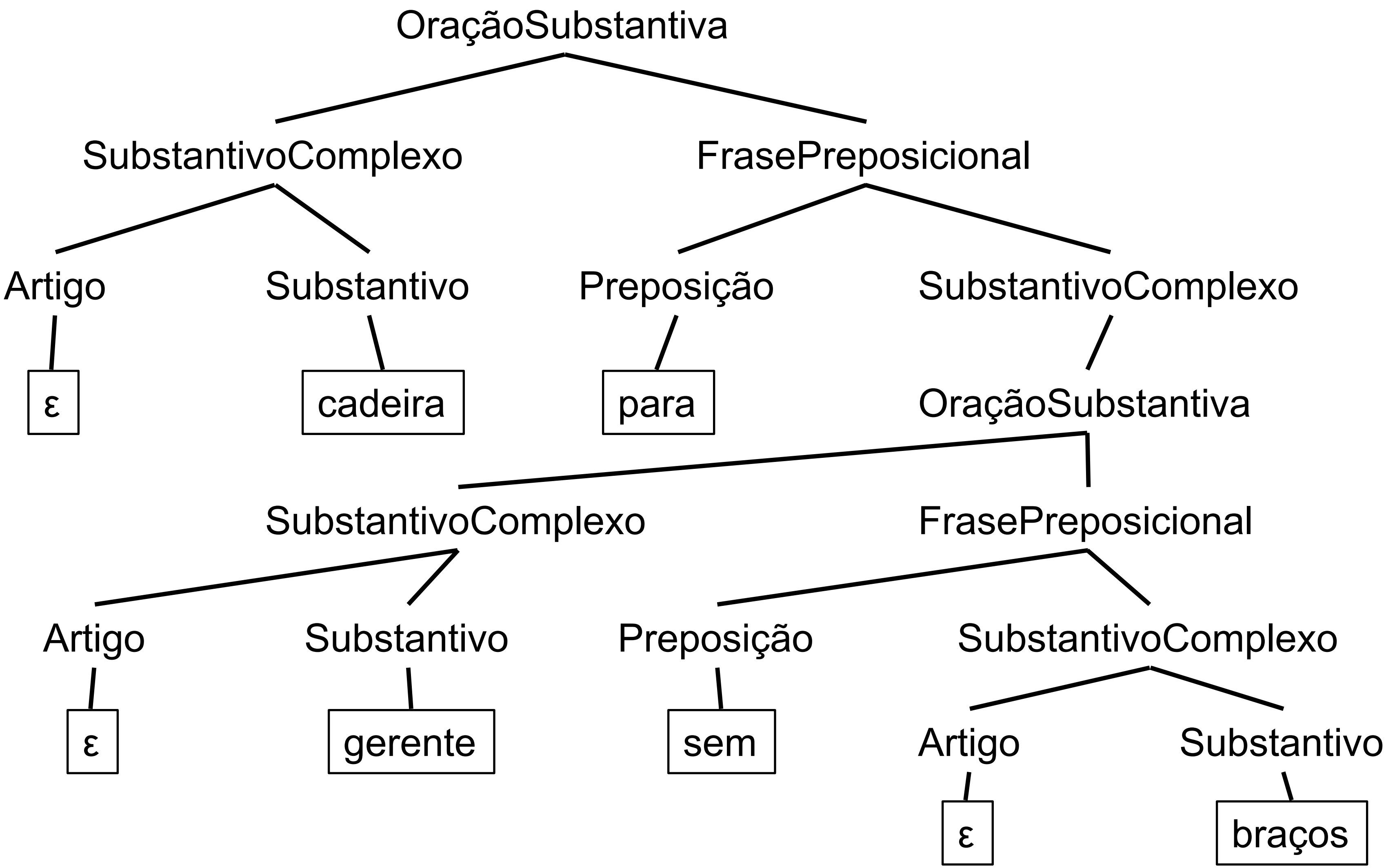
Ambiguidade

- Gramática da língua portuguesa (trecho)

```
OraçãoSubstantiva → SubstantivoComplexo
                  |  SubstantivoComplexo FrasePreposicional ;
FraseVerbal → VerboComplexo
            |   VerboComplexo FrasePreposicional ;
FrasePreposicional → Preposição SubstantivoComplexo ;
SubstantivoComplexo → Artigo Substantivo | OraçãoSubstantiva ;
VerboComplexo → Verbo | Verbo OraçãoSubstantiva ;
Artigo → o | a | um | uma | ε ;
Substantivo → armário | funcionário | gerente | cadeira | braços |
  aço ;
Verbo → gosta | brinca | olha ;
Preposição → para | com | de | sem ;
```

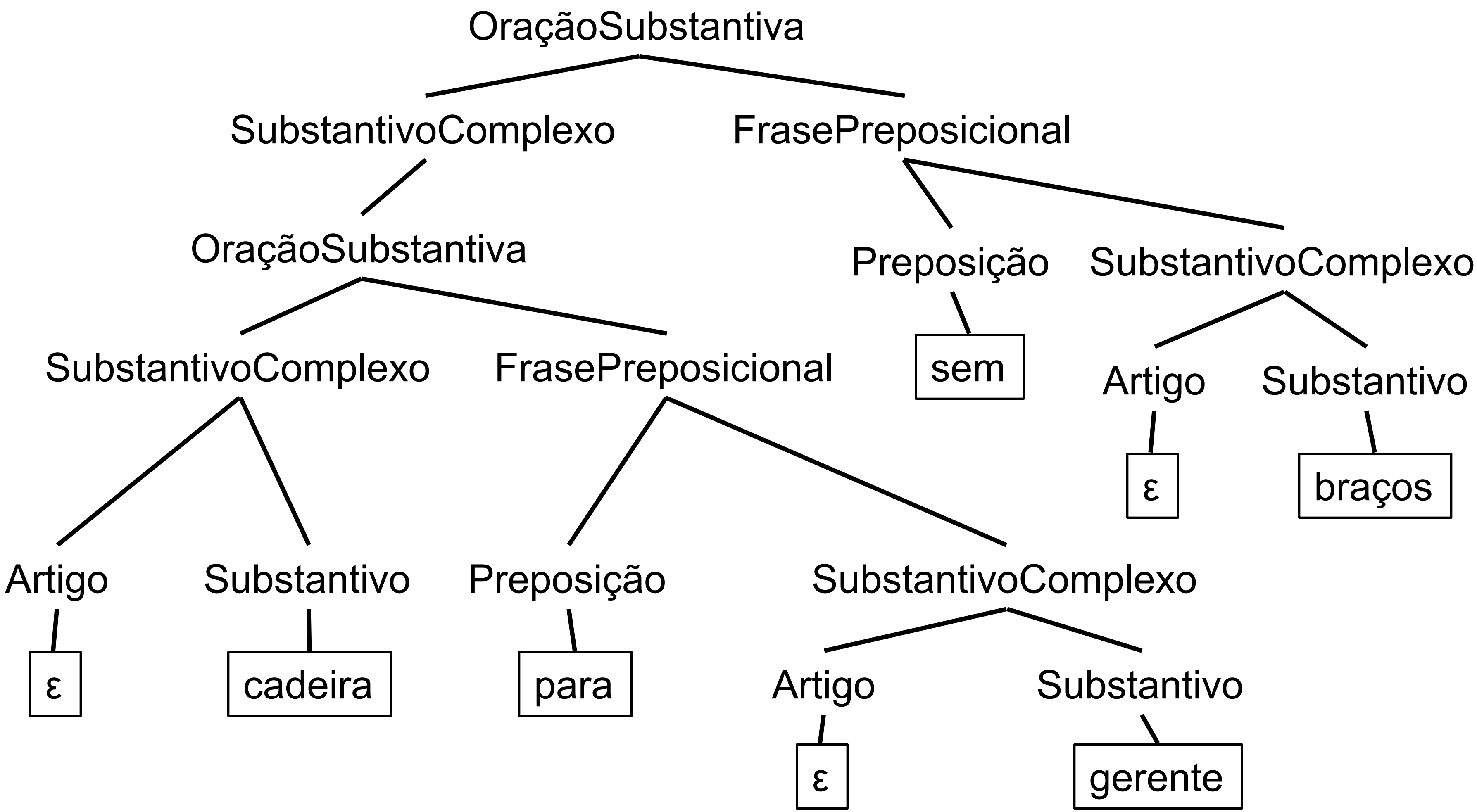
Ambiguidade

- Análise sintática da frase (1): cadeira para gerente sem braços



Ambiguidade

- Análise sintática da frase (2): cadeira para gerente sem braços



Ambiguidade

- Outro exemplo, gramática à direita
  - Encontre derivações mais à esquerda para a cadeia a + b*a
- Respostas:
- E ⇒ E + E ⇒ I + E ⇒ a + E ⇒ a + E * E ⇒ a + I * E ⇒ a + b * E ⇒ a + b * I ⇒ a + b * a

- E ⇒ E * E ⇒ E + E * E ⇒ I + E * E ⇒ a + E * E ⇒ a + I * E ⇒ a + b * E ⇒ a + b * I ⇒ a + b * a

E → I
E → E + E
E → E * E
E → (E)
I → a
I → b
I → Ia
I → Ib
I → I0
I → I1

Ambiguidade

- A diferença entre as árvores e as derivações mais à esquerda é significativa
  - Dependendo de qual árvore usar, o gerente pode ficar sem braços
    - Cadeira para ___
    - ___ sem braços
  - Dependendo de qual derivação à esquerda usar, a adição pode ocorrer antes da multiplicação
    - a + ___
    - ___ * a

Ambiguidade

- Gramáticas são usadas para dar estrutura para programas, documentos, etc
  - Supõe-se que essa estrutura é única
  - Caso não seja, podem ocorrer problemas
- Nem toda gramática fornece estruturas únicas
  - Ambiguidade
  - Algumas vezes é possível reprojetar a gramática para eliminar a ambiguidade
  - Em outras vezes, isso é impossível
    - Ou seja, existem linguagens “inerentemente ambíguas”
    - Isto é: toda gramática para esta linguagem será fatalmente ambígua

Ambiguidade

- O que caracteriza ambiguidade
  - A existência de duas ou mais árvores de análise sintática para pelo menos uma cadeia da linguagem
- Formalmente:
  - Uma Gramatica Livre de Contexto G = (V,T,P,S) é ambígua se existe pelo menos uma cadeia w em T* para o qual podemos encontrar duas árvores de análise sintática diferentes, cada qual com uma raiz identificada como S e um resultado w.

  - Se TODAS as cadeias tiverem no máximo uma árvore de análise sintática, a gramática é não-ambígua

Ambiguidade

- Também pode-se pensar na ambiguidade em termos de derivações
- Teorema: Para cada gramática G = (V,T,P,S) e cadeia w em T*, w tem duas árvores de análise sintática distintas se e somente se w tem duas derivações mais à esquerda distintas a partir de S
  - Corolário: Se para uma gramática G = (V,T,P,S), e uma cadeia w em T*, for possível encontrar duas derivações mais à esquerda distintas, G é ambígua
- O mesmo vale para derivações mais à direita

Exercícios

- Prove que a seguinte gramática é ambígua
  - $S \rightarrow aS \mid aSbS \mid \varepsilon$
- Sugestão: mostre que a cadeia aab tem duas:
  - Árvores de análise sintática, derivações mais à esquerda ou derivações mais à direita
- Resposta (usando derivações mais à esquerda):
  - $S \rightarrow aS \rightarrow aaSbS \rightarrow aabS \rightarrow aab$
  - $S \rightarrow aSbS \rightarrow aaSbS \rightarrow aabS \rightarrow aab$

Exercícios

- Prove que a seguinte gramática é ambígua:
  - $S \rightarrow aSbS \mid aS \mid \varepsilon$
- Sugestão: mostre que a cadeia aab tem duas:
  - Árvores de análise sintática, derivações mais à esquerda ou derivações mais à direita
- Resposta (usando derivações mais à direita):
  - $S \rightarrow aSbS \rightarrow aSb \rightarrow aaSb \rightarrow aab$
  - $S \rightarrow aS \rightarrow aaSbS \rightarrow aaSb \rightarrow aab$

Exercícios

- Considere a seguinte gramática:
  - S → ε | SS | “if” C “then” S “else” S | “if” C “then” S
  - S → “System.out.print(” STRING “)”
  - C → “(i <= 1)”
  - STRING → “””[^”””]*”””
- A gramática é ambígua?
- Sugestão, analise a seguinte cadeia:

```
System.out.print('Teste')
if (i <= 1) then
    System.out.print('Alo Mundo')
    if (i< = 1) then
        System.out.print('Adeus Mundo')
    else
        System.out.print('Alo de novo')
System.out.print('Fim do programa')
```

Ambiguidade

- Eliminando a ambiguidade
  - Problemas
- Primeiro: saber se uma gramática é ambígua é um problema indecidível, ou seja, descobrir que uma gramática é ambígua depende de análise, exemplos e um pouco de sorte!
- Segundo: existem linguagens inerentemente ambíguas, ou seja, TODA GLC será ambígua
- Terceiro: mesmo para uma linguagem que não é inerentemente ambígua, não existe um algoritmo para remover a ambiguidade

Ambiguidade

- Existem algumas técnicas bem conhecidas, para alguns casos de ambiguidade
- Primeira técnica: forçar a precedência de terminais introduzindo novas regras
- Segunda técnica: modificar ligeiramente a linguagem
- Terceira técnica: forçar a precedência de terminais diretamente no analisador

Eliminando ambiguidade

- Forçando a precedência modificando-se as regras:
  - No exemplo à direita, há duas causas para ambiguidade:
    - Terminais “+” e “*” tem a mesma precedência
    - Terminais idênticos podem se agrupar a partir da esquerda ou direita
- Para resolver, vamos forçar a precedência e associatividade

$E \rightarrow I$
$E \rightarrow E + E$
$E \rightarrow E * E$
$E \rightarrow (E)$
$I \rightarrow a$
$I \rightarrow b$
$I \rightarrow Ia$
$I \rightarrow Ib$
$I \rightarrow I0$
$I \rightarrow I1$

Eliminando a ambiguidade

- Vamos pensar em expressões aritméticas em termos de:
  - Fatores
    - Elementos indivisíveis, ou seja, um fator não pode ser “quebrado” por um * ou +
    - Neste caso: identificadores e expressões entre parênteses
  - Termos
    - Elementos compostos de fatores multiplicados. Ou seja, um termo é uma multiplicação de um ou mais fatores, que não pode ser “quebrada” por um +
  - Expressões
    - Elementos compostos de termos somados. Ou seja, uma expressão é a soma de um ou mais termos

Eliminando ambiguidade

$E \rightarrow I$
$E \rightarrow E + E$
$E \rightarrow E * E$
$E \rightarrow (E)$
$I \rightarrow a$
$I \rightarrow b$
$I \rightarrow Ia$
$I \rightarrow Ib$
$I \rightarrow I0$
$I \rightarrow I1$

$E \rightarrow T \mid E + T$
$T \rightarrow F \mid T * F$
$F \rightarrow I \mid (E)$
$I \rightarrow a$
$I \rightarrow b$
$I \rightarrow Ia$
$I \rightarrow Ib$
$I \rightarrow I0$
$I \rightarrow I1$

Eliminando ambiguidade

- Faça o teste agora, para a cadeia a + b * a (com derivações mais à esquerda)
- $E \Rightarrow E + T \Rightarrow T + T \Rightarrow$

$F + T \Rightarrow I + T \Rightarrow a + T \Rightarrow$

$a + T * F \Rightarrow a + F * F \Rightarrow$

$a + I * F \Rightarrow a + b * F \Rightarrow$

$a + b * I \Rightarrow a + b * a$

```
E → T | E + T
T → F | T * F
F → I | (E)
I → a
I → b
I → Ia
I → Ib
I → I0
I → I1
```

Eliminando ambiguidade

- Segunda técnica: modificando ligeiramente a linguagem

  S → ε | SS | “if” C “then” S “else” S “endif” | if” C “then” S “endif”

  S → “System.out.print(” STRING “)”

  C → “(i <= 1)”

  STRING → “”” [^”””]*”””

- Veja o resultado

```
System.out.print('Teste')
if (i <= 1) then
   System.out.print('Alo Mundo')
   if (i< = 1) then
      System.out.print('Adeus Mundo')
   else
      System.out.print('Alo de novo')
   endif
endif
System.out.print('Fim do programa')
```

Eliminando ambiguidade

- Terceira técnica: forçar a precedência de terminais diretamente no analisador

Gramática
(com ambiguidade)

```
E → I
E → E + E
E → E * E
E → (E)
I → a
I → b
I → Ia
I → Ib
I → I0
I → I1
```

+

Regras de precedência
e associatividade

```
%left '+'
%left '*'
```

Exemplo no analisador YACC

Eliminando ambiguidade

- Sempre que houver um ponto de dúvida (ambiguidade), o YACC resolve olhando a precedência e associatividade
- Ex: a + b * a (no YACC)
  - Após ler o caractere "a", o YACC faz a inferência usando as regras $I \rightarrow a$ e $E \rightarrow I$, resultando em E + b * a
  - Após ler o caractere "+", não há inferência possível
  - Após ler o caractere "b", o YACC faz a inferência usando as regras $I \rightarrow b$ e $E \rightarrow I$, resultando em E + E * a
  - Neste ponto, ocorre um conflito (decorrente da ambiguidade da gramática)
    - O YACC pode:
      - Fazer a inferência, reduzindo E + E para E, resultando em E * a
      - Continuar a leitura, lendo o caractere "*", para só depois fazer a inferência

Eliminando a ambiguidade

- Como resolver esse conflito?
  - Através da precedência e associatividade dos terminais
  - No exemplo abaixo, * tem precedência sobre o +
  - E ambos são associativos à esquerda

```
%left '+'
%left '*'
```

- Ou seja, no momento o YACC está na seguinte configuração
  - E + E <YACC está aqui> * a
- Ele então olha para o terminal mais à direita do seu lado esquerdo (+), e o terminal mais à esquerda do seu lado direito (*)
  - Neste caso, * tem precedência sobre +, então a decisão é não inferir neste momento, e sim continuar lendo:
  - E + E * <YACC> a → E + E * a <YACC> → E + E * E → E + E → E

Ambiguidade inerente

- Algumas linguagens são inerentemente ambíguas
- Ex: $L = \{a^n b^n c^m d^m \mid n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$
- Ex: $L = \{a^i b^j c^k \mid i = j \text{ ou } j = k\}$
- Demonstração é complexa
  - Envolve analisar profundamente a linguagem e a característica de suas cadeias
- Para estes casos, é impossível remover a ambiguidade

Formatos e Formais Normais

Formatos

- A notação utilizada é a BNF ou Backus-Naur Form
- Basicamente é a mesma utilizada até agora, porém no seguinte formato:
  - <símbolo> ::= <expressão>
  - Onde <símbolo> é sempre um não-terminal ou variável, e expressão é uma sequência de terminais e/ou não terminais
  - O símbolo “|” significa a união de duas produções
  - Ex: S ::= E “+” E | E “*” E | “(“ E “)”
- Existe também a BFN estendida, ou EBNF
  - Basicamente existem algumas notações especiais
    - [ ] = opcional
    - { } = repetição
    - ( ) = agrupamento
    - Etc...

Formatos e formas normais

- Formatos são úteis para utilização prática
  - Ex: BNF surgiu para analisar programas em ALGOL
- Mas também existem as formas normais
  - São “maneiras” de escrever as produções, seguindo algumas propriedades
  - Ex: ao invés de escrever
    - $S \rightarrow abc$
  - Escrevo:
    - $S \rightarrow aB$
    - $B \rightarrow bC$
    - $C \rightarrow c$

Formais Normais

- Possuem algumas propriedades úteis, permitindo
  - Simplificar gramáticas
  - Analisar aspectos da gramática
  - Estudo teórico e formal
- Apresentaremos duas
  - Forma Normal de Chomsky
  - Forma Normal de Greibach

Forma Normal de Chomsky

- CNF – Chomsky Normal Form
- É uma forma normal simplificada, onde todas as produções estão em uma entre duas formas simples:
  - $A \rightarrow BC$, onde A, B e C são todas variáveis, ou
  - $A \rightarrow a$, onde A é uma variável e a é um terminal
- Qualquer linguagem livre de contexto é gerada por uma gramática livre de contexto na CNF
  - Ou seja, é possível converter uma gramática para a CNF

Forma Normal de Chomsky

- Antes de fazer a conversão, porém, é necessário:
  - Eliminar símbolos inúteis
  - Eliminar produções vazias ($A \rightarrow \varepsilon$)
  - Eliminar produções unitárias ($A \rightarrow B$, onde B é uma variável)

Eliminação de símbolos inúteis

- Formalmente:
  - Um símbolo X é útil para uma gramática G=(V,T,P,S), se existe alguma derivação da forma $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$, onde w está em T*
- Informalmente, X é útil se:
  - É gerador, ou seja, $X \Rightarrow^* w$ para alguma cadeia de terminas w
  - E é alcançável, ou seja, $S \Rightarrow^* \alpha X \beta$ para algum α e β
- Um símbolo útil é ao mesmo tempo gerador e alcançável, caso contrário ele é inútil
  - Símbolos inúteis podem ser removidos da gramática, junto com as produções que os envolvem
- Procedimento: primeiro eliminam-se os símbolos não geradores, e depois os símbolos não alcançáveis
  - Só sobram os símbolos úteis

Eliminação de símbolos inúteis

- Exemplo:
  - $S \rightarrow AB \mid a$
  - $A \rightarrow b$
- Todos os símbolos com exceção de B são geradores
  - a e b geram a si mesmos
  - S gera a, e A gera b
- Elimina-se o B, e as produções que o envolvem ($S \rightarrow AB$)
- Resultado:
  - $S \rightarrow a$
  - $A \rightarrow b$

Eliminação de símbolos inúteis

- Apenas S e a são alcançáveis a partir de S. Eliminando A e b, fica:
  - $S \rightarrow a$
- Essa gramática denota a mesma linguagem que a gramática original
- Observe que se tivéssemos primeiro removido os símbolos inalcançáveis e depois os geradores, o resultado seria errado

Cálculo de símbolos geradores e alcançáveis

- Algoritmo para calcular os símbolos geradores
  - Base: todo símbolo de T é sem dúvida gerador, pois ele gera a si próprio
  - Indução: suponha que exista uma produção $A \rightarrow \alpha$, e todo símbolo de α já seja conhecido como gerador. Então, A é gerador.
    - Obs: Se $A \rightarrow \varepsilon$, então A é gerador

Cálculo de símbolos geradores e alcançáveis

- Algoritmo para calcular os símbolos alcançáveis
  - Base: S é sem dúvida alcançável (pois é o símbolo inicial)
  - Indução: suponha que descobrimos que alguma variável A é alcançável. Então para todas as produções com A na cabeça, os símbolos dos corpos dessas produções também são alcançáveis.

Exercício

- Encontre uma gramática equivalente à seguinte, sem símbolos inúteis
  - $S \to AB \mid CA$
  - $A \to a$
  - $B \to BC \mid AB$
  - $C \to aB \mid b$
- Resposta:
  - Símbolos geradores: {a,b,A,C,S}
    - B não é gerador, portanto eliminamos as produções envolvendo B:
      - $S \to CA$
      - $A \to a$
      - $C \to b$
  - Símbolos alcançáveis: {S, A, C}
    - Não há símbolos não alcançáveis, portanto a gramática acima não possui símbolos inúteis

Eliminação de produções vazias

- Produções do tipo $A \to \varepsilon$ são convenientes no projeto de gramáticas, mas não são essenciais
- Podem ser removidas para simplificar a gramática
  - Por exemplo, a CNF não permite este tipo de produção
- Porém, há um efeito colateral
  - Se a linguagem da gramática inclui a cadeia vazia (ou seja, $S \Rightarrow^* \varepsilon$), a remoção das produções vazias obviamente elimina a cadeia vazia da linguagem

Eliminação de produções vazias

- Estratégia:
  - Descobrir quais variáveis são “anuláveis”
    - Ou seja, quais variáveis podem ser substituídas (em uma ou mais derivações) pela cadeia vazia
    - A é anulável se $A \Rightarrow^* \varepsilon$
- Algoritmo:
  - Base: se $A \rightarrow \varepsilon$ é uma produção, então A é anulável
  - Indução: se existe uma produção $B \rightarrow C_1C_2...C_k$, onde cada C_i é anulável, então B é anulável
    - Obs: cada C_i deve ser uma variável, pois terminais não são anuláveis

Eliminação de produções vazias

- Uma vez descobertos os símbolos anuláveis
  - Iremos modificar a gramática, mas sem modificar a linguagem
    - Com exceção, é claro, do fato de que a nova linguagem não mais aceita a cadeia vazia
- Algoritmo:
  - Para cada produção $A \rightarrow X_1X_2 \ldots X_k$, onde $k \geq 1$, suponha que m dos k valores de X_i sejam símbolos anuláveis
    - Criaremos $2^m - 1$ novas produções, de forma que todas as combinações possíveis em que cada X_i anulável está presente ou ausente façam parte da gramática

Eliminação de produções vazias

- Exemplo: Produção B → CADA, A e D são anuláveis
- Devemos produzir as seguintes produções (7):
  - B → C (A e D ausentes)
  - B → CAD (Segundo A ausente)
  - B → CAA (D ausente)
  - B → CDA (Primeiro A ausente)
  - B → CA (D e segundo A ausentes)
  - B → CD (Ambos os As ausentes)
  - B → CA (Primeiro A e D ausentes)
- Neste caso, a regra B → CA aparece duas vezes, e uma delas pode ser eliminada

Exercício

- Elimine as produções vazias da seguinte gramática
  - $S \rightarrow AB$
  - $A \rightarrow aAA \mid \varepsilon$
  - $B \rightarrow bBB \mid \varepsilon$
- Resposta:
  - Símbolos anuláveis: {A,B,S}
  - $S \rightarrow AB \mid A \mid B$
  - $A \rightarrow aAA \mid aA \mid a$
  - $B \rightarrow bBB \mid bB \mid b$

Eliminação de produções unitárias

- Produção unitária: $A \rightarrow B$, onde A e B são variáveis
  - Obs: $A \rightarrow b$ não é uma produção unitária, se b for um terminal
- Produções unitárias podem complicar certas provas, e introduzir etapas extras em derivações
- Método simples: expandir produções unitárias até que desapareçam

Eliminação de produções unitárias

- Exemplo:
  - $I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
  - $F \rightarrow I \mid (E)$
  - $T \rightarrow F \mid T * F$
  - $E \rightarrow T \mid E + T$
- Há três produções unitárias:
  - $F \rightarrow I$, $T \rightarrow F$ e $E \rightarrow T$
- Eliminando $E \rightarrow T$:
  - $E \rightarrow F \mid T * F \mid E + T$
- Agora apareceu outra: $E \rightarrow F$, eliminando:
  - $E \rightarrow I \mid (E) \mid T * F \mid E + T$
- Apareceu mais uma: $E \rightarrow I$, eliminando:
  - $E \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \mid (E) \mid T * F \mid E + T$
- Assim fazemos sucessivamente

Eliminação de produções unitárias

- Esse método tem um problema
  - Se houver um ciclo de produções unitárias, como $A \rightarrow B$, $B \rightarrow C$ e $C \rightarrow A$
- Existe uma segunda técnica
  - Primeiro, encontre todos os pares unitários
    - Base: (A,A) é um par unitário para qualquer variável A
    - Indução: suponha que descobrimos que (A,B) é um par unitário, e $B \rightarrow C$ é uma produção, onde C é uma variável. Então, (A,C) é um par unitário

Eliminação de produções unitárias

- Exemplo (gramática anterior):
  - Base: pares unitários = (E,E),(T,T),(F,F),(I,I)
  - Seguindo as demais produções unitárias, descobrimos outros pares unitários:
    - (E,T), (E,F), (E,I), (T,F), (T,I), (F,I)
- Agora, basta listar todos os pares unitários (incluindo aqueles do tipo (A,A) em uma tabela
  - E, para cada par (A,B), adicionar as produções $A \rightarrow \alpha$, onde $B \rightarrow \alpha$ é uma produção não-unitária na gramática original

Eliminação de produções unitárias

- Exemplo

I → a | b | Ia | Ib | I0 | I1
F → I | (E)
T → F | T * F
E → T | E + T

Par unitário	Produções
(E,E)	E → E + T
(E,T)	E → T * F
(E,F)	E → (E)
(E,I)	E → a | b | Ia | Ib | I0 | I1
(T,T)	T → T * F
(T,F)	T → (E)
(T,I)	T → a | b | Ia | Ib | I0 | I1
(F,F)	F → (E)
(F,I)	F → a | b | Ia | Ib | I0 | I1
(I,I)	I → a | b | Ia | Ib | I0 | I1

Eliminação de produções unitárias

- Exemplo

Par unitário	Produções
(E,E)	$E \rightarrow E + T$
(E,T)	$E \rightarrow T * F$
(E,F)	$E \rightarrow (E)$
(E,I)	$E \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
(T,T)	$T \rightarrow T * F$
(T,F)	$T \rightarrow (E)$
(T,I)	$T \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
(F,F)	$F \rightarrow (E)$
(F,I)	$F \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
(I,I)	$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Gramática:

$E \rightarrow E + T \mid T * F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

$T \rightarrow T * F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

$F \rightarrow (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Simplificações

- É preciso um certo cuidado na ordem de aplicação das simplificações:
  - Primeiro, eliminam-se as produções vazias
  - Segundo, eliminam-se as produções unitárias
  - Terceiro, elminam-se os símbolos inúteis
    - Primeiro eliminando-se os símbolos não-geradores
    - Depois eliminando-se os símbolos não-alcançáveis
- Essa ordem garante que não “sobra” nenhuma simplificação por fazer

Forma Normal de Chomsky

- Uma gramática na Forma Normal de Chomsky:
  - Não possui nenhum símbolo inútil
  - Não possui produções vazias
  - Não possui produções unitárias
    - Obs: neste ponto, com certeza toda produção terá a forma $A \rightarrow a$, ou terá um corpo de comprimento 2 ou mais
  - E todas as produções estão em uma dentre duas formas simples:
    - $A \rightarrow BC$, onde A, B e C são variáveis, ou
    - $A \rightarrow a$, onde A é uma variável e a é um terminal.
      - Obs: esta condição já é parcialmente aceita caso os três itens acima sejam satisfeitos

Forma Normal de Chomsky

- São necessárias duas tarefas:
  - a) Organizar todos os corpos de comprimento 2 ou mais que consistem apenas em variáveis
  - b) Desmembrar os corpos de comprimento 3 ou mais em uma cascata de produções, cada uma com um corpo consistindo em duas variáveis
- Para satisfazer a), basta fazer o seguinte:
  - Para todo terminal a que aparecer em um corpo de comprimento 2 ou mais, crie uma nova variável, digamos A, com apenas uma produção: $A \rightarrow a$
    - Usamos A em lugar de a em todo lugar que a aparecer em um corpo de comprimento 2 ou mais
  - Ex: $S \rightarrow XaYY \mid aaT$
  - Transformando:
    - $S \rightarrow XAYY \mid AAT$
    - $A \rightarrow a$

Forma Normal de Chomsky

- Neste ponto, toda produção terá um corpo que será um único terminal ou pelo menos duas variáveis e nenhum terminal
  - Agora basta desmembrar as produções $A \rightarrow B_1B_2...B_k$ para $k \geq 3$, em um grupo de produções com duas variáveis em cada corpo
  - Exemplo: $A \rightarrow B_1B_2B_3B_4B_5$
  - Resultado:
    - $A \rightarrow B_1C_1$
    - $C_1 \rightarrow B_2C_2$
    - $C_2 \rightarrow B_3C_3$
    - $C_3 \rightarrow B_4B_5$

Forma Normal de Chomsky

- Exemplo
- Considere a seguinte gramática:

$$S \rightarrow ASA \mid aB$$
$$A \rightarrow B \mid S \mid \varepsilon$$
$$B \rightarrow b \mid \varepsilon$$

Forma Normal de Chomsky

- Passo 1: Remover as produções vazias

S → ASA | aB
A → B | S | ε
B → b | ε

S → ASA | aB **| a**
A → B | S | ε
B → b ~~| ε~~

S → ASA | aB | a
S → SA | AS | S
A → B | S ~~| ε~~
B → b

Forma Normal de Chomsky

- Passo 2: Remover produções unitárias (A→B) (neste caso não há ciclos)

S → ASA | aB | a | SA | AS ~~| S~~
A → B | S
B → b

S → ASA | aB | a | SA | AS
A → ~~B |~~ S **| b**
B → b

S → ASA | aB | a | SA | AS
A → ~~S |~~ b **| ASA | aB | a | SA | AS**
B → b

Forma Normal de Chomsky

- Passo 3: Converter as regras remanescentes para a forma apropriada, criando novas variáveis quando necessário

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
$A \rightarrow b \mid ASA \mid aB \mid a \mid SA \mid AS$
$B \rightarrow b$

$S \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS$
$A \rightarrow b \mid AA_1 \mid UB \mid a \mid SA \mid AS$
$B \rightarrow b$
$A_1 \rightarrow SA$
$U \rightarrow a$

Exercícios

- Comece com a gramática

  - $S \rightarrow ASB \mid \varepsilon$
  - $A \rightarrow aAS \mid a$
  - $B \rightarrow SbS \mid A \mid bb$

- Elimine as produções vazias
- Elimine quaisquer produções unitárias na gramática resultante
- Elimine símbolos inúteis
- Coloque a gramática resultante na forma normal de Chomsky

Exercícios

- Eliminando produções vazias
  - Símbolos anuláveis: {S}
- Nova gramática:
  - $S \rightarrow ASB \mid AB$
  - $A \rightarrow aAS \mid aA \mid a$
  - $B \rightarrow SbS \mid bS \mid Sb \mid b \mid A \mid bb$

Exercícios

- Eliminando produções unitárias
  - A única produção unitária é $B \rightarrow A$, basta substituir o corpo
- Nova gramática
  - $S \rightarrow ASB \mid AB$
  - $A \rightarrow aAS \mid aA \mid a$
  - $B \rightarrow SbS \mid bS \mid Sb \mid b \mid aAS \mid aA \mid a \mid bb$

Exercícios

- Eliminando símbolos inúteis
  - Símbolos não-geradores
  - Símbolos não-alcançáveis
- Não existem, a gramática permanece a mesma

Exercícios

- Transformando para CNF (passo 1)
  - $S \rightarrow ASB \mid AB$
  - $A \rightarrow XAS \mid XA \mid a$
  - $B \rightarrow SYS \mid YS \mid SY \mid b \mid XAS \mid XA \mid a \mid YY$
  - $X \rightarrow a$
  - $Y \rightarrow b$
- Transformando para CNF (passo 2)
  - $S \rightarrow AE \mid AB$
  - $A \rightarrow CF \mid CA \mid a$
  - $B \rightarrow SG \mid DS \mid SD \mid b \mid CF \mid CA \mid a \mid DD$
  - $C \rightarrow a$
  - $D \rightarrow b$
  - $E \rightarrow SB$
  - $F \rightarrow AS$
  - $G \rightarrow YS$

Forma Normal de Greibach

- Toda produção é da forma $A \rightarrow a\alpha$
- Onde a é um terminal e α é uma cadeia de zero ou mais variáveis
- Conversão é complexa, mas existem algumas aplicações teóricas
  - Pelo fato de que cada produção introduz exatamente um único terminal
  - Ou seja, uma cadeia de comprimento n tem n etapas de derivação

Fim