Linguagens Formais e Autômatos

Aula 13 - Projetando Gramáticas Livres de Contexto

Prof. Dr. Daniel Lucrédio
Departamento de Computação / UFSCar
Última revisão: ago/2015

Projetando gramáticas livres de contexto

- Assim como autômatos
  - Requer conhecimento do formalismo
    - O que pode e o que não pode
  - Requer criatividade
    - Até mais, devido à recursividade
  - Requer prática e experiência
    - Para formar um “arsenal” de soluções e padrões típicos
  - Existem algumas técnicas que podem ajudar

Projetando gramáticas livres de contexto

- Muitas CFLs são a união de CFLs mais simples
- Pode ser mais fácil quebrar a linguagem em partes mais simples
  - Linguagem L = L1 ∪ L2
  - Projete uma CFG G1 para L1
  - Projete uma CFG G2 para L2
  - Crie uma CFG G, unindo G1 e G2, da seguinte forma:
    - Copie as produções de G1 e G2 para G
    - Crie uma nova produção, no início, com um novo símbolo inicial, da seguinte forma: S → S1 | S2
    - Onde S1 e S2 são os símbolos iniciais de G1 e G2, respectivamente

Projetando gramáticas livres de contexto

- Ex: $\{0^n1^n|n\geq 0\} \cup \{1^n0^n|n\geq 0\}$
  - G1:

    $S_1 \rightarrow 0S_1 1|\varepsilon$

  - G2:

    $S_2 \rightarrow 1S_2 0|\varepsilon$

  - G:

    $S \rightarrow S_1 \mid S_2$
    $S_1 \rightarrow 0S_1 1|\varepsilon$
    $S_2 \rightarrow 1S_2 0|\varepsilon$

Projetando gramáticas livres de contexto

- Outra técnica: certas CFLs contém cadeias com duas subcadeias que são “ligadas” da seguinte forma:
  - Para reconhecer uma das subcadeias é necessário memorizar informações sobre a outra
    - Ex: $\{ww^R\}$, $\{0^n1^n\}$
  - Pode-se usar regras do tipo $R \rightarrow uRv$
    - Ou seja, u e v estão “ligados”, de forma que não é possível reconhecer uma cadeia onde u e v não estejam presentes da maneira definida nesta produção
  - Ex: $\{0^n1^m|m=2n\}$
  - Solução $S \rightarrow 0S11 \mid \varepsilon$

Projetando gramáticas livres de contexto

- Mais uma técnica: em linguagens mais complexas, as cadeias podem conter certas estruturas que aparecem recursivamente como parte de outras estruturas (ou delas mesmas)
  - Ex: $E \rightarrow E + E \mid E * E \mid (E)$
- Para projetar este tipo de produção, use recursividade
  - Por exemplo: você está definindo E
    - Ou seja, ele está na cabeça de uma ou mais produções
  - Mas E ainda não está pronto (ainda faltam produções)
  - Mas mesmo assim, assuma que E está pronto, e use-o no corpo de E, caso necessário
- Não esqueça de definir uma condição de parada
  - $E \rightarrow \varepsilon$ ou $E \rightarrow a \mid b$

Projetando gramáticas livres de contexto

- Finalmente: testar
  - Experimente a gramática
  - Faça derivações, inferências recursivas, árvores de análise sintática
  - Experimente com cadeias que fazem parte da linguagem e que não fazem
    - Experimente também com cadeias no “extremo”
    - Ex: cadeia vazia, cadeia só com um símbolo, etc

Repassando

- União
  - A → a | b | c | d
  - B → e | f | g | h
  - C → A | B
  - D → AB
  - E → AAA
    - C leva a: a ou b ou c ou d ou e ou f ou g ou h
    - D leva a: ae, af, ag, ah, be, bf, bg, bh, ...
    - E leva a: aaa, aab, aac, aad, baa, bab, bac, bad, ...

- Concatenação
  - A → abcd
  - B → efgh
  - C → AB

- Opcional
  - C → aAb
  - A → c | ε
    - A leva a: ab ou acb

Repassando

- A → A b c d | a
  - Recursividade à esquerda, cria listas que “crescem” para a esquerda
    - bcd, **bcd**bcd, **bcd**bcdbcd, **a**bcdbcdbcd
- A → a b c A | d
  - Recursividade à direita, cria listas que “crescem” para a direita
    - abc, abc**abc**, abcabc**abc**, abcabcabc**d**
- A → a b A d e | c
  - Recursividade no centro, cria listas que “crescem” de dentro para fora
    - abde, ab**abde**de, abab**abde**dede, abababc**c**dededede

Exercícios

- Dê uma gramática livre de contexto gerando a seguinte linguagem:
  - O conjunto de cadeias sobre o alfabeto {a,b} com o mesmo números de as e bs
- Para testar, encontre derivações para os seguintes exemplos:
  - aabb, abab, abba, baba, bbaa, b, bbb
- Resposta
  - $S \rightarrow aSbS \mid bSaS \mid \varepsilon$

Exercícios

- Dê uma gramática livre de contexto gerando a seguinte linguagem:
  - O conjunto de cadeias sobre o alfabeto {a,b} com mais as do que bs
- Para testar, encontre derivações para os seguintes exemplos:
  - aab, aaaab, aba, ab, a
- Resposta
  - $S \rightarrow TaT$
  - $T \rightarrow TT \mid aTb \mid bTa \mid a \mid \varepsilon$

Exercícios

- Dê uma gramática livre de contexto gerando a seguinte linguagem:
  - Conjunto de cadeias sobre o alfabeto {(,)} onde para cada ocorrência de (existe uma ocorrência de), na ordem inversa
- Para testar, encontre derivações para os seguintes exemplos:
  - ()(), (()(())), (()), ())
- Resposta
  - $S \rightarrow (S) \mid SS \mid \varepsilon$

Exercícios

- Dê uma gramática livre de contexto gerando a seguinte linguagem:
  - $\{a^n\#b^m \mid m = 2n\}$
- Para testar, encontre derivações para os seguintes exemplos:
  - aa#bbbb, a#bba, aa#bb
- Resposta
  - $S \rightarrow aSbb$
  - $S \rightarrow B$
  - $B \rightarrow \#$

Exercícios

- Dois alunos de LFA, cansados de irem mal na disciplina, decidiram que iriam “colar” durante a próxima prova
- Inventaram um código para que pudessem se comunicar facilmente, passando as respostas das questões de forma cifrada, para que o professor não tivesse prova alguma de sua culpa
- Sendo estudantes de LFA, decidiram descrever o formato do código com uma gramática livre de contexto, para que ambos pudessem compreender bem o método

Exercícios

- O código consiste na seguinte idéia:
- Para codificar uma mensagem, como por exemplo: “verdadeiro”:
- Utilizam-se somente letras minúsculas (sem números ou outros símbolos)
- Coloca-se uma letra qualquer após cada letra da mensagem original, obtendo-se uma nova mensagem. Neste exemplo: “vdefrhdiaedheoiprror”
- Ao final da nova mensagem, insere-se o caractere “(“
- Após o “(“, repete-se a mensagem original, porém agora invertida, e colocando-se duas letras quaisquer após cada letra da mensagem original invertida. Neste exemplo: “vdefrhdiaedheoiprror(oacrfdigrejjdtrawedrrrhnelpvqa”
- O trecho após o “(“ serve como validação, caso parte da mensagem se torne ilegível por algum motivo (como gotas de suor de medo caindo sobre o papel)

Exercícios

- Projete uma gramática livre de contexto que reconhece esta linguagem
- Resposta:

S→aQSaQQ | bQSbQQ | cQScQQ | dQSdQQ |
eQSeQQ | fQSfQQ | ... | zQSzQQ

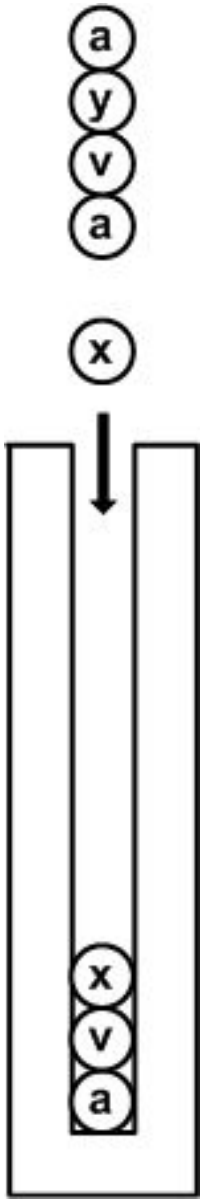
S→Par

Par → (

Q → a | b | c | d | e | f | ... | z

Exercícios

- Imagine um jogo de computador estilo “Tetris”, porém mais simples
- Neste jogo, bolinhas caem umas sobre as outras, em um corredor estreito onde só cabe uma fileira, como na figura ao lado.
- Cada bolinha tem uma letra (a-z), e as letras das bolinhas são escolhidas aleatoriamente
- Sempre que duas bolinhas com a mesma letra se tocam, ambas são destruídas
- O jogo é jogado da seguinte forma: o jogador precisa ser rápido ao identificar a letra certa e clicar nas bolinhas que caem, destruindo-as antes que as mesmas cheguem ao fundo
- Algumas bolinhas caem devagar, e outras depressa, de forma que provavelmente algumas chegarão ao fundo
- Com o passar do tempo, a velocidade média das bolinhas aumenta, tornando o jogo mais difícil
- O jogo começa com algumas bolinhas já no corredor, e termina quando o jogador conseguir limpar o corredor

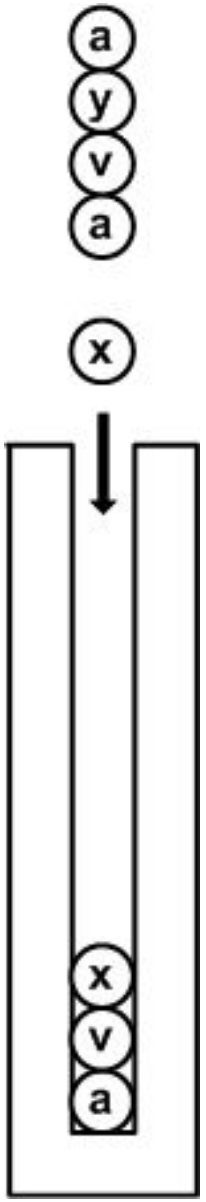


Exercícios

- Imagine que você é o projetista deste jogo, e o implementou da forma descrita
- Durante os testes, os jogadores reclamaram que em determinado momento, o jogo se tornava “injusto”, especialmente nas velocidades mais rápidas
  - pois o algoritmo aleatório acabava mandando muitas bolinhas que não serviam para destruir aquelas no topo da pilha
- É necessário um mecanismo que deixe o jogo mais “justo”, ou seja, de vez em quando é preciso mandar a bolinha certa
  - Além disso, foi identificado que, entre as próximas bolinhas a serem “sorteadas”, devem constar todas as bolinhas necessárias, e na ordem certa, para que o jogador possa vencer o jogo (podem haver outras)
  - Como um requisito adicional, foi identificado que deve aparecer a bolinha certa no máximo a cada cinco bolinhas

Exercícios

- Descreva o problema anterior em forma de linguagem, onde cada palavra ou cadeia representa um determinado momento do jogo, de acordo com os requisitos anteriores
- Resposta:
  - O alfabeto (ou conjunto de símbolos terminais) são as letras que aparecem nas bolinhas
  - As cadeias são compostas pelas bolinhas que já estão no corredor, e as bolinhas que ainda estão por vir, separadas por um espaço em branco
  - No exemplo ao lado, a cadeia “avx xavya” descreve o momento do jogo
  - Os requisitos se refletem na cadeia da seguinte forma:
    - As letras à esquerda do espaço em branco devem aparecer após o espaço em branco, na ordem inversa, e intercaladas por no máximo cinco letras quaisquer



Exercícios

- Projete uma gramática livre de contexto para essa linguagem
- Resposta 1:

S → aSQQQQQa | bSQQQQQb | cSQQQQQc |
dSQQQQQd | ... | zSQQQQQz | “ ”
Q → a | b | c | d | e | f | ... | z
Q → ε

Exercícios

- Problema com essa resposta: Q pode ser qualquer símbolo
- Ex: observe a seguinte cadeia, pode ser derivada a partir dessa gramática (faça a derivação como exercício):

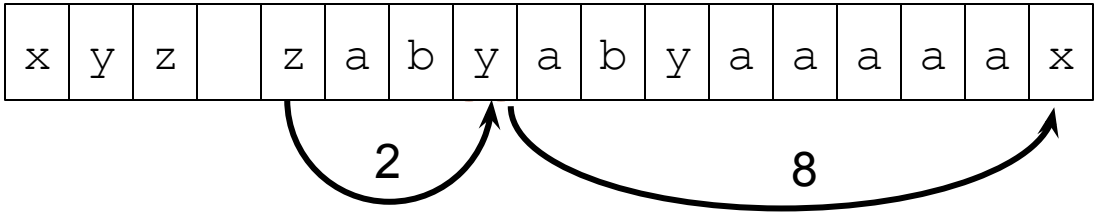
  “xyz zabyabyaaaaax”

  S ⇒ xSQQQQQX ⇒ xySQQQQQyQQQQQx ⇒ xyzSQQQQQzQQQQQyQQQQQx ⇒ xyz QQQQQzQQQQQyQQQQQx ⇒ xyz zQQQQQyQQQQQx ⇒ xyz zabyabyQQQQQx ⇒ xyz zabyabyaaaaax

- A cadeia significa que: no corredor, estão as bolinhas xyz, quer dizer que precisa vir, na ordem, zyx
- Além disso, não pode haver mais do que cinco bolinhas sem que apareça a bolinha correta

Exercícios

- Mas veja:



- Não pode existir essa situação
- Deve haver no máximo cinco bolinhas entre duas bolinhas “certas”

Exercícios

- Resposta 2:

$S \rightarrow aSQ_aQ_aQ_aQ_aQ_aa \mid bSQ_bQ_bQ_bQ_bQ_bb \mid cSQ_cQ_cQ_cQ_cQ_cc \mid dSQ_dQ_dQ_dQ_dQ_dd \mid ... \mid zSQ_zQ_zQ_zQ_zQ_zz \mid$ “ ”

$Q_a \rightarrow b \mid c \mid d \mid e \mid f \mid ... \mid z \mid \varepsilon$

$Q_b \rightarrow a \mid c \mid d \mid e \mid f \mid ... \mid z \mid \varepsilon$

$Q_c \rightarrow a \mid b \mid d \mid e \mid f \mid ... \mid z \mid \varepsilon$

...

$Q_z \rightarrow a \mid b \mid c \mid d \mid e \mid f \mid ... \mid y \mid \varepsilon$

Exercícios

- Achou muito complicado?
- Observe bem a gramática... existem partes que são linguagens regulares!!
  - Ex: $Q_aQ_aQ_aQ_aQ_a$, b | c | d | e | f | ... | z | ε
- LR → ER → Notações práticas!!!

$$Q_aQ_aQ_aQ_aQ_a = Q_a\{5\}$$

Q_a = [b-z]?
Q_b = [ac-z]?
Q_c = [abd-z]?
Q_d = [a-ce-z]?

Q_a = [a-z&&[^a]]?
Q_b = [a-z&&[^b]]?
Q_c = [a-z&&[^c]]?
Q_d = [a-z&&[^d]]?

Exercícios

- Simplificando a notação

S → aS[a-z&&[^a]]{0,5}a |

bS[a-z&&[^b]]{0,5}b |

cS[a-z&&[^c]]{0,5}c |

dS[a-z&&[^d]]{0,5}d |

...

zS[a-z&&[^z]]{0,5}z | “ ”

Exercícios

- Você precisa projetar um sistema de auxílio de controle de tráfego aéreo
- Além das informações do radar, a torre de controle recebe, via rede de dados, a identificação, posição, direção, velocidade e altitude de cada aeronave
- O sistema de tráfego aéreo armazena essas informações em um mesmo arquivo, usando uma linguagem específica

Exercícios

- Exemplo de arquivo

```
TAM2946:(41°25'01"N,120°58'57"W):79.44:884:15000

GOL1331:(81°24'21"S,145°14'33"E):179.33:712:13530

AZL5534:(32°24'51"N,104°53'16"E):150.45:500:5914

TAM5353:(27°10'37"N,120°49'22"E):89.54:891:13981
```

Exercícios

- Primeira parte: descreva os terminais usando ER
  - Identificação: três letras maiúsculas e quatro números (TAM2946, AZL5534, etc)
  - Posição: coordenadas geográficas:
    - (Latitude,Longitude)
    - Graus, minutos, segundos, ponto cardeal
    - Latitude: 0 a 90 graus
    - Longitude: 0 a 180 graus
    - Minutos, segundos: 0 a 59
    - `Ex: (41°25'01"N,120°58'57"W)`
  - Direção: número inteiro, de 0 a 359
  - Velocidade e altitude: números naturais

Exercícios

- Resposta:

```
Identificação → [A-Z]{3}[0-9]{4}

Posição → “(a” Latitude “,” Longitude “)”

Latitude → ([0-8][0-9]|90)°[0-5][0-9]'[0-5][0-9]"[NS]

Longitude → ([0-9][0-9]|1[0-7][0-9]|180)°[0-5][0-9]'[0-5][0-9]"[EW]

Direção → [0-2][0-9]{2}|3[0-5][0-9]

Velocidade → Inteiro

Altitude → Inteiro

Inteiro → [0-9]+
```

Exercícios

- Segunda parte: descreva a gramática usando os terminais anteriores
- Resposta (considerando que um arquivo tem uma ou mais aeronave):

```
Lista → Lista Aeronave | Aeronave

Aeronave → Identificação “:” Posição “:” Direção “:”
  Velocidade “:” Altura
```

- Obs: alguns sistemas também aceitam a seguinte notação

```
Lista → (Aeronave)+
```

Exercícios

- Você está projetando um sistema de auxílio à montagem em uma fábrica, baseado em CAD (Computer-Aided Design)
- A fábrica produz itens simples e complexos, e possui em um banco de dados todas as informações sobre os itens já projetados, assim como o seu desenho
- A funcionalidade que você está implementando é a seguinte:
  - Uma vez projetado o item, deve ser feito o cálculo do material e tempo necessários para sua produção
  - Para isso, o sistema CAD gera um relatório que descreve um item e todos os seus sub-itens
  - Este relatório deve ser automaticamente processado pelo sistema que você está projetando, que deve então fazer os cálculos necessários

Exercícios

- O arquivo gerado pelo sistema CAD tem o formato apresentado à direita
- Cada item pode conter zero ou mais subitens, sucessivamente
- O material pode ser Aço, Ferro, Alumínio ou Plástico
- Arquivo_CAD pode ou não ser especificado
- Você precisa projetar um analisador para este arquivo

```
Item {
Id="IT_015714"
Desc="Caixa de câmbio"
Dimensões=2400,1670,2020
Material=Aço
Arquivo_CAD="caixaCambio.cad"
  SubItem {
    Id="IT_044353"
    Desc="Parafuso 20"
    Dimensões=20,5,5
    Material=Ferro
  }
  SubItem {
    Id="39483"
    Desc="Engrenagem dentada 4"
    Dimensões=150,150,20
    Material=Alumínio
    Arquivo_CAD="engDent4.cad"
  }
  ...
}
```

Exercícios

- Descreva os terminais utilizando ERs
- Projete uma gramática livre de contexto para esse formato
- Resposta:

```
Arquivo → “Item” Item
Item → “{”
        “Id” “=” STRING
        “Desc” “=” STRING
        “Dimensões” “=” INT “,” INT “,” INT
        “Material” “=” Materiais
        ArquivoCad
        ListaSubItens
        “}”
ArquivoCad → “Arquivo_CAD” “=” STRING | ε
ListaSubItens → ListaSubItens “SubItem” Item | ε
Materiais → “Aço” | “Ferro” | “Alumínio” | “Plástico”;
STRING → “\”” [^\”]* “\””
INT → [0-9]+
```

Exercícios

- Outra resposta, com notação estendida:

```
Arquivo → “Item” Item
Item → “{”
        “Id” “=” STRING
        “Desc” “=” STRING
        “Dimensões” “=” INT “,” INT “,” INT
        “Material” “=” Materiais
        (“Arquivo_CAD” “=” STRING | ε)
        (“SubItem” Item)*
        “}”
Materiais → “Aço” | “Ferro” | “Alumínio” | “Plástico”;
STRING → “\”” [^\”]* “\””
INT → [0-9]+
```

Exercícios

- Outra resposta, com notação estendida:

```
Arquivo → “Item” Item
Item → “{”
        “Id” “=” STRING
        “Desc” “=” STRING
        “Dimensões” “=” INT “,” INT “,” INT
        “Material” “=” Materiais
        (“Arquivo_CAD” “=” STRING)?
        (“SubItem” Item)*
        “}”
Materiais → “Aço” | “Ferro” | “Alumínio” | “Plástico”;
STRING → “\”” [^\”]* “\””
INT → [0-9]+
```

Fim

Aula 13 - Projetando Gramáticas Livres de Contexto