Linguagens Formais e Autômatos

Linguagem Sensível ao Contexto

Eduardo Furlan Miranda

Baseado em: GARCIA, A. de V.; HAEUSLER, E. H.
Linguagens Formais e Autômatos. Londrina: EDA, 2017.

Linguagem Sensível ao Contexto (LSC)

- Algumas abreviações
  - LC Livre de Contexto
  - LLC Linguagem Livre de Contexto
  - GLC Gramática Livre de Contexto
  - RLC Regra Livre de Contexto
  - SC Sensível ao Contexto
  - LSC Linguagem Sensível ao Contexto
  - GSC Gramática Sensível ao Contexto
  - RSC Regra Sensível ao Contexto
  - LR Linguagem Regular
  - GR Gramática Regular
  - MT Máquina de Turing
  - AF Autômato Finito

Hierarquia de Chomsky

R	$a^n b$	ε, b, ab, aab	AF
LC	$a^n b^n$	ab, aabb	AFP
SC	$a^n b^n c^n$	abc, aabbcc	ALI
I	$a^{2\wedge n}$	a, aa, aaaa	MT

apenas estas

Gramáticas	Regras	Ex. de ling. geradas, e reconhecedor
GR (tipo 3) Regulares	A→aB, A→b, (A→ε apenas para o símbolo inicial, se permitido) A, B ∈ V (variáveis) a, b ∈ T (terminais)	• { ε, b, ab, aab, aaab, … } = { a^n b | n ≥ 0} ∪ {ε} • Autômato finito
GLC (tipo 2) Livres de Contexto	A → α A ∈ V , α ∈ (V ∪ T)* A: 1 única variável	• { ab, aabb, aaabbb, aaaabbbb, … } = { a^n b^n | n > 0 } • Autômato com pilha
GSC (tipo 1) Sensíveis ao Contexto	α → β α ∈ (V ∪ T)+ , β ∈ (V ∪ T)* | α | ≤ | β | S → ε, se S não aparece do lado direito de nenhuma regra	• { abc, aabbcc, aaabbbccc, … } = { a^n b^n c^n | n > 0 } • Autômato linearmente limitado
GI (tipo 0) Irrestrita ou geral	α → β α, β ∈ (V ∪ T)* α: pelo menos 1 símbolo de V	• { a, aa, aaaa, aaaaaaaa, … } = { a^{2^n} | n ≥ 0} • Máquina de Turing

Linguagens Livres de Contexto (LLC)

(já visto)

- Regras de produção na forma $A \rightarrow \alpha$
  - Geradas por GLC com regras de produção que dependem apenas do símbolo atual, e não do "contexto em que ele aparece"
    - Depende da sequência de símbolos que precede ou segue

- GLC possuem regras na forma
  $A \rightarrow \alpha$, onde $\alpha \in (V \cup T)^*$ e $A \in V$

R	$a^n b$	ε,b,ab,aab
LC	$a^n b^n$	ab, aabb
SC	$a^n b^n c^n$	abc, aabbcc
I	$a^{2\hat{}n}$	a, aa, aaaa

Tipos de Gramáticas	**Regras**	**Exemplos de linguagens geradas**
Livres de Contexto GLC (tipo 2)	$A \rightarrow \alpha$ $A \in V$, $\alpha \in (V \cup T)^*$ A: 1 única variável esq.	{ ab, aabb, aaabbb, aaaabbbb, ... } = { $a^n b^n$ | n > 0 }

Exemplo

- Preciso reconhecer o comando **print(texto)**

- A gramática pode ser definida como:
  S → "print" E
  E → "(" T ")"
  T → "texto"

"→" = símbolo de produção ou regra de derivação
"⇒" = símbolo de derivação

- Exemplo de reconhecimento:
  - S ⇒ "print" E ⇒ "print" "(" T ")" ⇒ "print" "(" "texto" ")"

  - Para que "print(texto)" seja reconhecido, precisa existir uma sequência de derivações começando de S até essa cadeia, seguindo as regras da gramática definida
  - "print" e "texto" seriam tokens reconhecidos na etapa de análise léxica

Linguagens Sensíveis ao Contexto (LSC)

RSC = Regra Sensível ao Contexto

- Regras de produção na forma $\alpha A \beta \rightarrow \alpha \gamma \beta \quad (\gamma \neq \epsilon)$
  - Geradas por gramáticas que têm regras de produção que "dependem do contexto em que o símbolo atual aparece"
    - Dependem não apenas do símbolo atual, mas também da sequência de símbolos que o precedem ou o seguem
      - α e β representam o contexto
        Ex.: aAc→aBc só pode ser aplicada se a A estiver entre a e c

- RSC : o tamanho do lado esquerdo é ≤ ao do lado direito

- GSC : é tupla G = (V, T, P, S) onde toda regra em P é uma RSC

| α | representa o tamanho (ou comprimento) da cadeia α

Sensíveis ao Contexto GSC (tipo 1)	$\alpha \rightarrow \beta$ $\alpha \in (V \cup T)^+$, $\beta \in (V \cup T)^*$ $\lvert \alpha \rvert \leq \lvert \beta \rvert$ S → ε , se S não aparece do lado dir.	{ abc, aabbcc, aaabbbccc, ... } = { $a^n b^n c^n \mid n > 0$ }

Compiladores e analisadores sintáticos

- Na prática, as GSC não são utilizadas na construção de analisadores sintáticos, pois acarretaria em um mecanismo de análise complicado e ineficiente
  - A derivação de uma cadeia em uma GSC não possui uma estrutura que represente tão bem a cadeia do ponto de vista hierárquico,
    - como é o caso das árvores de análise para as LLC
- Compiladores usam analisadores guiados por GLC ,
  - e alguns mecanismos mais sofisticados que as GSC , para dar conta de análise de contexto

Gramática Sensível ao Contexto (GSC)

- Seja a gramática G = (V, T, P, S). As RSC têm as formas:
  - $\alpha \rightarrow \beta$, com $\alpha, \beta \in (V \cup T)^{+}$, α possuindo pelo menos uma ocorrência de variável , e $|\alpha| \leq |\beta|$
    - Tanto α quanto β são sequências não vazias de símbolos, onde cada símbolo pode ser uma variável V ou um terminal T
  - $S \rightarrow \varepsilon$, se S é o símbolo inicial e não ocorre à direita de todas as regras da gramática
    - Esta é a condição adicional que permite que o símbolo inicial S seja substituído pelo símbolo vazio ε , desde que S não ocorra à direita de nenhuma regra da gramática

Linguagem Sensível ao Contexto (LSC)

- L é uma LSC se, e somente se, existe uma GLC G , e L = L(G) .
  - Uma linguagem L é uma Linguagem Livre de Contexto se, e somente se, existe uma Gramática Livre de Contexto G que a gere.
  - L = L(G) ← L(G) representa a linguagem gerada pela gramática G
    - L(G) é o conjunto de todas as cadeias de caracteres que podem ser derivadas a partir do símbolo inicial da gramática G seguindo suas regras de produção

- LSCs podem gerar cadeias, com um controle triplo, em comparação com as LLCs, caracterizadas pelo controle duplo

(αAβ)

(α→β)

R	$a^n b$	ε,b,ab,aab
LC	$a^n b^n$	ab, aabb
SC	$a^n b^n c^n$	abc, aabbcc
I	$a^{2\hat{}n}$	a, aa, aaaa

Exemplo - LSC

- Seja a linguagem $\{a^n b^n c^n \mid 0 < n\}$ gerada pela GSC:
  - S → ABc ← inicial
  - AB → AABBC
  - CB → BC
  - Cc → cc
  - Bc → bc
  - Bb → bb
  - Ab → ab
  - Aa → aa

Derivação de $a^3b^3c^3$:

S ⇒ ABc ⇒ AABBCc

AABBCc ⇒ AAABBCBCc

AAABBCBCc ⇒ AAABBBCCc

AAABBBCCc ⇒ AAABBBCcc

AAABBBCcc ⇒ AAABBBccc

AAABBBccc ⇒ AAABBbccc

AAABBbccc ⇒ (...) ⇒ Aaabbbccc

Aaabbbccc ⇒ aaabbbccc

- Na derivação de aaabbbccc , as regras poderiam ter sido aplicadas em outra ordem, que não a exibida, e assim mesmo a mesma cadeia seria derivada

- Por exemplo, a partir de AAABBCBCc , podemos derivar AAABBCBcc ao invés de AAABBBCCc
  - a partir de AAABBCBcc só poderíamos derivar AAABBBCcc,
    - enquanto que a partir da cadeia AAABBBCCc só é possível derivar AAABBBCcc

- Desta forma, podemos observar que qualquer sequência de aplicações de regras da gramática gera cadeias da linguagem em questão

Exemplo - GSC

GSC que gera a L = { ww | w ∈ { a, b }* } :

1) S → RT | aa | bb ← S = inicial

2) R → RaA | RbB | aaM | abN | baO | bbP

3) Aa → aA
   Ab → bA
   AT → Ta

4) Ba → aB
   Bb → bB
   BT → Tb

5) Ma → aM
   Mb → bM
   MT → aa

6) Na → aN
   Nb → bN
   NT → ab

7) Oa → aO
   Ob → bO
   OT → ba

8) Pa → aP
   Pb → bP
   PT → bb

(continua)

Exemplo - GSC (continuação)

- A gramática é projetada para processar palavras w de forma ordenada e sensível ao contexto
- Ela lê w no prefixo R e garante, por meio de regras intermediárias e sensíveis ao contexto, que o sufixo T será uma cópia idêntica de w, resultando em ww
- Exemplo de derivação

$$S \Rightarrow RT \Rightarrow RaAT \Rightarrow bbPaAT \Rightarrow bbaPAT \Rightarrow bbaPTa \Rightarrow bbabba$$

$R \to RaA$ $R \to bbP$ $Pa \to aP$ $AT \to Ta$ $PT \to bb$

Gramática Sensível ao Contexto (GSC)

- GSC, como possuem regras apenas na forma

  $\alpha \rightarrow \beta$, onde $|\alpha| \leq |\beta|$,

  garantem que é computável o problema de determinar se uma cadeia pertence ou não à linguagem gerada pela gramática G
  - Isto é, existe um programa de computador que implementa esta decisão

- Isto é consequência do fato de que a cada uso de regra no processo de verificação, a cadeia sendo gerada nunca diminui de tamanho

GSC

- Devido às regras das GSC, onde uma cadeia gerada nunca diminui de tamanho,
  - é possível criar um programa de computador que decida se uma cadeia pertence à linguagem gerada por uma gramática G

- Com um número finito de aplicações de regras que não aumentam o tamanho da cadeia, podemos concluir que a cadeia w pertence à linguagem L(G) e é derivada pela gramática

- Porém, se todas as possíveis derivações resultam em cadeias maiores que w, isso indica que G não pode derivar w

RLC é RSC

- Com a exceção de regras da forma $A \rightarrow \varepsilon$, toda RLC é uma RSC
- Toda GLC que não gera a cadeia vazia pode ser convertida em uma Gramática Livre de Contexto Sem Regras Nulas (GSRN) equivalente
- Toda LLC é também uma LSC, uma vez que a classe das linguagens sensíveis ao contexto inclui todas as linguagens livres de contexto

- Existem LSC que não são LLC
  - Ex.: $L_{abc} = \{ a^n b^n c^n \mid 0 < n \}$
    - Existe a necessidade de "manter a contagem" simultânea de três símbolos diferentes (a, b e c)
    - As GLCs têm uma capacidade limitada de "memória", pois só podem substituir um não terminal por vez, independentemente do contexto
    - Para garantir a igualdade entre as quantidades de a, b e c, seria necessário um mecanismo que as GLCs não possuem

- Com relação às operações de fechamento, podemos observar que o que foi feito para a união de LLC pode ser feito para a união de LSC

- Se L_1 e L_2 são LSC, então $L_1 \cup L_2$ também é uma LSC

- Se L_1 e L_2 são LSC, então existem gramáticas G_1 e G_2, tais que $L_1 = L(G_1)$ e $L_2 = L(G_2)$
  - G_1 é uma gramática que gera L_1, G_2 é uma gramática que gera L_2
  - A linguagem L_1 é exatamente a linguagem gerada pela gramática G_1
  - Idem para L_2

são operações que, quando aplicadas a elementos de um conjunto, produzem resultados que ainda pertencem ao mesmo conjunto

- Se S_1 e S_2 são os respectivos símbolos inicias das gramáticas,
  - deve-se renomear as variáveis de G_1 para que não haja interseção com as variáveis de G_2

- Considerando a gramática que reúne todas as regras de ambas as gramáticas, mais as regras $S \rightarrow S_1 \mid S_2$, que é uma RSC, temos uma GSC que gera $L_1 \cup L_2$

- Considerando a GSC L_1 anterior, e as regras $S \rightarrow S_1S \mid S_1$, temos uma gramática que gera L_1^+
- A regra $S \rightarrow S_1S \mid S_1$ é sensível ao contexto
- A união de L_1^+ com $\{ \varepsilon \}$ é sensível ao contexto, e, de fato $L_1^* = L_1^+ \cup \{ \varepsilon \}$
- Sejam L_1 e L_2 linguagens sensíveis ao contexto
  - As linguagens $L_1 \cup L_2$ e L_1^* também são sensíveis ao contexto