Gramáticas e Linguagens Livres de Contexto

Formalismo

Derivações

Derivações mais à esquerda e mais à direita

Definição Informal

- Uma *gramática livre de contexto* é uma notação para descrever linguagens.
- É muito mais poderosa que autômato finito ou RE's, mas ainda não define todas as linguagens possíveis.
- Útil para estruturas aninhadas, por exemplo, os parênteses em linguagens de programação.

Definição Informal – (2)

- Idéia básica é usar "variáveis" para representar conjuntos de strings (isto é, linguagens).
- Estas variáveis são definidas de forma recursiva, em termos de uma outra.
- Regras recursivas ("produções") envolvem somente concatenação.
- Regras alternativas para uma variável permitem a união.

Exemplo: CFG para $\{ 0^n1^n \mid n \geq 1\}$

- Produções:

  S -> 01

  S -> 0S1

- Base: 01 está na linguagem.
- Indução: se w está na linguagem, então 0w1 também está.

CFG - Definição

- *Terminais* = símbolos do alfabeto da linguagem que está sendo definida.
- *Variáveis* = *não terminais* = um conjunto finito de outros símbolos, cada um representa uma linguagem.
- *Símbolo de início* = a variável cuja linguagem é a que está sendo definida.

Produções

- Uma *produção* tem a forma
  variável → string de variáveis e terminais
- Convenção:
  - A, B, C,… são variáveis.
  - a, b, c,… são terminais.
  - …, X, Y, Z são terminais ou variáveis.
  - …, w, x, y, z são strings de terminais somente.
  - α, β, γ,… são strings de terminais e/ou variáveis.

CFG Formal

- Uma Gramática Livre de contexto G é uma gramática: G = (V, T, P, S)
- V: conjunto finito de variáveis
- T: conjunto finito de símbolos
- P: conjunto finito de produções ou regras
  - Com a restrição de que qualquer regra de produção de P é A -> α, A $\in$ V e $\alpha \in (V \cup T)^*$
- S: símbolo de início

Exemplo: CFG Formal

- Uma CFG formal para $\{ 0^n1^n \mid n \geq 1\}$.

  CFG G = ({S}, {0,1}, P, S)
- Terminais = {0, 1}.
- Variáveis = {S}.
- Símbolo inicial = S.
- Produções P =

  S -> 01

  S -> 0S1

Outro exemplo:

- Gramática livre de contexto para os palíndromos.

- Um palíndromo é um string lido da mesma forma, da esquerda para a direita ou da direita para a esquerda.
  - Exemplo: radar, 0110, 11011

Derivações – Intuição

- Nós *derivamos* strings na linguagem de um CFG começando com o símbolo inicial, e repetidamente substituindo alguma variável A pelo lado direito de uma das suas produções.
  - Istó é, uma "produção para A" são aquelas que tem A no lado esquerdo da →.

Derivações – Formalismo

- Nós dizemos que $\alpha A \beta \Rightarrow \alpha \gamma \beta$ se $A \rightarrow \gamma$ é uma produção.
- Exemplo: S -> 01; S -> 0S1.

  S => 0S1 => 00S11 => 000111.

*Importância: permite estabelecer analogia com estruturas de duplo balanceamento em linguagens de programação (exemplo: $begin^n end^n$ $(^n)^n$)

Derivação Estendida

- =>* significa "zero ou mais etapas de derivação."
- Base: α =>* α para algum string α.
- Indução: se α =>* β e β => γ, então α =>* γ.

Exemplo: Derivação Extendida

- S -> 01; S -> 0S1.
- S => 0S1 => 00S11 => 000111.
- Assim, S =>* S;

  S =>* 0S1;

  S =>* 00S11;

  S =>* 000111.

Forma Sentencial

- Um string de variáveis e/ou terminais derivado do símbolo inicial é chamado *forma sentencial*.
- Formalmente, α é uma forma sentencial se e somente se S =>* α.
- A linguagem L(G) é constituída pelas formas sentenciais que estão em T* (apenas terminais).

Linguagem de uma Gramática

- Se G é uma CFG, então L(G), a *linguagem de G*, é {w em T*| S =>* w}.
  - Note: w precisa ser um string de terminais e S é o símbolo inicial.
- Exemplo: G tem produções S -> ϵ e S -> 0S1.

  L(G) = $\{0^n1^n \mid n \geq 0\}$.

Note: ϵ é um lado direito legítimo.

Linguagens Livre de Contexto

- Uma linguagem que é definida por alguma CFG é chamada uma *context-free language*.
- Existem CFL's que não são linguagens regulares, tal como o exemplo dado.
- Mas nem todas linguagens são CFL's.
- Intuitivamente: CFL's podem contar duas coisas, não três.

Derivações mais à esquerda e mais a direita

- Derivações nos permitem substituir qualquer uma das variáveis em um string.
- Isto leva a muitas derivações diferentes da mesma cadeia.
- Ao forçar a variável mais à esquerda (ou, a variável mais à direita) para ser substituída, nós evitamos estas "distinções sem diferença."

Derivação mais à esquerda

- Dizemos $wA\alpha \Rightarrow_{lm} w\beta\alpha$ se w é um string de terminais e A -> β é uma produção.
- Também, $\alpha \Rightarrow^*_{lm} \beta$ se α torna-se β por uma sequência de 0 ou mais etapas $\Rightarrow_{lm}$

Exemplo: Dirivações mais à esquerda

- Gramática de parentêses balanceados:

  S -> SS | (S) | ()

- $S \Rightarrow_{lm} SS \Rightarrow_{lm} (S)S \Rightarrow_{lm} (())S \Rightarrow_{lm} (())()$
- Assim, $S \Rightarrow^*_{lm} (())()$
- S => SS => S() => (S)() => (())() é uma derivação, mas não uma derivação mais à esquerda.

Derivações mais à direita

- Dizemos que $\alpha Aw \Rightarrow_{rm} \alpha\beta w$ se w é um string de terminais e A -> β é uma produção.
- Também, $\alpha \Rightarrow^*_{rm} \beta$ se α torna-se β por uma sequência de 0 ou mais etapas $\Rightarrow_{rm}$

Exemplo: Derivações mais à direita

- Gramática de parentêses balanceados:

  S -> SS | (S) | ()

- $S \Rightarrow_{rm} SS \Rightarrow_{rm} S() \Rightarrow_{rm} (S)() \Rightarrow_{rm} (())()$
- Assim, $S \Rightarrow^*_{rm} (())()$
- S => SS => SSS => S()S => ()()S => ()()() não é nem uma derivação mais à direitais nem uma derivação mais à esquerda.

Exemplo

- CFG que representa expressões em uma linguagem de programação típica.
  - Vamos nos limitar aos operadores + e *
  - Todo identificador deve começar com *a* ou *b*, que podem ser seguidos por qualquer string em {*a*,*b*,0,1}*

Exercícios

Construa CFG's que gerem as seguintes linguagens sobre o alfabeto {0,1}

1) $\{a^i b^j c^k \mid i \neq j \text{ ou } j \neq k\}$
2) {w | w começa e termina com o mesmo símbolo}
3) {w | |w| é ímpar e o seu símbolo do meio é 0}
4) Strings binários com duas vezes mais 1's que 0's

Exercícios

- A Gramática a seguir gera a linguagem regular: 0*1(0+1)*

  S -> A1B

  A -> 0A | ε

  B -> 0B |1B | ε

1. Forneça derivações mais a equerda e mais à direita dos strings:
   a) 00101
   b) 1001
   c) 00011