Linguagens Formais e Autômatos

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Conteúdo I

1. Parte 1
   - Representação e prova de teoremas
   - Conjuntos, relações, funções, conjuntos enumeráveis, definições recursivas, indução matemática e grafos
   - Linguagens formais, gramáticas e problemas de decisão
   - Autômatos Finitos
   - Revisão e exercícios

Conteúdo II

2. Parte 2
   - O Lema do Bombeamento para Linguagens Regulares
   - Expressões Regulares
   - Gramáticas Regulares
   - Máquinas de Mealy e de Moore
   - Revisão e exercícios

Conteúdo III

3. Parte 3
   - Autômatos de Pilha
   - Gramáticas Livre do Contexto: parte 1
   - Gramáticas Livre do Contexto: parte 2
   - Propriedades de Linguagens Livre do Contexto
   - Revisão e exercícios

Linguagens Formais e Autômatos

Semana 0: Organização da disciplina

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Organização da Disciplina

$\Rightarrow$ 3 partes com duração de 5 semanas cada.

Cronograma

- Parte 1: Introdução às Linguagens Formais
- Parte 2: Linguagens Regulares e Autômatos
- Parte 3: Linguagens Livre do Contexto e Autômatos de Pilha

Atenção!

Cronograma sujeito a alterações e adaptações!

Organização da Disciplina

Metodologia

- Video aulas assíncronas (teóricas e práticas)
- Discussões via fórum do AVA
- Atendimento presencial via Discord

- Atendimento preferencial nas quintas-feiras de 13:00 às 15:00
- Atendimento não-preferencial nas quintas-feiras de 16:40 às 18:40

Organização da Disciplina

Avaliações

- 80% - Listas de exercícios semanais(Python e/ou URI)
- 20% - Contribuição e iniciativa
- Média geométrica
- Deadlines rigorosos

Utilize a Caixa de sugestões anônima e deixe críticas, comentários e dicas.

Linguagens Formais e Autômatos

Semana 0: Organização da disciplina

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Linguagens Formais e Autômatos

Semana 1: Representação e prova de teoremas

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Sobre este material

- Esses slides foram preparados para as disciplinas MATA50-Linguagens Formais e Autômatos
- O texto nos slides é extremamente resumido, utilize um dos livros indicado nas referências para um estudo mais aprofundado.

Referências

Esses slides foram baseados nas seguintes diversas fontes de conhecimento, tais como:

- Vieira, Newton., **Introdução aos fundamentos da computação: Linguagens e máquinas**, Cengage Learning, 2006.
- Hopcroft, J. E., **Introdução à Teoria De Autômatos, Linguagens E Computação**, Elsevier, 2002.
- Notas de aula do Prof. Newton Vieira, UFMG.

A matemática entre a entidade e a representação

Entidade	**Modelo Matemático**	**Representação**
Mês	Número inteiro no intervalo de [1,12]	Um dos caracteres 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 , A ou B
Remuneração	Número real positivo	Número real na base 10
FP	Relação	Tabela em que cada linha tem o nome, cargo, salário, etc.
Cálculo de FP	Algoritmo	Programa

- Representação por sequência de símbolos $\Rightarrow$ Linguagens Formais

Representações múltiplas

Exemplo

Diferentes representações para mês:

- Janeiro, fevereiro, ..., dezembro. Símbolos: letras a,b,...,z.
- Numerais na base decimal. Símbolos: dígitos 0 a 9.
- Numerais na base binária. Símbolos: dígitos 0 e 1.
- Numerais em algarismos romanos. Símbolos: letras I, V e X.
- Sequências de um a doze 0s. Símbolos: apenas o dígito 0.

$\Rightarrow$ Cada conjunto é uma linguagem e cada sequência de símbolos representa um mês.

Linguagens de programação

Exemplo de linguagem formal: linguagem de programação

- Linguagem de programação: conjunto de programas
- Programa: uma sequência de símbolos
- Símbolos: caracteres

- Linguagens formais: úteis para caracterizar o conceito de computabilidade
- Existe uma infinidade de funções não computáveis

Quiz

Em uma representação numérica binária, quantos bits são necessários para representar os 12 meses do ano?

a) 1
b) 2
c) 3
d) 4

Características de provas de teoremas

- Estilo:
  - formal $\times$ informal;
  - conciso $\times$ prolixo.
- Prova:
  - vocabulário limitado: se ... então, contradição, etc.
  - usa técnicas de prova.

Conectivos lógicos

Os conectivos lógicos: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists$

Tabela verdade para a negação

Negação	
α	$\neg\alpha$
V	F
F	V

Conectivos lógicos

Tabela verdade para a conjunção

Conjunção

α	β	$\alpha \wedge \beta$
V	V	V
V	F	F
F	V	F
F	F	F

Conectivos lógicos

Tabela verdade para a disjunção

Disjunção

α	β	$\alpha \vee \beta$
V	V	V
V	F	V
F	V	V
F	F	F

Conectivos lógicos

Tabela verdade para a condicional

condicional

α	β	$\alpha \to \beta$
V	V	V
V	F	F
F	V	V
F	F	V

Tabela verdade para a bicondicional

Bicondicional

α	β	$\alpha \leftrightarrow \beta$
V	V	V
V	F	F
F	V	F
F	F	V

Quantificação

- Quantificação universal: $\forall x P(x)$
  - $P(x)$ é verdadeira para todo x do universo
- Quantificação existencial: $\exists x P(x)$
  - $P(x)$ é verdadeira para algum x do universo

Exemplo

Expressar formalmente: todo número natural par ao quadrado é par:
$\forall n[n \in \mathbb{N} \rightarrow (n \text{ é par } \rightarrow n^2 \text{ é par})]$

Afirmativa válida

Verdadeira para todos os valores-verdade de suas subafirmativas.

- $\alpha \vee \neg\alpha$
- $\alpha \rightarrow \alpha$
- $\alpha \vee (\alpha \rightarrow \beta)$
- $P(a) \rightarrow \exists x P(x)$
- $\forall x P(x) \leftrightarrow \neg\exists x \neg P(x)$

Contradição

Falsa para todos os valores-verdade de suas subafirmativas.

- $\alpha \wedge \neg\alpha$
- $\alpha \leftrightarrow \neg\alpha$
- $(\alpha \wedge (\alpha \rightarrow \beta)) \wedge \neg\beta$
- $P(a) \wedge \neg\exists x P(x)$
- $\forall x P(x) \leftrightarrow \neg\exists x \neg P(x)$

Equivalência lógica

$\alpha \equiv \beta$ se o valor-verdade de α e β é o mesmo para todos os valores-verdade de suas subafirmativas.

- $\alpha \vee \beta \equiv \beta \vee \alpha$
- $\alpha \vee (\beta \wedge \gamma) \equiv (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
- $\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta$
- $\alpha \rightarrow \beta \equiv \neg\alpha \vee \beta$
- $\alpha \leftrightarrow \beta \equiv (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$
- $\neg\forall x P(x) \equiv \exists x \neg P(x)$

Consequência lógica (implicação lógica)

$\Gamma \Rightarrow \beta$ se β é verdadeira sempre que as afirmativas em Γ são.

- $\{\alpha \to \beta, \alpha\} \Rightarrow \beta$ volte na tabela verdade de "$\to$"
- $\{\alpha \to \beta, \neg\beta\} \Rightarrow \neg\alpha$
- $\{\alpha \to \beta, \neg\alpha \to \beta\} \Rightarrow \beta$
- $\{\alpha \to \beta, \beta \to \gamma\} \Rightarrow \alpha \to \gamma$
- $\{\alpha \to \beta, \beta \to \gamma\} \Rightarrow \alpha \to \gamma$
- $\{P(a)\} \Rightarrow \exists x P(x)$
- $\{P(a), \forall x(P(x) \to Q(x)\} \Rightarrow Q(a)$

Exemplos de **regras de inferência**

$\Gamma \Rightarrow \beta$ se β é verdadeira sempre que as afirmativas em Γ são.

$$\frac{\alpha \quad \alpha \to \beta}{\beta} \qquad \frac{\alpha \quad \neg\alpha \vee \beta}{\beta} \qquad \frac{\neg\beta \quad \alpha \to \beta}{\neg\alpha}$$

$$\frac{\alpha \to \beta \quad \neg\alpha \to \beta}{\beta} \qquad \frac{\alpha \to \beta \quad \alpha \to \gamma}{\alpha \to \gamma} \qquad \frac{\alpha \leftrightarrow \beta \quad \beta \leftrightarrow \gamma}{\alpha \leftrightarrow \gamma}$$

Relação entre $\to$ e $\Rightarrow$:
se $\Gamma \cup \{\alpha\} \Rightarrow \beta$, então $\Gamma \Rightarrow \alpha \to \beta$.

Técnica de prova: direta

Prova direta da implicação

Para provar $\alpha \rightarrow \beta$:

1. Supor α
2. Provar β

Exemplo

- n é par $\rightarrow n^2$ é par.

Veja a solução na página 8 do livro texto.

Técnica de prova: pela contrapositiva

Prova da implicação pela contrapositiva

Para provar $\alpha \to \beta$:

1. Supor $\neg\beta$.
2. Provar $\neg\alpha$.

Exemplo

- n^2 é par $\to$ n é par.

Técnica de prova: pela universal

Prova de uma universal

Para provar $\forall x P(x)$:

1. Supor um x arbitrário.
2. Provar $P(x)$.

Exemplo

- $\forall n \in \mathbb{N}(n \text{ é par} \rightarrow n^2 \text{ é par})$.

Técnica de prova: por contradição

Prova de uma afirmativa por contradição

Para provar α:

1. Supor um $\neg\alpha$.
2. Provar uma contradição.

Exemplo

- Existe uma infinidade de números primos.

Técnica de prova: por construção

Prova de uma existencial por construção

Para provar $\exists x \in A, P(x)$:

1. Encontrar um $a \in A$ tal que $P(a)$.
2. Provar $P(a)$.

Exemplo

- $\forall n \in \mathbb{N}, \exists k \in \mathbb{N}$, tal que k tem n divisores distintos.

Técnica de prova: por casos

Prova de uma afirmativa por casos

Para provar β:

1. Provar $\alpha_1 \vee \ldots \vee \alpha_n$.
2. Provar $\alpha_1 \to \beta, \ldots, \alpha_n \to \beta$.

Exemplo

- $\forall x, y \in \mathbb{R}, \min(x, y) + \max(x, y) = x + y$.

Veja a solução na página 10 do livro texto.

Técnica de prova: para bicondicional

Prova de uma bicondicional em duas partes

Para provar $\alpha \leftrightarrow \beta$:

1. Provar $\alpha \rightarrow \beta$.
2. Provar $\beta \rightarrow \alpha$.

Exemplo

- $\forall n \in \mathbb{N}, n$ par $\leftrightarrow n^2$ par.

Veja a solução na página 10 do livro texto.

Linguagens Formais e Autômatos

Semana 1: Representação e prova de teoremas

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Linguagens Formais e Autômatos

Semana 2: Conjuntos, relações, funções, conjuntos enumeráveis, definições recursivas, indução matemática e grafos

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

O que é um conjunto

Abstração matemática que visa capturar o conceito de coleção.
Lista não ordenada de elementos ou membros: $\{1, 2\} = \{2, 1\}$

Notação

1. $a \in A$: a pertence a A.
2. $a \notin A$: a não pertence a A.

Exemplos

- {Mércúrio, Vênus, Terra, Marte, Júpter}
- {10,Marte,{0},{Terra,1,2,3}}

Tipos de conjuntos e conjuntos importantes

- O conjunto vazio: $\emptyset$.
- Conjuntos unitário, finito, infinito.
- $\mathbb{N}$: números naturais.
- $\mathbb{Z}$: números inteiros.
- $\mathbb{R}$: números reais.
- $\mathbb{Q}$: números racionais.

Notações importantes

- $\{x \mid P(x)\}$. Exemplo: $\{k \mid k = 2n + 1 \text{ e } n \in \mathbb{N}\}$
- $\{x \in A \mid P(x)\}$. Exemplo: $\{k \in \mathbb{R} \mid 0 \leq k \leq 1\}$.

Relacionamentos entre conjuntos

Relacionamentos básicos entre conjuntos

- Subconjunto: $A \subseteq B$ se e somente se $\forall x(x \in A \rightarrow x \in B)$
- Subconjunto próprio: $A \subset B$ se e somente se $A \subseteq B$ e $A \neq B$.

Exemplos

- $\emptyset \subseteq A$
- $\emptyset \subset A$ se $A \neq \emptyset$
- $\emptyset \subseteq \emptyset$

Operações sobre conjuntos

Operações básicas entre conjuntos

- União: $A \cup B = \{x \mid x \in A \text{ ou } x \in B\}$
- Interseção: $A \cap B = \{x \mid x \in A \text{ e } x \in B\}$
- Diferença: $A - B = \{x \mid x \in A \text{ e } x \notin B\}$
- Complemento: $\overline{A} = U - A$

Exemplos de identidades

- $A \cup B = B \cup A$ (comutatividade)
- $A \cap B = B \cap A$
- $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ (distributividade)
- $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
- $\overline{A \cup B} = \overline{A} \cap \overline{B}$ (leis de *De Morgan*)
- $\overline{A \cap B} = \overline{A} \cup \overline{B}$
- $A - B = A \cap \overline{B}$

Igualdade

$A = B$ se e somente se $A \subseteq B$ e $B \subseteq A$.

Prova de igualdade de conjuntos

Para provar A=B:

- Provar $A \subseteq B$.
- Provar $B \subseteq A$.

Algumas vezes é possível provar encadeando-se $\leftrightarrow$
Exemplo

- $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

Conjuntos disjuntos

A e B são disjuntos se e somente se $A \cap B = \emptyset$

Exemplos

- $\{0, 2, 4\}$ e $\{1, 3, 5\}$.
- $\emptyset$ e A.
- A e $\overline{A}$.
- $A - B$ e $B - A$.

União e interseção generalizadas

União de n conjuntos

$$\bigcup_{i=1}^{n} A_i = A_1 \cup A_2 \cup \ldots \cup A_n.$$

Interseção de n conjuntos

$$\bigcap_{i=1}^{n} A_i = A_1 \cap A_2 \cap \ldots \cap A_n.$$

Partição

Partição de um conjunto

Uma partição de A é o conjunto $\{B_1, \ldots, B_n\}$ tal que:

- $B_i \neq \emptyset$ para $1 \leq i \leq n$
- $B_i \cap B_j = \emptyset$ para $1 \leq i < j \leq n$; e
- $\bigcup_{i=1}^{n} B_i = A$.

Quais são as partições de $\{1, 2, 3\}$?

Conjunto potência; número de elementos

Conjunto potência

Conjunto potência de A : $\mathcal{P}(A) = \{X \mid X \subseteq A\}$.

Exemplo: que conjunto é P$(\{1, 2, 3\})$?

Notação para número de elementos de A: $|A|$.
Exemplos:

- $|\emptyset| = 0$,
- $|\emptyset, 1, 2, \{1, 2, 3, 4, 5\}| = 4$,
- $|P(A)| = 2^{|A|}$.

Produto cartesiano

Pares (a, b) ou $[a, b]$ (similarmente: tripla, quádrupla, etc.)

Produto cartesiano de dois conjuntos

$A \times B = \{(a, b) \mid a \in A \text{ e } b \in B\}$

Exemplos:

- $\emptyset \times \{1, 2\} = \emptyset$
- $\{1, 2\} \times \{1, 2\} = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$.
- $|A \times B| = |A| \cdot |B|$ se A e B forem finitos.

Produto cartesiano de n conjuntos: A^n.

O que é relação

Relação de n argumentos sobre $A_1, ..., .A_n$

Um subconjunto de $A_1 \times A_2 \times ... \times A_n$.

Relação **binária**: $R \subseteq A \times B$

Domínio: A

Contradomínio: B

Imagem: $\{y | (x, y) \in R \text{ para algum } x\}$

Notação: $(x, y) \in R$ é o mesmo que xRy.

Um exemplo de relação binária

Relação $< \subseteq N \times N$:

Domínio: $\mathbb{N}$;

Contradomínio: $\mathbb{N} - \{0\}$;

Imagem: $\mathbb{N} - \{0\}$.

Propriedades

Inversa de R

$R^{-1} = \{(y, x) | (x, y) \in R\}$

Propriedades de uma relação binária $R \subseteq A \times A$

- **Reflexiva**: $\forall x \in A[xRx]$;
- **Simétrica**: $\forall x, y \in A[xRy \rightarrow yRx]$; e
- **Transitiva**: $\forall x, y, z \in A[(xRy \wedge yRz) \rightarrow xRz]$.

Quiz

Considere as relações

1. $<$ sobre $\mathbb{N}$;
2. $\leq$ sobre $\mathbb{N}$;
3. $\subseteq$ sobre $\mathcal{P}(\mathbb{N})$;
4. $\equiv$ sobre o conjunto das afirmativas da lógica proposicional.

Para cada uma, deixe nos comentários se a mesma é reflexiva, simétrica e/ou transitiva.

Relação de equivalência

Relação de equivalência

Aquela que é reflexíva, simétrica e transitiva.

$\Rightarrow$ Induz **classes de equivalência**

Exemplos:

- $(\text{mod } n) = \{(x, y) \in \mathbb{N}^2 \mid x \mod n = y \mod n\}$
- fazem aniversário no mesmo dia

Que classes de equivalência induz (mod 2)?

Fechos de uma relação

Fecho reflexivo

O **fecho reflexivo** de $R \subseteq A \times A$ é a relação S tal que:

- $R \subseteq S$;
- S é reflexiva;
- se $R \subseteq T$ e T é reflexiva, $S \subseteq T$.

O fecho reflexivo obtém-se acrescentando à relação o mínimo de elementos necessários para a tornar reflexiva.

Fechos **simétrico** e **transitivo**: análogos.

Qual o fecho reflexivo de $<$?

O que é uma função

Função parcial

Uma função $f : A \to B$ é uma relação $f \subseteq A \times B$ tal que:

$$\text{se } (x, y) \in f \text{ e } (x, z) \in f \text{ então } y = z$$

- $(x, y) \in f$ é o mesmo que $f(x) = y$
- f é **indefinida** para x se não há y tal que $f(x) = y$.
- Função total: definida para todo argumento.
- Função $f : A \to B$ de n argumentos:

  $A = A_1 \times ... \times A_n$

Exemplos de funções

- $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ (total)
- $/ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ (parcial)

Composição de funções

Composição: $g \circ f(x) = g(f(x))$.
Sejam:

$f : \mathbb{Z} \to \mathbb{N}$ tal que $f(n) = |n| + 1$

$g : \mathbb{N} \to \mathbb{Z}$ tal que $g(n) = 1 - n$

Então:

- $g \circ f : \mathbb{Z} \to \mathbb{Z}$ é tal que $(g \circ f)(n) = g(f(n)) = g(|n| + 1) = 1 - (|n| + 1) = -|n|$.
- $f \circ g : \mathbb{N} \to \mathbb{N}$ é tal que $(f \circ g)(n) = f(g(n)) = f(1 - n) = |1 - n| + 1$.

Tipos de funções

Uma função total $f : A \to B$ é:

- **Injetora**: se $\forall x, y[x \neq y \to f(x) \neq f(y)]$.
  Ex. $f : \mathbb{N} \to \mathbb{N}$ tal que $f(n) = 2n$.
- **Sobrejetora**: se B é a imagem de f.
  Ex. $g : \mathbb{Z} \to \mathbb{N}$ tal que $g(n) = |n|$.
- **Bijetora**: se é injetora e sobrejetora.
  Ex. $h : \mathbb{Z} \to \mathbb{N}$ tal que $h(n) = 2n$ se $n \geq 0$, e $h(n) = -(-2n + 1)$ se $n < 0$.

O que é conjunto enumerável

Cardinalidade

$card(A) = card(B)$ se existe uma função bijetora de A para B.

$\Rightarrow card(A) = card(B)$ se $|A| = |B|$ caso A e B sejam finitos.
$\Rightarrow A$ é infinito se existe $X \subset A$ tal que $card(X) = card(A)$.

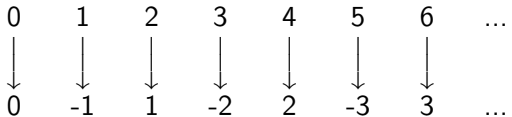
Conjunto enumerável

A é enumerável se $card(A) = card(\mathbb{N})$

Conjunto **contável**: finito e enumerável.

Um exemplo de conjunto enumerável

O conjunto $\mathbb{Z}$ é enumerável:



Um teorema facilitador

As seguintes afirmativas são equivalentes:

1. A é contável;
2. Existe função injetora de A para $\mathbb{N}$;
3. $A = \emptyset$ ou existe uma função sobrejetora de $\mathbb{N}$ para A.

Resultados importantes

1. Todo subconjunto de conjunto contável é contável;
2. $A \times B$ é contável se A e B são contáveis;
3. $A \cup B$ são contáveis se A e B são contáveis.

O que é definição recursiva

Definição recursiva do conjunto A

1. **Base**: especificação de um conjunto base $B \subset A$.
2. **Passo recursivo**: como obter elementos de A a partir de elementos de A.
3. **Fechamento**: só pertencem a A os referidos em 1. e 2.

Definição recursiva dos naturais

O conjunto $\mathbb{N}$ pode ser definido assim:

1. **Base**: $0 \in \mathbb{N}$.
2. **Passo recursivo**: se $n \in \mathbb{N}$, então $s(n) \in \mathbb{N}$.
3. **Fechamento**: só pertencem a $\mathbb{N}$ o número que pode ser obtido de acordo com 1. e 2.

Definição recursiva de fatorial

A função fatorial, $fat : \mathbb{N} \to \mathbb{N}$ pode ser definida assim:

1. $fat(0) = 1$;
2. $fat(n) = n \times fat(n-1)$, para $n \geq 1$

Indução fraca

Baseada na validade de $[P(0) \wedge \forall n(P(n) \rightarrow P(n+1))] \rightarrow \forall n P(n)$:

Princípio da indução fraca

Se

1. $P(0)$, e
2. $\forall n(P(n) \rightarrow P(n+1)))$,

então $\forall n P(n)$.

Estrutura de demonstração por indução fraca

1. Provar $P(0)$.
2. Seja $n \geq 0$ arbitrário.
3. Suponha $P(n)$ (**hipótese de indução**).
4. Provar $P(n+1)$.
5. Concluir $\forall n P(n)$.

Indução forte

Baseada na validade de $\forall n(\forall k < nP(k) \rightarrow P(n)) \rightarrow \forall nP(n)$:

Princípio da indução forte

Se

- $\forall n(\forall k < nP(k) \rightarrow P(n))$

então $\forall nP(n)$

Estrutura de demonstração por indução forte

1. Seja $n \geq 0$ arbitrário.
2. Suponha $\forall k < n P(k)$ (**hipótese de indução**).
3. Provar $P(n)$.
4. Concluir $\forall n P(n)$.

O que é grafo

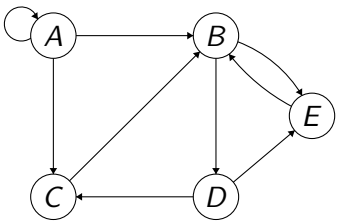
Grafo

Um grafo pode ser ser definido como uma tripla $G = (V, E, \psi)$ no qual:

1. V é o conjunto dos elementos chamados vértices
2. E é o conjunto distinto de V de elementos chamados arestas e
3. ψ é uma função que associa cada elemento de E a um par de elementos de V.

- Grafo **dirigido**: as arestas são pares **ordenados**
- Grafo **não-dirigido**: as arestas são pares **não-ordenados**

Exemplo de grafo dirigido



- Vértices: $\{A, B, C, D, E\}$
- Arestas: $\{(A, A), (A, B), (A, C), (B, D), ...\}$

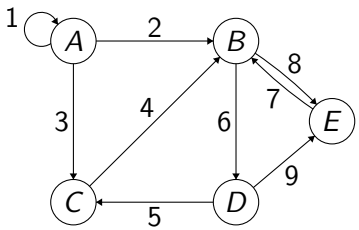
Grafos rotulados

Grafo dirigido rotulado

Um grafo dirigido rotulado pode ser definido como uma tripla $G = (V, A, R)$, no qual:

1. V é o conjunto de vértices
2. A é o conjunto de arestas rotuladas, e
3. R é um conjunto de rótulos

Exemplo de grafo dirigido rotulado



- Vértices: $\{A, B, C, D, E\}$
- Arestas: $\{(A, A, 1), (A, B, 2), (A, C, 3), (B, D, 6), ...\}$

Conceitos importantes

Grau de um vértice

Número de arestas incidentes ao vértice.

Caminho de comprimento n de a para b

Sequência de vértices e arestas $v_0 x_1 v_1 x_2 v_2 \ldots v_{n-1} x_n v_n$:

- $v_0 = a$;
- $v_n = b$; e
- $x_i = (v_{i-1}, v_i)$

Terminologia associada a caminhos

- Caminho vazio: caminho de comprimento zero.
- Caminho fechado (ciclo): aquele em que $v_0 = v_n$.
- Laço: ciclo de comprimento 1
- Grafo acíclico: grafo sem ciclos
- Grafo conexo: aquele em que existe um caminho entre quaisquer par de vértices

Árvore

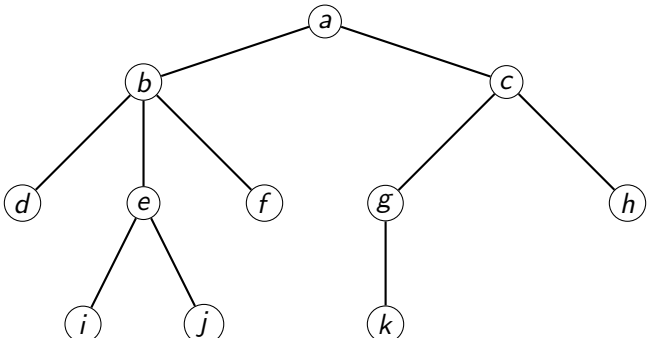
Uma árvore é um grafo acíclico conexo.

Árvore com raiz

Uma árvore com raiz é uma tripla (V, A, r) tal que:

1. $(\{v\}, \emptyset, v)$ é uma árvore;
2. se (V, A, r) é uma árvore, $v \in V$ e $v' \in V$, então $(V \cup \{v'\}, A \cup \{\{v, v'\}\}, r)$ é uma árvore;
3. nada mais é árvore.

Exemplo de árvore com raiz



Terminologia associada a árvores

- Filhos, pai, irmãos, descendente, ancestral.
- Vértice **interno**, **folha**.
- **Nível** de um vértices, **altura** da árvore.
- Árvore **dirigida**, **ordenada**.
- **Fronteira** de uma árvore.

Linguagens Formais e Autômatos

Semana 2: Conjuntos, relações, funções, conjuntos enumeráveis, definições recursivas, indução matemática e grafos

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Linguagens Formais e Autômatos

Semana 3: Linguagens formais, gramáticas e problemas de decisão

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Conceitos iniciais

Alfabeto

Conjunto finito não vazio (de símbolos).

Exemplos:

- $\{1\}$
- $\{0, 1\}$
- $\{a, b, c\}$
- Conjunto dos caracteres do teclado.

Conceitos iniciais

Definição de palavra

Uma palavra (string) em um alfabeto Σ é uma sequência finita de símbolos de Σ.

Exemplos de palavras em $\Sigma = \{0, 1\}$:

- λ (a palavra vazia), 0, 1, 00, 01, 10, 11, 000, etc.

Comprimento de uma palavra w

$|w|$ = número de símbolos de w.

Exemplos

$|\lambda| = 0$, $|0| = 1$, $|000| = 3$, $|11010| = 5$, etc, ...

Abreviação

Definição

a^n abrevia n ocorrências do caractere a em sequência.

Exemplos

- $1^0 = \lambda$
- $0^4 = 0000$
- $1^3 01^2 = 111011$
- $1^{1000} =$ foi mal...

Conjunto de todas as palavras

Definição

Σ^* é o conjunto de todas as palavras sobre Σ

Exemplos

- $\{1\}^* = \{\lambda, 1, 11, 111, 1^4, 1^5, ...\}$
- $\{0, 1\}^* = \{\lambda, 0, 1, 00, 01, 10, 11, ...\}$

Linguagem

Definição

Uma linguagem de alfabeto Σ é um subconjunto de Σ^*.

Exemplos

Linguagens de alfabeto $\{0, 1\}$

- $\emptyset$
- $\{\lambda\}$
- $\{\lambda, 0\}$
- $\{w \in \{0,1\}^* | \ 1 \leq |w| \leq 5\}$
- $\{0^n \mid n \text{ é um número primo}\}$
- $\{0^n 1^n \mid n \in \mathbb{N}\}$
- $\{0, 1\}^*$

Operações sobre conjuntos se aplicam a linguagens

Exemplos

Sejam as linguagens L_1 sobre Σ_1 e L_2 sobre Σ_2:

- $L_1 \cup L_2$ é uma linguagem sobre $\Sigma_1 \cup \Sigma_2$
- $L_1 \cap L_2$ é uma linguagem sobre $\Sigma_1 \cap \Sigma_2$
- $L_1 - L_2$ é uma linguagem sobre Σ_1
- $\overline{L_1} = \Sigma^* - L_1$ é uma linguagem sobre Σ_1
- $\mathcal{P}(L_1)$ é um conjunto de linguagens sobre Σ_1
- $\mathcal{P}(\Sigma_1^*)$ é o conjunto de todas as linguagens sobre Σ_1

Concatenação de palavras

Definição

A concatenação de $x = a_1 a_2 ... a_m$ e $y = b_1 b_2 ... b_n$ é $xy = a_1 a_2 ... a_m b_1 b_2 ... b_n$

Exemplos

Sejam $x = 001$ e $y = 10$:

- $xy = 00110$
- $\lambda w = w\lambda = w$
- $x(yz) = (xy)z = xyz$
- $wwwww = w^5$
- $w^0 = \lambda$

Uma operação e uma propriedade de palavras

Reverso

O reverso de $w = a_1 a_2 ... a_n$ é $w^R = a_n a_{n-1} ... a_1$

Exemplos

- $\lambda^R = \lambda$
- $a^R = a$
- $(abcaabb)^R = (bbaacba)$

Palíndromo

Uma palavra tal que $w = w^R$

Exemplos: λ, a, bb, ccc, aba, $baab$, $abcba$

Prefixos, sufixos e subpalavras de uma palavra

Definições

- x é prefixo de w se e somente se existe y tal que $w = xy$
- y é sufixo de w se e somente se existe x tal que $w = xy$
- z é subpalavra de w se e somente se existem x e y tal que $w = xzy$

Exemplos

- Prefixos de abc: λ, a, ab e abc
- Sufixos de abc: λ, c, bc e abc
- Subpalavras de abc: λ, a, b, c, ab, bc e abc

Concatenação de linguagens

Definição

A concatenação de L_1 e L_2 é $L_1L_2 = \{xy \mid x \in L_1 \text{ e } y \in L_2\}$

Exemplos

Sejam $L_1 = \{w \in \{0,1\}^* \mid |w| = 5\}$ e $L_2 = \{0y \mid y \in \{0,1\}^*\}$

- $\emptyset L_1 = \emptyset$
- $\{\lambda\}L_1 = L_1$
- $L_1L_1 = \{w \in \{0,1\}^* \mid |w| = 10\}$
- $L_1L_2 = \{w \in \{0,1\}^* \mid |w| \geq 6 \text{ e o sexto dígito de } w \text{ é } 0\}$
- $L_2L_1 = \{w \in \{0,1\}^* \mid |w| \geq 6 \text{ e } w \text{ começa com } 0\}$
- $L_2L_2 = \{0y \mid y \in \{0,1\}^* \text{ e } y \text{ contém no mínimo um } 0\}$

Fecho de Kleene de uma linguagem

L^n designa $LL...L$(n vezes). $L^0 = \{\lambda\}$ (por que?)

Definição

O fecho de Kleene de L, L^* é definido por:

- $\lambda \in L^*$;
- se $x \in L^*$ e $y \in L$, então $xy \in L^*$

Pode-se mostrar que:

$$L^* = \bigcup_{n \in \mathbb{N}} L^n = L^0 \cup L^1 \cup L^2 \cup \cdots = \{\lambda\} \cup L \cup LL \cup \cdots$$

Fecho de positivo de Kleene de uma linguagem

Definição

O fecho positivo de Kleene de L é $L^+ = LL^*$

Pode-se mostrar que:

$$L^+ = \bigcup_{n \geq 1} L^n = L^1 \cup L^2 \cup \cdots = L \cup LL \cup \cdots$$

Segue diretamente das definições que $L^* = L^+ \cup \{\lambda\}$

Exemplificando fechos de Kleene

Definição

O fecho positivo de Kleene de L é $L^+ = LL^*$

- $\emptyset^* = \{\lambda\}$
- $\emptyset^+ = \emptyset$
- $\{\lambda\}^* = \{\lambda\}^+ = \{\lambda\}$
- $\{0\}^* = \{0^n | n \in \mathbb{N}\}$
- $\{0\}^+ = \{0^n | n \geq 1\}$
- $\{00\}^* = \{0^{2n} | n \in \mathbb{N}\}$
- $\{00\}^+ = \{0^{2n} | n \geq 1\}$
- $\{\lambda, 00, 11\}^* = \{\lambda, 00, 11\}^+ = \{\lambda\} \cup \{00, 11\}^+$

Responda nos comentários

Descrevendo linguagens de alfabeto $\{0, 1\}$

O fecho positivo de Kleene de L é $L^+ = LL^*$

1. o conjunto de palavras que começam com o 0
2. o conjunto das palavras que contêm 00 ou 11
3. o conjunto das palavras que terminam com 0 seguido de um número ímpar de 1s consecutivos
4. o conjunto das palavras que começam ou terminam com 0
5. o conjunto das palavras de tamanho par que começam com 0 ou terminam com 0
6. o conjunto das palavras com um prefixo de um ou mais 0s seguido (imediatamente) de um sufixo de 1s de mesmo tamanho
7. o conjunto das palavras formadas por concatenações de palavras da forma 0^n1^n para $n \geq 1$.

Definição recursiva de linguagens

- Σ^* é enumerável
- Logo, linguagens podem ser definidas recursivamente. Exemplo:
  1. $\lambda \in L$;
  2. se $x \in L$ então $0x1 \in L$

Conceitos envolvidos em gramáticas

- Gramática: formalismo, que permite o uso de recursão especialmente projetado para a definição de linguagens.
- **Terminais**: alfabeto de linguagem definida
  Exemplo: $\Sigma = \{0, 1\}$
- **Variáveis** (não terminais): alfabeto de símbolos auxiliares
  Exemplo: $\Gamma = \{A, B\}$
- **Regra**: par ordenado (u, v), tradicionalmente escrito na forma $u \to v$
  Exemplo: $0AB \to 10A$

Um exemplo de derivação

Derivação (geração) de $110A10A$ a partir de $0ABB0AB$:

$$
\begin{aligned}
0ABB0AB &\Rightarrow 10AB0AB \text{ (aplicando-se a regra } 0AB \rightarrow 10A) \\
&\Rightarrow 110A0AB \text{ (aplicando-se a regra } 0AB \rightarrow 10A) \\
&\Rightarrow 110A10A \text{ (aplicando-se a regra } 0AB \rightarrow 10A)
\end{aligned}
$$

Definição Informal de Gramática

- Uma gramática é constituída de uma **variável de partida** e um **conjunto de regras**
- Toda **derivação** deve iniciar pela variável de partida
- **Forma sentencial**: palavra constituída de terminais e/ou variáveis
- **Sentença**: forma sentencial constituída de terminais apenas.
- **Linguagem gerada**: sentenças que podem ser derivadas.

Notação: $L(G)$ é a linguagem gerada pela gramática G.

Dois exemplos de gramática

A gramática de variável P e as duas regras a seguir gera $\{0\}^*$

1. $P \to 0P$
2. $P \to \lambda$

A gramática de variável P e as duas regras a seguir gera $\{0^n1^n | n \geq 0\}$

1. $P \to 0P1$
2. $P \to \lambda$

Um outro exemplo de gramática

Seja a gramática G constituída pela variável de partida P e pelas regras:

1. $P \rightarrow aAbc$
2. $A \rightarrow aAbC$
3. $A \rightarrow \lambda$
4. $Cb \rightarrow bC$
5. $Cc \rightarrow cc$

Um exemplo de derivação

$$P \Rightarrow aAbc \text{ (regra 1)}$$
$$\Rightarrow abc \text{ (regra 3)}$$

Isso mostra que $abc \in L(G)$

Outra derivação

$$
\begin{aligned}
P &\Rightarrow aAbc \text{ (regra 1)} \\
&\Rightarrow aaAbCbc \text{ (regra 2)} \\
&\Rightarrow aaaAbCbCbc \text{ (regra 2)} \\
&\Rightarrow aaabCbCbc \text{ (regra 3)} \\
&\Rightarrow aaabbCCbc \text{ (regra 4)} \\
&\Rightarrow aaabbCbCc \text{ (regra 4)}
\end{aligned}
$$

Isso mostra que $a^3b^3c^3 \in L(G)$. O que é L(G)?

Definição de gramática

Gramática

Uma gramática é uma quádrupla (V, Σ, R, P), em que:

1. V é um conjunto finito de elementos denominados variáveis;
2. Σ é um alfabeto; $V \cap \Sigma = \emptyset$
3. $R \subseteq (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$ é um conjunto finito de pares ordenados chamados regras; e
4. $P \in V$ é uma variável conhecida como variável de partida.

Derivação em n passos

Definição

A relação $\stackrel{n}{\Rightarrow}$ pode ser definida recursivamente para a gramática G:

1. $x \stackrel{0}{\Rightarrow} x$ para toda forma sentencial x de G;
2. se $w \stackrel{n}{\Rightarrow} xuy$ e $u \to v$ é regra de G, então $w \stackrel{n+1}{\Longrightarrow} xvy$

Derivação em vários passos

Derivação em zero ou mais passos

$x \overset{*}{\Longrightarrow} y$ se e somente se existe $n \geq 0$ tal que $x \overset{n}{\Longrightarrow} y$.

Derivação em um ou mais passos

$x \overset{+}{\Longrightarrow} y$ se e somente se existe $n \geq 1$ tal que $x \overset{n}{\Longrightarrow} y$.

Linguagem gerada por uma gramática

Definição

Seja $G = (V, \Sigma, R, P)$

$$L(G) = \{w \in \Sigma^* | P \overset{*}{\Rightarrow} w\}$$

Notação simplificada

Duas regras com o mesmo lado esquerdo:

$$u \to v \text{ e } u \to v'$$

podem ser escritas assim:

$$u \to v \mid v'$$

Exemplo de esquema de derivação

Seja $G = (\{P, B\}, \{a, b\}, R, P)$ em que R consta das regras:

1,2. $P \to aPb \mid Bb$

3,4. $B \to Bb \mid \lambda$

Esquema de derivação para $a^n b^{n+1+k}$, $n \geq 0, k \geq 0$:

$$
\begin{aligned}
P &\overset{n}{\Rightarrow} a^n P b^n && \text{(regra 1 } n \text{ vezes, } n \geq 0) \\
&\Rightarrow a^n B b^{n+1} && \text{(regra 2)} \\
&\overset{k}{\Rightarrow} a^n B b^{n+1+k} && \text{(regra 3 } k \text{ vezes, } k \geq 0) \\
&\Rightarrow a^n b^{n+1+k} && \text{(regra 4)}
\end{aligned}
$$

Logo, $\{a^n b^m \mid n < m\} \subseteq L(G)$

Equivalência de gramáticas

Definição

Duas gramáticas G e G' são ditas **gramáticas equivalentes** quando $L(G) = L(G')$

Um exemplo de gramática mais "prático"

$G = (V, \Sigma, R, E)$, em que:

- $V = \{E, T, N, D\}$;
- $\Sigma = \{+, -, (,), 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$;
- R contém as regras:

$$
\begin{aligned}
E &\to E + T \mid E - T \mid T \\
T &\to (E) \mid N \\
N &\to DN \mid D \\
D &\to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Recursão à esquerda

Gerando somas e/ou subtrações de Ts:

$$
\begin{aligned}
E &\Rightarrow E + T && \text{(regra } E \to E + T\text{)} \\
&\Rightarrow E - T + T && \text{(regra } E \to E - T\text{)} \\
&\Rightarrow E - T - T + T && \text{(regra } E \to E - T\text{)} \\
&\Rightarrow T - T - T + T && \text{(regra } E \to T\text{)}
\end{aligned}
$$

$\Rightarrow$ recursão à esquerda.

Recursão à direita

Gerando sequências de 4 dígitos:

$$
\begin{aligned}
N &\Rightarrow DN && \text{(regra } N \to DN\text{)} \\
&\Rightarrow DDN && \text{(regra } N \to DN\text{)} \\
&\Rightarrow DDDN && \text{(regra } N \to DN\text{)} \\
&\Rightarrow DDDDN && \text{(regra } N \to D\text{)}
\end{aligned}
$$

$\Rightarrow$ recursão à direita.

Recursão indireta

$$
\begin{aligned}
E &\Rightarrow E + T && (\text{regra } E \to E + T) \\
&\Rightarrow T + T && (\text{regra } E \to T) \\
&\Rightarrow (E) + T && (\text{regra } T \to (E))
\end{aligned}
$$

$\Rightarrow$ A variável E reaparece (recursivamente) na forma sentencial entre "(" e ")".

O que é problema de decisão?

Definição

Um problema de decisão é uma questão que faz referência a um conjunto finito de parâmetros e que, para valores específicos dos parâmetros, tem como resposta sim ou não.

Exemplos

1. determinar se o número 123654789017 é um número primo;
2. determinar se um número natural n é um numero primo;
3. determinar se existe um ciclo em um grafo G;
4. determinar se uma palavra w é gerada por uma gramática G

Instâncias de um problema de decisão

Definição

Uma instância de um problema de decisão é uma questão obtida dando aos parâmetros valores específicos.

Exemplos

1. O problema de decisão "determinar se um número natural n é um numero primo" tem um conjunto infinito de instâncias:
   - determinar se 0 é um número primo;
   - determinar se 1 é um número primo;
   - determinar se 2 é um número primo;
   - e assim por diante.
2. O problema de decisão "determinar se o número 123654789017 é um número primo" tem uma única instância.

Solução para um problema de decisão

Definição

Uma solução para um problema de decisão, denominada de **procedimento de decisão**, é um algoritmo que, para qualquer instância do problema de decisão, retorna a resposta correta.

O que é **algoritmo**?

Problema decidível

Definição

Um problema de decisão que tem solução é dito ser **decidível**, e um PD que não tem solução, **indecidível**.

$\Rightarrow$ todo problema de decisão com um conjunto finito de instâncias é decidível!

Restrição de um problema de decisão

Definição

Uma restrição de um problema de decisão P é aquele que se obtém restringindo-se o conjunto de valores possíveis de um ou mais parâmetros de P.

Exemplos

- "determinar se 123654789017 é um número primo" é uma restrição de "determinar se um número natural n é um número primo".
- "determinar se uma palavra w é gerada por uma gramática G_0 em que o lado esquerdo de cada regra tem apenas uma variável", é uma restrição de "determinar se uma palavra w é gerada por uma gramática G".

Relação entre problemas de decisão e linguagens

- As instâncias de um problema de decisão P podem ser expressas em uma linguagem:

$$L_P = \{w \in \Sigma^* \mid \ w \text{ representa uma instância de } P\}$$

- Linguagem constituída das instâncias em que a resposta é "sim":

$$L_P^S = \{w \in L_P \mid \ \text{a resposta para } w \text{ é sim}\}$$

- Determinar se P tem solução é equivalente a:

$$\text{dada } w \in L_P, \text{ determinar se } w \in L_P^S$$

Relação entre problemas de decisão e linguagens

Exemplo

PD: determinar se $n \in \mathbb{N}$ é primo. Representando-se cada instância por uma palavra binária, temos:

- $L_{PD} = \{0,1\}^+$;
- $L_{PP}^S = \{w \in \{0,1\}^+ \mid \eta(w) \text{ é primo}\}$

em que $\eta(w)$ é o número representado por w. Com isto:

$$n \text{ é primo se e somente se } w \in L_{PD}, \text{ em que } \eta(w) = n.$$

Linguagens Formais e Autômatos

Semana 3: Linguagens formais, gramáticas e problemas de decisão

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Linguagens Formais e Autômatos

Semana 4: Autômatos Finitos

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Quebra-cabeça

O homem, o leão, o coelho e o repolho

Um homem, um leão, um coelho e um repolho devem atravessar um rio usando uma canoa, com a restrição de que o homem deve transportar no máximo um dos três de cada vez de uma margem à outra. Além disso, o leão não pode ficar na mesma margem que o coelho sem a presença do homem, e o coelho não pode ficar com o repolho sem a presença do homem. O problema consiste em determinar se é possível fazer a travessia, e caso seja, a sequência de movimentações que propicie a travessia.

Modelagem de um problema

Na fase de modelagem:

1. As informações relevantes são identificadas (abstração);
2. As informações relevantes são estruturadas (representadas), de forma a facilitar a posterior solução.

$\Rightarrow$ Menos informações $\rightarrow$ solução mais fácil e/ou eficiente.

Modelagem do quebra-cabeça

Informações relevantes abstraídas para o quebra-cabeças:

1. em um dado instante, em que margem do rio estão o homem, o leão, o coelho, e o repolho;
2. a sequência de movimentações entre as margens que propiciou a situação indicada em 1.

Representação das informações:

- $A \subseteq \{\texttt{h},\texttt{l},\texttt{c},\texttt{r}\}$ representa que os elementos em A estão na margem esquerda e os em $\{\texttt{h},\texttt{l},\texttt{c},\texttt{r}\} - A$ na margem direita;
- palavra $a_0 a_1 a_2 ... a_n$, em que cada a_i pode ser `s`,`l`,`c` ou `r`, representa a sequência de movimentos até o momento.

Modelagem por meio de estados e transições

- Estado: uma fotografia da realidade
- Transição de um estado para o outro: provocada por uma ação
- Solução: sequência de ações que levam de um estado inicial a um estado final

Para o quebra cabeça:

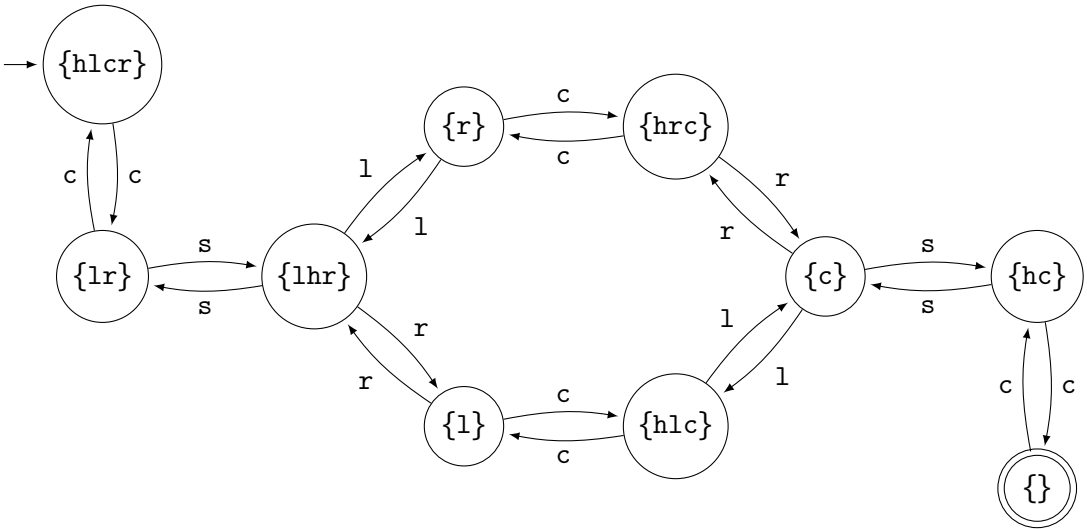
- Estado: um de subconjunto de $\{\texttt{h},\texttt{l},\texttt{c},\texttt{r}\}$
  - inicial: $\{\texttt{h},\texttt{l},\texttt{c},\texttt{r}\}$
  - final: $\{\}$
- Transição: provocada por uma das ações: `s`,`l`,`c` ou `r`.
- Solução: uma palavra no alfabeto $\{\texttt{s},\texttt{l},\texttt{c},\texttt{r}\}$

Diagrama de estados

Representação dos estados e transições por meio de um grafo:

- Estado: vértice do grafo
  - inicial: ressaltado por uma seta que o aponta
  - final: oval dupla
- Transição de e para e' provocada por $\mathtt{s}$: aresta de e para e' com rótulo $\mathtt{s}$.

Diagrama de estados para o quebra-cabeça



Computação

Dada $w \in \{\texttt{s},\texttt{l},\texttt{c},\texttt{r}\}^*$:

- w é reconhecida se o "caminho correspondente" vai do estado inicial ao final.
- w é rejeitada, caso contrário.

Configuração instantânea durante processamento de $w : [e, y]$, sendo:

- e: o estado atual após processar um prefixo x de w;
- y: o sufixo ainda não processado (assim, $w = xy$).

Relação "resulta em" $\vdash$: Dizemos que $[e_1, w] \vdash [e_2, y]$ se existe uma transição de e_1 para e_2 sob a e $w = ay$.

Exemplo de computação:

$[\{\texttt{l, r}\}, \texttt{sllr}] \vdash [\{\texttt{h, l, r}\}, \texttt{llr}] \vdash [\{\texttt{r}\}, \texttt{lr}] \vdash [\{\texttt{h, l, r}\}, \texttt{r}] \vdash [\{\texttt{l}\}, \lambda]$

Características do autômato do quebra-cabeças

- para cada transição existe uma transição inversa;
- há um único estado final;
- **determinismo**: para cada par (estado, símbolo) existe, no máximo, uma transição;
- o conjunto de estados é **finito**.

$\Rightarrow$ Única característica geral: a última.

O que é autômato finito determinístico

AFD

Um autômato finito determinístico (AFD) é uma quíntupla $(E, \Sigma, \delta, i, F)$ em que:

- E é um conjunto finito não vazio de estados;
- Σ é um alfabeto;
- $\delta : E \times \Sigma \to E$ é a função de transição, uma função total;
- $i \in E$ é o estado inicial;
- $F \subseteq E$ é o conjunto de estados finais.

Propriedades dos AFDs

- Determinismo: a partir do estado inicial é atingido um único estado, para uma dada palavra de entrada.
- Função de transição total: para toda palavra de entrada, atinge-se um estado consumindo-se toda a palavra.
- Um único estado inicial: com vários o poder computacional não é maior.
- Vários estados finais: com um só, o poder computacional é menor.
- Conjunto finito de estados: com conjunto infinito, o poder computacional é maior.

AFD para o quebra-cabeças

$$M = (E, \{s, l, c, r\}, \delta, \{h, l, c, r\}, \{\{\}\})$$

$$E = \{\{h, l, c, r\}, \{l, r\}, \{h, l, r\}, \{l\}, \{r\}, \{h, l, c\}, \{h, c, r\}, \{c\}, \{h, c\}, \{\}, t\}$$

δ	s	l	c	r
$\{h, l, c, r\}$	t	t	$\{l, r\}$	t
$\{l, r\}$	{h,l,r}	t	{h,l,c,r}	t
$\{h, l, r\}$	{l,r}	{r}	t	{l}
$\{l\}$	t	t	{h,l,c}	{h,l,r}
$\{r\}$	t	{h,l,r}	{h,c,r}	t
{h,l,c}	t	{c}	{l}	t
{h,c,r}	t	t	{r}	{c}
{c}	{h,c}	{h,l,c}	t	{h,c,r}
{h,c}	c	t	{}	t
{}	t	t	{h,c}	t
t	t	t	t	t

Uma convenção em diagramas de estados

Se não há transição de e sob a no diagrama de estados, então existe e' (estado de **erro**) tal que:

- existe uma transição de e para e' sob a;
- e' não é estado final;
- existe uma transição de e' para e' sob cada símbolo do alfabeto.

Diagrama de estados simplificado: aquele em que foram omitidos todos os estados de erro porventura existentes.

Função de transição estendida

Seja um AFD $M = (E, \Sigma, \delta, i, F)$. A função de transição estendida para $M, \hat{\delta} : E \times \Sigma^* \to E$, é definida recursivamente como segue:

1. $\hat{\delta}(e, \lambda) = e$;
2. $\hat{\delta}(e, ay) = \hat{\delta}(\delta(e, a), y)$, para todo $a \in \Sigma$ e $y \in \Sigma^*$.

Linguagem reconhecida por um AFD

A linguagem reconhecida por um AFD

A linguagem reconhecida por um AFD $M = (E, \Sigma, \delta, i, F)$ é:

$$L(M) = \{w \in \Sigma^* \mid \hat{\delta}(i, w) \in F\}.$$

Uma palavra $w \in \Sigma^*$ é dita ser reconhecida por M se $\hat{\delta}(i, w) \in F$.

AFDs equivalentes

Dois AFDs, M_1 e M_2, são ditos equivalentes se, e somente se, $L(M_1) = L(M_2)$.

Algoritmo para simular AFDs

Entrada: 1. o AFD, dado por i, F e D, e
2. a palavra de entrada, dada por *prox*.
Saída: sim ou não
$e \leftarrow i$;
$s \leftarrow prox()$;
enquanto $s \neq fs$ **faça**
  $e \leftarrow D[e, s]$;
  $s \leftarrow prox()$;
fim
se $e \in F$ **então**
  retorna *sim*
fim
senão
  retorna *não*
fim

Desempenho: $O(n)$ (n : número de símbolos da palavra).

Dois exemplos

Vamos construir AFDs que reconheçam:

- $\{w \in \{0,1\}^* \mid w \text{ tem número par de símbolos}\}$.

Dois exemplos

Vamos construir AFDs que reconheçam:

- $\{w \in \{0,1\}^* \mid w$ tem número par de símbolos$\}$.
- $\{w \in \{0,1\}^* \mid w$ tem número par de 0s e número par de 1s$\}$.

Duas perguntas

- Existe AFD mínimo?
- Há como construir um AFD, a partir de AFDs M_1 e M_2, que reconheça:
  - $L(M_1) \cup L(M_2)$?
  - $L(M_1) \cap L(M_2)$?
  - $L(M_1) - L(M_2)$?
  - $L(M_1)L(M_2)$?
  - $L(M1)^*$

$\Rightarrow$

Duas perguntas

- Existe AFD mínimo?
- Há como construir um AFD, a partir de AFDs M_1 e M_2, que reconheça:
  - $L(M_1) \cup L(M_2)$?
  - $L(M_1) \cap L(M_2)$?
  - $L(M_1) - L(M_2)$?
  - $L(M_1)L(M_2)$?
  - $L(M1)^*$

$\Rightarrow$ Resposta para ambas:

Duas perguntas

- Existe AFD mínimo?
- Há como construir um AFD, a partir de AFDs M_1 e M_2, que reconheça:
  - $L(M_1) \cup L(M_2)$?
  - $L(M_1) \cap L(M_2)$?
  - $L(M_1) - L(M_2)$?
  - $L(M_1)L(M_2)$?
  - $L(M1)^*$

$\Rightarrow$ Resposta para ambas: SIM!!!

O que é AFD mínimo?

AFD mínimo

Um AFD M é dito ser um AFD mínimo para a linguagem $L(M)$ se nenhum AFD para $L(M)$ contém menor número de estados que M.

Para obter um AFD mínimo:

1. Eliminar estados não alcançáveis a partir do estado inicial.
2. Substituir cada grupo de **estados equivalentes** por um único estado.

Equivalência de estados

Estados equivalentes

Seja um AFD $M = (E, \Sigma, \delta, i, F)$. Então $e, e' \in E$ são ditos equivalentes, $e \approx e'$, se, e somente se:

$$\forall y \in \Sigma^*, \hat{\delta}(e, y) \in F \leftrightarrow \hat{\delta}(e', y) \in F.$$

Por que reduzir estados equivalentes a um só?

Seja um AFD $M = (E, \sigma, \delta, i, F)$.

1. se $e \approx e'$: um sufixo y é reconhecido passando-se por e, se, e somente se, ele é reconhecido passando-se por e'; logo e e e' podem se tornar um só!
2. se $e \not\approx e'$ (existe $y \in \Sigma^*, \hat{\delta}(e, y) \in F$ e $\hat{\delta}(e', y) \neq F$ ou vice-versa): se um sufixo y levar a estado de F , a palavra é aceita, caso contrário, não é. Logo, e e e' não podem se tornar um só!

O conceito de autômato reduzido

[e]: classe de equivalência de e na partição induzida por $\approx$.

AFD reduzido

Seja um AFD $M = (E, \Sigma, \delta, i, F)$. Um autômato reduzido correspondente a M é o AFD $M' = (E', \Sigma, \delta', i', F')$, em que:

- $E' = \{[e] \mid e \in E\}$;
- $\delta'([e], a) = [\delta(e, a)], \forall e \in E, \forall a \in \Sigma$;
- $i' = [i]$;
- $F' = \{[e] \mid e \in F\}$.

Obtenção de $\approx$ passo a passo

Seja um AFD $M = (E, \Sigma, \delta, i, F)$ e um estado $e \in E$. Então:

1. $e \approx_0 e'$ se,e somente se, $(e, e' \in F$ ou $e, e' \in E - F)$;
2. para $n \geq 0 : e \approx_{n+1} e'$ se, e somente se, $e \approx_n e'$ e $\delta(e, a) \approx_n \delta(e', a), \forall a \in \Sigma$.

Teorema que justifica $\approx_n$:

Seja um AFD $M = (E, \Sigma, \delta, i, F)$. Então $e \approx_n e'$ se, e somente se, para todo $w \in \Sigma^*$ tal que $|w| \leq n$, $\hat{\delta}(e, w) \in F \leftrightarrow \hat{\delta}(e', w) \in F$.

Obtenção das partições $[e]_{\approx}$ passo a passo

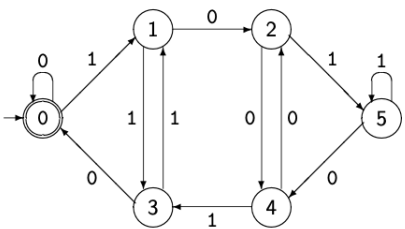
$[e]_n$: classe de equivalência de e na partição induzida por $\approx_n$.

Obtenção de cada $[e]_n$:

1. $[e]_0 = \begin{cases} F & \text{se } e \in F \\ E - F & \text{se } e \in E - F \end{cases}$
2. para $n \geq 0, [e]_{n+1} = \{e' \in [e]_n \mid [\delta(e', a)]_n = [\delta(e, a)]_n, \forall a \in \Sigma\}$.

Um exemplo de minimização

Autômato inicial

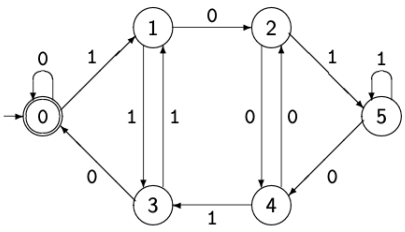


Evolução das partições

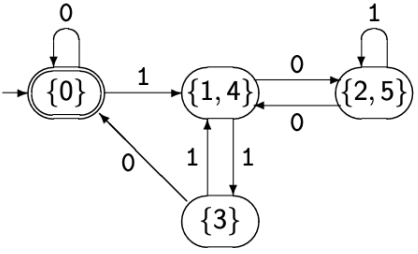
S_0: $\{0\}, \{1,2,3,4,5\}$
S_1: $\{0\}, \{1,2,4,5\}, \{3\}$
S_2: $\{0\}, \{1,4\}, \{2,5\}, \{3\}$
S_3: $\{0\}, \{1,4\}, \{2,5\}, \{3\}$

Um exemplo de minimização

Autômato inicial



Autômato reduzido



Produto de AFDs

Simulação do funcionamento em paralelo de dois AFDs:

- Sejam $M_1 = (E_1, \Sigma, \delta_1, i_1, F_1)$ e $M_2 = (E_2, \Sigma, \delta_2, i_2, F_2)$.
- $M_3 = (E_3, \Sigma, \delta_3, i_3, F_3)$, em que:
  - $E_3 = E_1 \times E_2$;
  - $\delta_3([e_1, e_2], a) = [\delta_1(e_1, a), \delta_2(e_2, a)], \forall e_1 \in E_1, \forall e_2 \in E_2, \forall a \in \Sigma$;
  - $i_3 = [i_1, i_2]$;
  - F_3: depende do que se quer para M_3.

Sejam $e_1 \in E_1$ e $e_2 \in E_2$ de M_3. Então,

$$\hat{\delta}([e_1, e_2], w) = [\hat{\delta}(e_1, w), \hat{\delta}(e_2, w)], \forall w \in \Sigma^*.$$

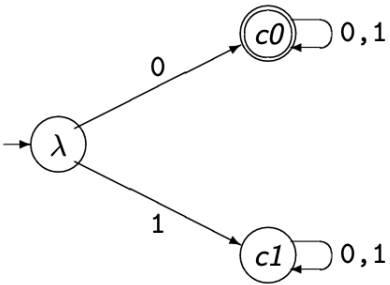
Complemento, interseção e união

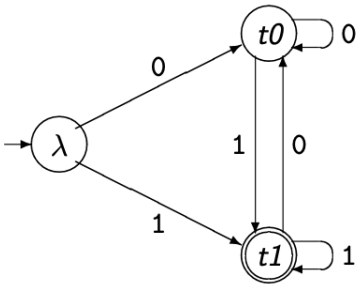
Sejam $M_1 = (E_1, \Sigma, \delta_1, i_1, F_1)$ e $M_2 = (E_2, \Sigma, \delta_2, i_2, F_2)$.

1. AFD para $\overline{L(M_1)}$:
   $(E_1, \Sigma, \delta_1, i_1, E_1 - F_1)$.
2. $L(M_1) \cap L(M_2)$:
   produto de M_1 e M_2 com $F_3 = F_1 \times F_2$.
3. $L(M_1) \cup L(M_2)$:
   produto de M_1 e M_2 com $F_3 = (F_1 \times E_2) \cup (E_1 \times F_2)$

Exemplo de produto/interseção e união

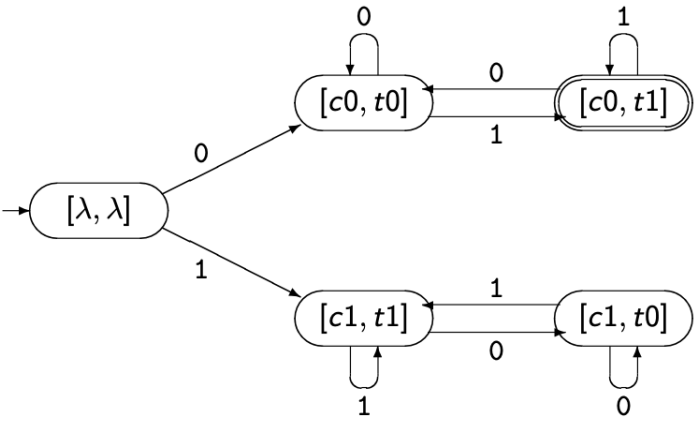
Reconhecendo $\{0\}\{0,1\}^*$ e $\{0,1\}^*\{1\}$:





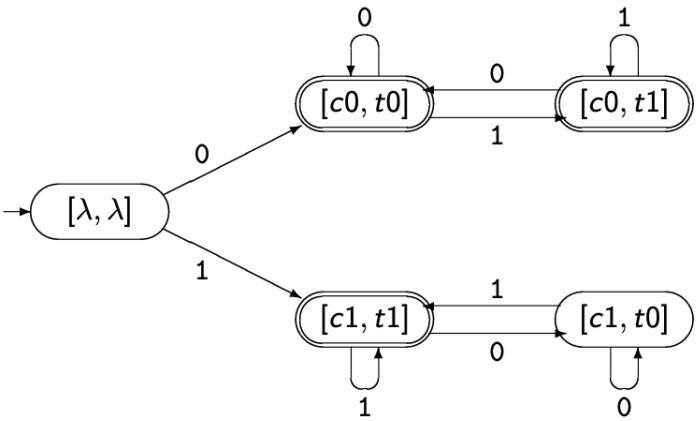
Exemplo de produto/interseção

Reconhecendo $\{0\}\{0,1\}^* \cap \{0,1\}^*\{1\}$:



Exemplo de produto/interseção

Reconhecendo $\{0\}\{0,1\}^* \cup \{0,1\}^*\{1\}$:



Linguagens finitas

- Para toda linguagem finita existe um AFD.
- AFD com diagrama de estados simplificado sem ciclos: árvore em que o estado inicial é a raiz.

Exemplo: Um pequeno dicionário:
$K = \{\texttt{a}, \texttt{alma}, \texttt{asa}, \texttt{barco}, \texttt{brasa}, \texttt{broa}, \texttt{ca}, \texttt{calma}, \texttt{casa}, \texttt{disco}\}$

Alguns problemas de decisão envolvendo AFDs

Existem procedimentos de decisão para determinar, para qualquer AFD M, se:

1. $L(M) = \emptyset$;
2. $L(M)$ é finita.

Seja M' um AFD mínimo equivalente a M. Então:

1. $L(M) = \emptyset$ se, e somente se, M' não tiver estados finais;
2. $L(M)$ é finita se, e somente se, o diagrama de estados simplificado de M' não possui ciclo.

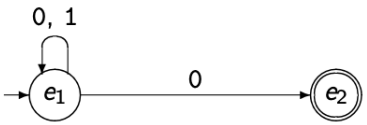
O que é Autômato Finito Não-Determinístico (AFN)

Definição de AFN

Um AFN é uma quíntupla $(E, \Sigma, \delta, I, F)$, em que:

- E é um conjunto finito não vazio de estados;
- Σ é um alfabeto;
- $I \subseteq E$ é um conjunto não vazio de estados iniciais;
- $F \subseteq E$ é um conjunto de estados finais;
- $\delta : E \times \Sigma \to \mathcal{P}(E)$ é a função de transição, uma função total.

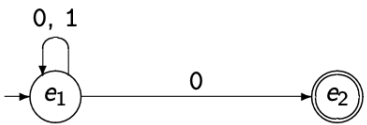
Exemplo de AFN



$$(\{e_1, e_2\}, \{0, 1\}, \delta, \{e_1\}, \{e_2\})$$

δ	0	1
e_1	$\{e_1, e_2\}$	$\{e_1\}$
e_2	$\emptyset$	$\emptyset$

Exemplo de autômato finito não determinístico (AFN)



1. Não determinismo: no estado e_1 existem **duas** transições possíveis sob o símbolo 0.
2. Computações possíveis para a palavra 1010:

$$[e_1, 1010] \vdash [e_1, 010] \vdash \begin{cases} [e_1, 10] \vdash [e_1, 0] \vdash \begin{cases} [e_1, \lambda] \\ [e_2, \lambda] \end{cases} \\ [e_2, 10] \end{cases}$$

Reconhecimento para AFN

- Uma palavra é reconhecida se, e somente se, existe uma computação que a consome e termina em estado final;
- Em todo ponto de indecisão, a máquina adivinha qual escolha (se houver alguma) leva a uma computação que resulta em sucesso no reconhecimento.

Linguagem reconhecida por um AFN

A função de transição estendida para um AFN $M = (E, \Sigma, \delta, I, F)$, $\hat{\delta} : \mathcal{P}(E) \times \Sigma^* \to \mathcal{P}(E)$, é definida recursivamente como segue:

- $\hat{\delta}(\emptyset, w) = \emptyset, \forall w \in \Sigma^*$;
- $\hat{\delta}(A, \lambda) = A, \forall A \subseteq E$;
- $\hat{\delta}(A, ay) = \hat{\delta}(\bigcup_{e \in A} \delta(e, a), y)$, para $A \subseteq E, a \in \Sigma$ e $y \in \Sigma^*$.

Linguagem reconhecida por um AFN

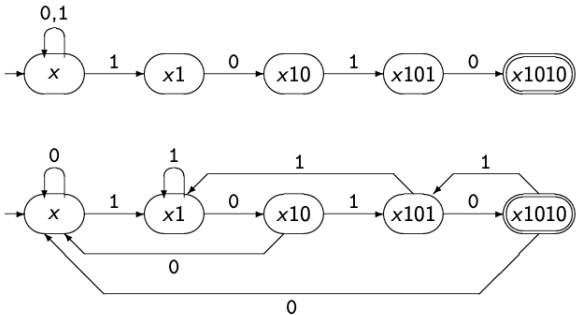
Seja $M = (E, \Sigma, \delta, I, F)$.

$$L(M) = \{w \in \Sigma^* \mid \hat{\delta}(I, w) \cap F \neq \emptyset\}$$

.

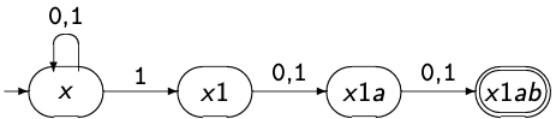
Exemplo 1: AFN × AFD

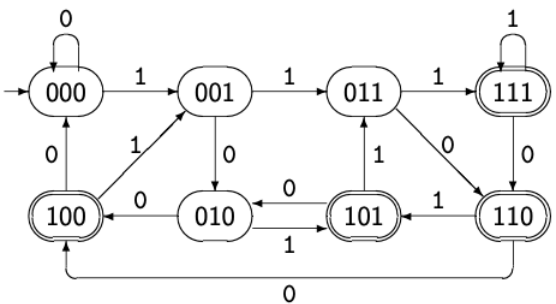
AFN e AFD que aceitam $\{0,1\}^*\{1010\}$:



Exemplo 2: AFN × AFD

AFN e AFD que aceitam $\{w \in \{0,1\}^* \mid |w| \geq 3$ e o terceiro símbolo da direita para a esquerda é $1\}$:





Equivalência entre AFDs e AFNs

Para qualquer AFN existe um AFD equivalente.

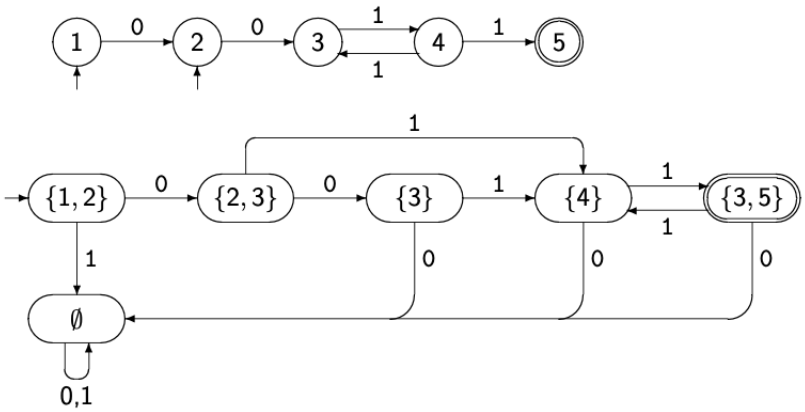
Idéia: Um estado será um conjunto, significando todos os estados do AFN atingidos por todas as computações possíveis para a mesma palavra.

Um AFD equivalente a um AFN $M = (E, \Sigma, \delta, I, F)$, é:
$M' = (\mathcal{P}(E), \Sigma, \delta', I, F')$, em que:

- para cada $X \subseteq E$ e $a \in \Sigma$, $\delta'(X, a) = \bigcup_{e \in X} \delta(e, a)$;
- $F' = \{X \subseteq E \mid X \cap F \neq \emptyset\}$.

Equivalência entre AFDs e AFNs

Diagrama de estados de um AFN e do AFD equivalente.



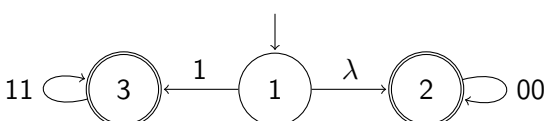
O que é AFN estendido

AFN estendido

Um AFN estendido é uma quíntupla $(E, \Sigma, \delta, I, F)$, em que :

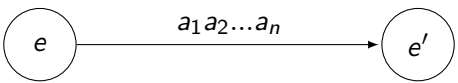
- E, Σ, I e F são como em AFNs; e
- δ é uma função parcial de $E \times D$ para $\mathcal{P}(E)$, em que D é algum subconjunto finito de Σ^*.

Exemplo: $L = \{w \in \{0\}^* \mid |w| \text{ é par}\} \cup \{w \in \{1\}^* \mid |w| \text{ é ímpar}\}$

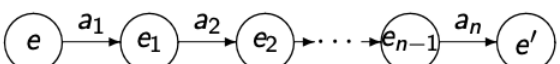


Tirando transições sob palavras de mais de um símbolo

Uma transição da forma



pode ser substituída por n transições:



AFN com transições λ

Definição de AFNλ

Um AFNλ é uma quíntupla $(E, \Sigma, \delta, I, F)$, em que:

- E, Σ, I e F são como em AFNs; e
- δ é uma função total de $E \times (\Sigma \cup \{\lambda\})$ para $\mathcal{P}(E)$.

A função fecho λ

Seja um $AFN\lambda$ $M = (E, \Sigma, \delta, I, F)$.

A função **fecho** λ para M, $f\lambda : \mathcal{P}(E) \to \mathcal{P}(E)$, é definida recursivamente como:

- $X \subseteq f\lambda(X)$;
- se $e \in f\lambda(X)$, então $\delta(e, \lambda) \subseteq f\lambda(X)$.

A função de transição estendida

Seja um $AFN\lambda$ $M = (E, \Sigma, \delta, I, F)$.

A **função de transição estendida**, $\hat{\delta} : \mathcal{P}(E) \times \Sigma^* \to \mathcal{P}(E)$, é definida recursivamente como:

1. $\hat{\delta}(\emptyset, w) = \emptyset, \forall w \in \Sigma^*$;
2. $\hat{\delta}(A, \lambda) = f\lambda(A), \forall A \subseteq E$;
3. $\hat{\delta}(A, ay) = \hat{\delta}(\bigcup_{e \in f\lambda(A)} \delta(e, a), y), \forall A \subseteq E, \forall a \in \Sigma, \forall y \in \Sigma^*$.

Linguagem reconhecida por um AFNλ

Seja $M = (E, \Sigma, \delta, I, F)$.

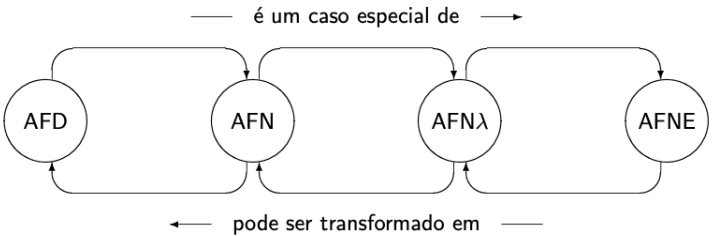
$$L(M) = \{w \in \Sigma^* \mid \hat{\delta}(I, w) \cap F \neq \emptyset\}.$$

Obtenção de AFN equivalente a AFNλ

Seja um AFNλ $M = (E, \Sigma, \delta, I, F)$.
Um AFN equivalente a M é $M' = (E, \Sigma, \delta', I', F)$, em que:

- $I' = f\lambda(I)$; e
- $\delta'(e, a) = f\lambda(\delta(e, a)), \forall e \in E$ e $\forall a \in \Sigma$.

Relações entre autômatos finitos



Linguagens Formais e Autômatos

Semana 5: Revisão e exercícios

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Linguagens Formais e Autômatos

Semana 6: O Lema do Bombeamento para Linguagens Regulares

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

O que é Linguagem Regular

Linguagem regular

Uma linguagem é dita ser uma linguagem regular se existe um autômato finito que a reconhece.

Dada uma linguagem L:

- É possível determinar se ela pertence ou não à classe das linguagens regulares?
- É possível facilitar a obtenção de um AF para L?

Um teorema sobre linguagens regulares

O Lema do Bombeamento

Seja L uma linguagem regular. Então existe uma constante $k > 0$ tal que para qualquer palavra $z \in L$ com $|z| \geq k$ existem u, v e w que satisfazem as seguintes condições:

- $z = uvw$;
- $|uv| \leq k$;
- $v \neq \lambda$; e
- $uv^i w \in L$ para todo $i \geq 0$.

Uma aplicação do Lema do Bombeamento

O LB pode ser usado para provar que uma linguagem infinita, L, não é regular da seguinte forma:

1. supõe-se que L seja linguagem regular;
2. supõe-se $k > 0$, a "constante do LB";
3. escolhe-se uma palavra z tal que $|z| \geq k$;
4. mostra-se que, para toda decomposição de z em u v e w tal que $|uv| \leq k$, e $v \neq \lambda$, existe i tal que $uv^i w \notin L$.

Exemplo de uso do lema do bombeamento

Demonstração que $L = \{a^n b^n \mid n \in \mathbf{N}\}$ não é regular

Suponha que L seja uma linguagem regular. Seja k a constante do LB e $z = a^k b^k$. Como $|z| > k$, o lema diz que existem u, v e w tais que:

- $z = uvw$;
- $|uv| \leq k$;
- $v \neq \lambda$; e
- $uv^i w \in L$ para todo $i \geq 0$.

Neste caso, v só tem as, pois $uvw = a^k b^k$ e $|uv| \leq k$, e v tem pelo menos um a, porque $v \neq \lambda$. Isso implica que $uv^2 w = a^{k+|v|} b^k \notin L$, o que contradiz o LB. Portanto, L não é linguagem regular.

Outro exemplo de uso do lema do bombeamento

Demonstração que $L = \{0^m 1^n \mid m > n\}$ não é regular

Suponha que L seja uma linguagem regular. Seja k a constante do LB, e seja $z = 0^{k+1}1^k$. Como $|z| > k$, o lema diz que existem u, v e w tais que:

- $z = uvw$;
- $|uv| \leq k$;
- $v \neq \lambda$; e
- $uv^i w \in L$ para todo $i \geq 0$.

Como $uvw = 0^{k+1}1^k$ e $0 < |v| \leq k$, v só tem 0s e possui no mínimo um 0. Logo, $uv^0 w = 0^{k+1-|v|}1^k \notin L$, contradizendo o LB. Portanto,L não é regular.

Mais um exemplo de uso do lema do bombeamento

Demonstração que $L = \{0^n \mid n \text{ é primo}\}$ não é regular

Suponha que L seja regular. Seja k a constante do LB, e seja $z = 0^n$, em que n é um número primo maior que k. Como $|z| > k$, para provar que L não é regular, basta mostrar um i tal que $uv^iw \notin L$ supondo que $z = uvw$, $|uv| \leq k$ e $v \neq \lambda$. Como $z = 0^n$, $uv^iw = 0^{n+(i-1)|v|}$. Assim, i deve ser tal que $n + (i-1)|v|$ não seja um número primo. Ora, para isso, basta fazer $i = n + 1$, obtendo-se $n + (i-1)|v| = n + n|v| = n(1 + |v|)$, que não é primo (pois $|v| > 0$). Desse modo, $uv^{n+1}w \notin L$, contradizendo o LB. Logo, L não é linguagem regular.

Algumas propriedades de fechamento

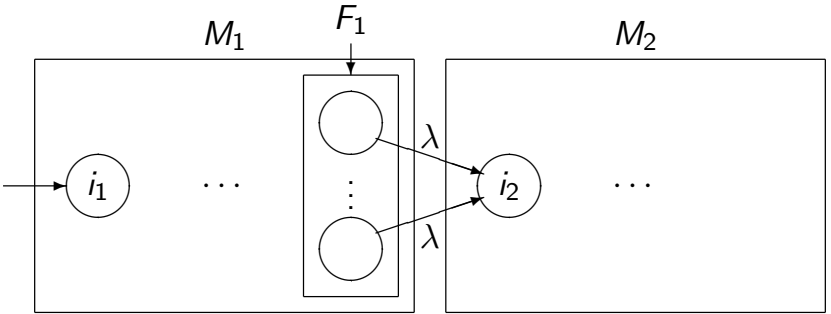
O que é fechamento

Seja uma classe de linguagens, $\mathcal{L}$, e uma operação sobre linguagens, O. Diz-se que $\mathcal{L}$ é *fechada* sob O se a aplicação de O a linguagens de $\mathcal{L}$ resulta sempre em uma linguagem de $\mathcal{L}$.

A classe das linguagens regulares é fechada sob:

1. complementação;
2. união;
3. interseção;
4. concatenação;
5. fecho de Kleene.

Fechamento sob concatenação/esquema



Fechamento sob concatenação

Sejam dois AFDS:

$$M_1 = (E_1, \Sigma_1, \delta_1, i_1, F_1) \text{ e } M_2 = (E_2, \Sigma_2, \delta_2, i_2, F_2),\ E_1 \cap E_2 = \emptyset.$$

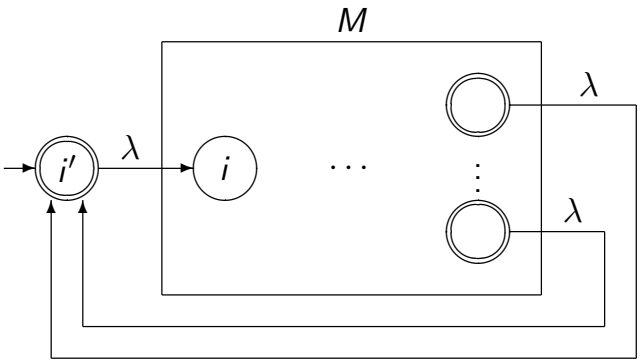
O AFNλ M_3 reconhece $L(M_1)L(M_2)$:

$$M_3 = (E_1 \cup E_2, \Sigma_1 \cup \Sigma_2, \delta_3, \{i_1\}, F_2)$$

em que δ_3 é dada por:

- $\delta_3(e, a) = \{\delta_1(e, a)\}$ para todo $e \in E_1$ e $a \in \Sigma_1$;
- $\delta_3(e, a) = \{\delta_2(e, a)\}$ para todo $e \in E_2$ e $a \in \Sigma_2$;
- $\delta_3(e, \lambda) = \{i_2\}$ para todo $e \in F_1$, e
  $\delta_3(e, \lambda) = \emptyset$ para $e \in (E_1 \cup E_2) - F_1$.

Fechamento sob fecho de kleene/esquema



Fechamento sob fecho de kleene

Seja um AFD $M = (E, \Sigma, \delta, i, F)$.

O AFNλ M' reconhece $L(M)^*$:

$$M' = (E \cup \{i'\}, \Sigma, \delta', \{i'\}, F \cup \{i'\})$$

em que $i' \notin E$, e δ' é dada por:

- $\delta'(i', \lambda) = \{i\}$;
- $\delta'(e, a) = \{\delta(e, a)\}$ para todo $e \in E$ e $a \in \Sigma$;
- $\delta'(e, \lambda) = \{i'\}$ para todo $e \in F$, e
  $\delta'(e, \lambda) = \emptyset$ para $e \in E - F$.

Aplicações das propriedades de fechamento

Três aplicações das propriedades de fecho das linguagens regulares:

1. provar que uma linguagem é regular;
2. provar que uma linguagem não é regular;
3. facilitar a obtenção de AF para uma linguagem regular.

Linguagens Formais e Autômatos

Semana 6: O Lema do Bombeamento para Linguagens Regulares

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Linguagens Formais e Autômatos

Semana 7: Expressões Regulares

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Denotação e geração de linguagens regulares

Duas novas formas de especificar as linguagens regulares: expressões regulares e gramáticas regulares.

- **Expressão Regular:** especifica uma linguagem por meio de uma expressão que a denota.
- **Gramática Regular:** especifica uma linguagem por meio de um conjunto de regras que a gera.

O que é expressão regular

Expressão regular

Uma expressão regular (ER) sobre um alfabeto Σ é definida recursivamente como segue:

1. $\emptyset$, λ, e a para qualquer $a \in \Sigma$ são expressões regulares; elas denotam $\emptyset$, $\{\lambda\}$ e $\{a\}$;
2. se r e s são expressões regulares, então são expressões regulares: $(r + s)$, (rs), e r^*; elas denotam $L(r) \cup L(s)$, $L(r)L(s)$ e $L(r)^*$.

Exemplos de expressões regulares

ERs sobre $\Sigma = \{0, 1\}$ e conjuntos regulares denotados por elas:

ER	*Linguagem denotada*
$\emptyset$	$\emptyset$
λ	$\{\lambda\}$
(01)	$\{0\}\{1\} = \{01\}$
$(0 + 1)$	$\{0\} \cup \{1\} = \{0, 1\}$
$((0 + 1)(01))$	$\{0, 1\}\{01\} = \{001, 101\}$
0^*	$\{0\}^* = \{0^n \mid n \geq 0\}$
$(0 + 1)^*$	$\{0, 1\}^* = \Sigma^*$
$(((0 + 1)^*1)(0 + 1))$	$\{0, 1\}^*\{1\}\{0, 1\}$

Prioridades dos operadores

Regras para omissão de parênteses:

a) Como a união é associativa, pode-se escrever $(r_1 + r_2 + \cdots + r_n)$, omitindo-se os parênteses internos.

b) Idem, para a concatenação.

c) Os parênteses externos podem ser omitidos.

d) Fecho de Kleene tem precedência sobre união e concatenação.

e) Concatenação tem precedência sobre união.

Algumas equivalências

1. $r + s = s + r$
2. $r + \emptyset = r$
3. $r + r = r$
4. $r\lambda = \lambda r = r$
5. $r\emptyset = \emptyset r = \emptyset$
6. $(r + s)t = rt + st$
7. $r(s + t) = rs + rt$
8. $(r + s)^* = (r^*s)^*r^*$
9. $(r + s)^* = r^*(sr^*)^*$
10. $(rs)^* = \lambda + r(sr)^*s$
11. $r^{**} = r^*$
12. $r^* = (rr)^*(\lambda + r)$
13. $\emptyset^* = \lambda$
14. $\lambda^* = \lambda$
15. $r^*r^* = r^*$
16. $rr^* = r^*r$
17. $(r^* + s)^* = (r + s)^*$
18. $(r^*s^*)^* = (r + s)^*$
19. $r^*(r + s)^* = (r + s)^*$
20. $(r + s)^*r^* = (r + s)^*$

Algumas observações sobre a tabela

- Qualquer equivalência que não envolva fecho de Kleene pode ser derivada a partir de 1 a 7 mais as propriedades de associatividade da união e da concatenação.
- Com o fecho de Kleene, não há um conjunto finito de equivalências a partir das quais se possa derivar qualquer outra.
- Algumas equivalências são redundantes. Por exemplo, a 13 pode ser obtida de 2, 5 e 10:

$$
\begin{aligned}
\emptyset^* &= (r\emptyset)^* && \text{por 5} \\
&= \lambda + r(\emptyset r)^*\emptyset && \text{por 10} \\
&= \lambda + \emptyset && \text{por 5} \\
&= \lambda && \text{por 2}
\end{aligned}
$$

Simplificação de expressões regulares

$$
\begin{aligned}
\underline{(00^* + 10^*)}0^*(1^* + 0)^* &= (0 + 1)\underline{0^*0^*}(1^* + 0)^* && \text{por 6} \\
&= (0 + 1)0^*\underline{(1^* + 0)^*} && \text{por 15} \\
&= (0 + 1)0^*\underline{(1 + 0)^*} && \text{por 17} \\
&= (0 + 1)0^*\underline{(0 + 1)^*} && \text{por 1} \\
&= (0 + 1)(0 + 1)^* && \text{por 19}
\end{aligned}
$$

Diga por que:

- $(r + rr + rrr + rrrr)^* = r^*$.
- $((0(0 + 1)1 + 11)0^*(00 + 11))^*(0 + 1)^* = (0 + 1)^*$.
- $r^*(r + s^*) = r^*s^*$.

Notações úteis

- r^+ significa (rr^*).
- r^n, $n \geq 0$ é assim definida, recursivamente:
  a) $r^0 = \lambda$;
  b) $r^n = rr^{n-1}$, para $n \geq 1$.

Exemplos:

$(0 + 1)^{10}$.

$r^* = (r^n)^*(\lambda + r + r^2 + \cdots + r^{n-1})$, para $n > 1$.

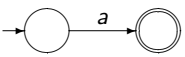
Obtendo AF a partir de expressão regular

Toda expressão regular denota uma linguagem regular.

1. AFs que reconhecem $\emptyset$, $\{\lambda\}$ e $\{a\}$:

AF para $\emptyset$

AF para $\{\lambda\}$



AF para $\{a\}$

2. Dados AFs para L_1 e L_2, é possível construir AFs para $L_1 \cup L_2$, $L_1 L_2$ e L_1^*.

Obtendo expressões regular a partir de AF

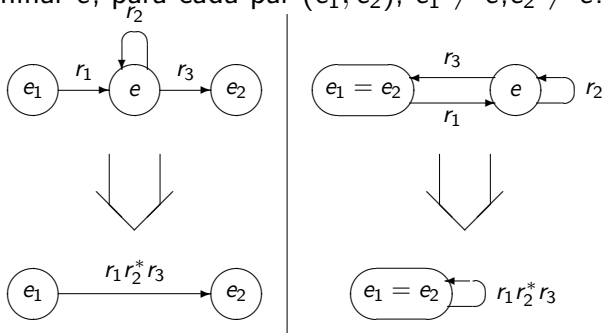
Toda linguagem regular é denotada por alguma expressão regular.

Seja um AFN $M = (E, \Sigma, \delta, I, F)$.

1. Obtenha AFNλ $M' = (E', \Sigma, \delta, i, \{f\})$ equivalente a M tal que:
   - $i \notin \delta(e, a)$ para todo par $(e, a) \in E' \times \Sigma$;
   - $\delta(f, a) = \emptyset$ para todo $a \in \Sigma$.
2. Obtenha diagrama ER inicial a partir de M': substitua transições de e para e' sob s_1, s_2, ..., s_m, por uma só transição de e para e' sob $s_1 + s_2 + \cdots + s_m$.

Obtendo expressões regular a partir de AF

3. Elimine um a um os estados do diagrama ER, exceto i e f.
   - Para eliminar e, para cada par (e_1, e_2), $e_1 \neq e, e_2 \neq e$:



(a) $e_1 \neq e_2$ **(b) $e_1 = e_2$**

   - Se havia transição de e_1 para e_2 sob s substitua-a por transição de e_1 para e_2 sob $s + r_1 r_2^* r_3$.
4. A ER resultante é o rótulo da transição de i para f.

Exemplo

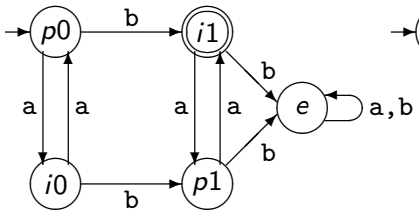


Diagrama de estados

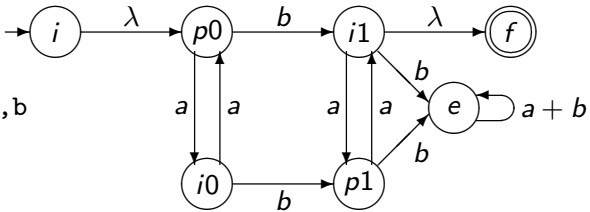
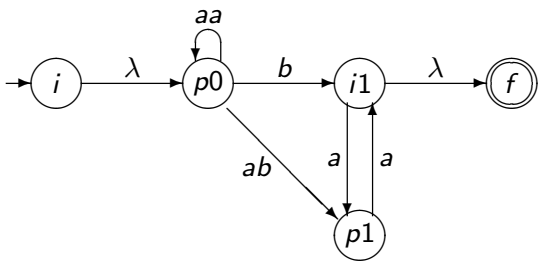


Diagrama ER

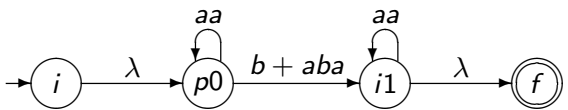
Exemplo (cont.)

1. **Eliminando** e. Como não existe transição de e para algum e_2 diferente de e, ele é simplesmente eliminado.
2. **Eliminando** $i0$:

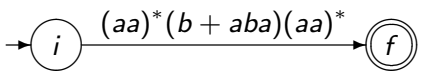


Exemplo (cont.)

3. **Eliminando** $p1$:



4. **Eliminando** $p0$ **e** i_1:



Linguagens Formais e Autômatos

Semana 7: Expressões Regulares

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Linguagens Formais e Autômatos

Semana 8: Gramáticas Regulares

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

O que é uma gramática regular

Gramática regular

Uma gramática regular (GR) é uma gramática (V, Σ, R, P), em que cada regra tem uma das formas:

- $X \to a$;
- $X \to aY$;
- $X \to \lambda$;

$X, Y \in V$ e $a \in \Sigma$.

$\Rightarrow$ Formato das formas sentenciais: wA, $w \in \Sigma^+$, $A \in V$.

Exemplo

$L = \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid w \text{ não contém } \mathtt{abc}\}$.

Uma GR que gera L: $(\{A, B, C\}, \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}, R, A)$, em que R contém:

$$A \rightarrow \mathtt{a}B \mid \mathtt{b}A \mid \mathtt{c}A \mid \lambda$$
$$B \rightarrow \mathtt{a}B \mid \mathtt{b}C \mid \mathtt{c}A \mid \lambda$$
$$C \rightarrow \mathtt{a}B \mid \mathtt{b}A \mid \lambda$$

AF a partir de GR

Toda gramática regular gera uma linguagem regular.

Seja $G = (V, \Sigma, R, P)$ e $Z \notin V$.
Um AFN que reconhece L(G): $M = (E, \Sigma, \delta, \{P\}, F)$, em que

- $E = \begin{cases} V \cup \{Z\} & \text{se } R \text{ contém regra da forma } X \to a \\ V & \text{caso contrário.} \end{cases}$
- Para toda regra da forma:
  - $X \to aY$ faça $Y \in \delta(X, a)$,
  - $X \to a$ faça $Z \in \delta(X, a)$.
- $F = \begin{cases} \{X | X \to \lambda \in R\} \cup \{Z\} & \text{se } Z \in E \\ \{X | X \to \lambda \in R\} & \text{caso contrário.} \end{cases}$

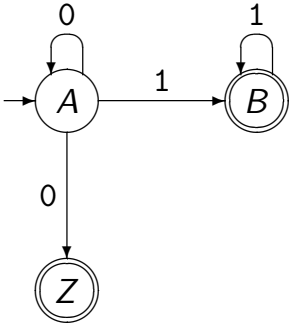
Exemplo

G:

$A \to 0A|1B|0$

$B \to 1B|\lambda$

$L(G) = 0^*(0 + 1^+)$. AFN para $L(G)$:



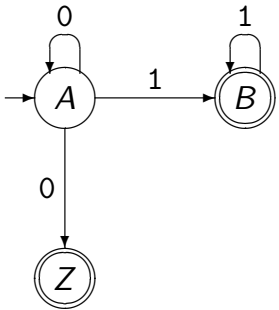
Gramáticas Regulares

Toda linguagem regular é gerada por gramática regular.

Seja um AFN $M = (E, \Sigma, \delta, \{i\}, F)$.
Uma GR que gera $L(M)$: $G = (E, \Sigma, R, i)$, em que:

$$R = \{e \to ae' \mid e' \in \delta(e, a)\} \cup \{e \to \lambda \mid e \in F\}.$$

Exemplo



GR:

$A \rightarrow 0A|0Z|1B$

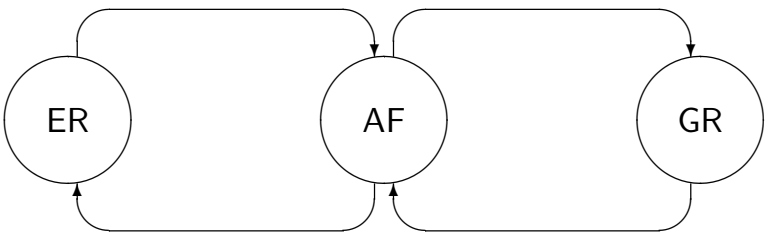
$B \rightarrow 1B|\lambda$

$Z \rightarrow \lambda$

Uma síntese

$\Longrightarrow$ ERs, GRs e AFs são formalismos alternativos para linguagens regulares.

Transformações entre formalismos:



Linguagens Formais e Autômatos

Semana 9: Máquinas de Mealy e de Moore

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Associando saída aos estados

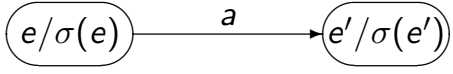
Máquina de Moore

Uma máquina de Moore é uma sêxtupla $(E, \Sigma, \Delta, \delta, \sigma, i)$, em que:

- E (o conjunto de estados), Σ (o alfabeto de entrada), δ (a função de transição) e i (o estado inicial) são como em AFDs;
- Δ é o alfabeto de saída; e
- $\sigma : E \to \Delta$ é a função de saída, uma função total.

Diagrama de estados de uma máquina de Moore

Em um diagrma de estados, a transição $\delta(e, a) = e'$ é representada assim, juntamente com $\sigma(e)$ e $\sigma(e')$:



A saída computada por uma máquina de Moore

Seja uma máquina de Moore $M = (E, \Sigma, \Delta, \delta, \sigma, i)$. A **função de saída estendida** para M, $r : E \times \Sigma^* \to \Delta^*$ é definida recursivamente como segue:

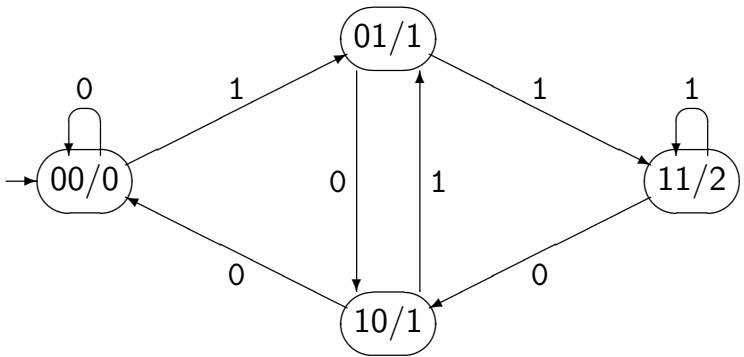
a) $r(e, \lambda) = \sigma(e)$;

b) $r(e, ay) = \sigma(e)r(\delta(e, a), y)$, para todo $a \in \Sigma$ e $y \in \Sigma^*$.

A **saída computada** por uma máquina de Moore $M = (E, \Sigma, \Delta, \delta, \sigma, i)$ para a palavra $w \in \Sigma^*$ é $r(i, w)$.

Exemplo de máquina de Moore

Último símbolo de $r(00, w)$: número de 1s nos dois últimos símbolos de w.



Associando saída às transições

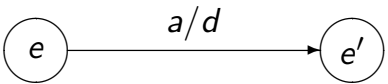
Máquina de Mealy

Uma máquina de Mealy é uma sêxtupla $(E, \Sigma, \Delta, \delta, \sigma, i)$, em que:

- E *(o conjunto de estados)*, Σ *(o alfabeto de entrada)*, δ *(a função de transição)* e i *(o estado inicial)* são como em AFDs;
- Δ é o alfabeto de saída;
- $\sigma : E \times \Sigma \to \Delta$ é a função de saída, uma função total.

Diagrama de estados de uma máquina de Mealy

Uma transição $\delta(e, a) = e'$ com a saída $\sigma(e, a) = d$ é representada em um diagrama de estados assim:



A saída computada por uma máquina de Mealy

Seja uma máquina de Mealy $M = (E, \Sigma, \Delta, \delta, \sigma, i)$. A **função de saída estendida** para M, $s : E \times \Sigma^* \to \Delta^*$, é definida recursivamente como segue:

a) $s(e, \lambda) = \lambda$;

b) $s(e, ay) = \sigma(e, a)s(\delta(e, a), y)$, para todo $a \in \Sigma$ e $y \in \Sigma^*$.

A **saída computada** por uma máquina de Mealy $M = (E, \Sigma, \Delta, \delta, \sigma, i)$ para a palavra $w \in \Sigma^*$ é $s(i, w)$.

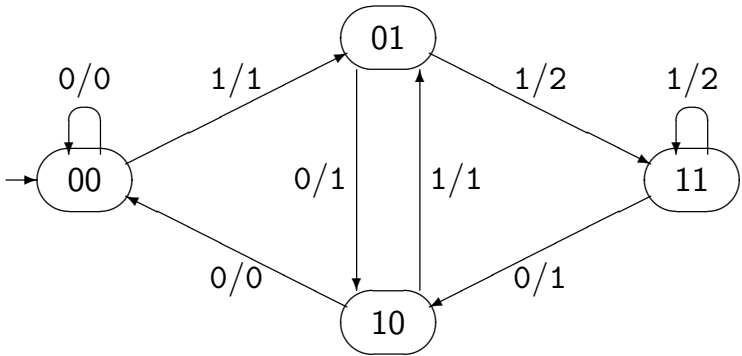
Equivalência de máquinas de Moore e de Mealy

Máquinas equivalentes

Uma máquina de Moore $(E_1, \Sigma, \Delta, \delta_1, \sigma_1, i_1)$ e uma máquina de Mealy $(E_2, \Sigma, \Delta, \delta_2, \sigma_2, i_2)$ são ditas equivalentes se, para todo $w \in \Sigma^*$, $r(i_1, w) = \sigma_1(i_1)s(i_2, w)$.

Um exemplo

Máquina de Moore Moore $\Rightarrow$ Máquina de Mealy:



Linguagens Formais e Autômatos

Semana 9: Máquinas de Mealy e de Moore

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Linguagens Formais e Autômatos

Semana 10: Revisão e exercícios

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

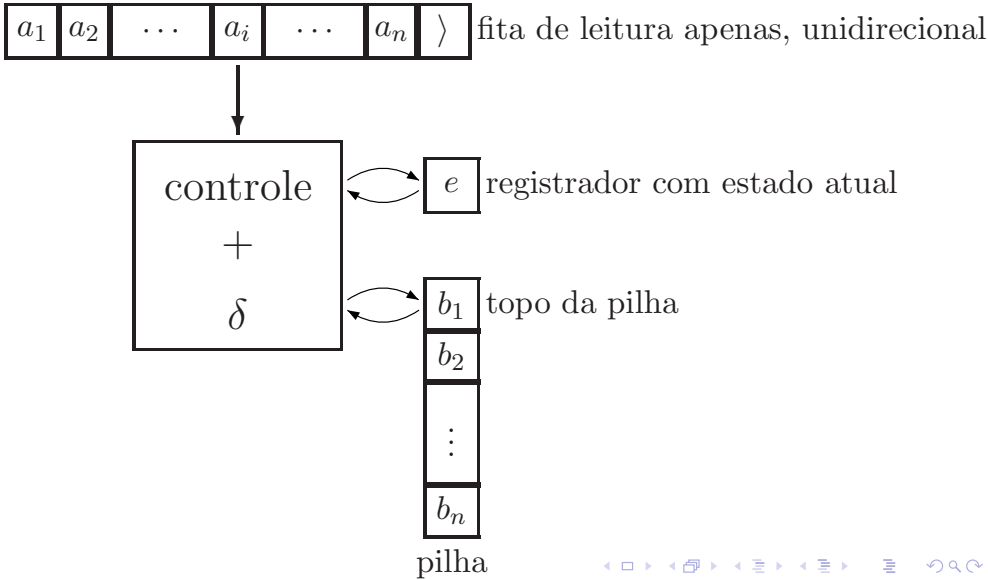
Linguagens Formais e Autômatos

Semana 11: Autômatos de Pilha

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Arquitetuta de um autômato de pilha determinístico



Configuração inicial

No início:

- o registrador contém o estado inicial
- a palavra de entrada está na fita a partir da primeira célula
- o cabeçote da fita está posicionado na primeira célula
- a pilha está vazia

Transição de um AP

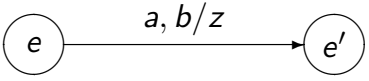
Sejam:

- E: conjunto (finito) de estados
- Σ: alfabeto de entrada
- Γ: alfabeto de pilha

Cada transição do AP é da forma

$$\delta(e, a, b) = [e', z]$$

$e, e' \in E$, $a \in \Sigma_\lambda$, $b \in \Gamma_\lambda$ e $z \in \Gamma^*$.
Em um diagrama de estados:

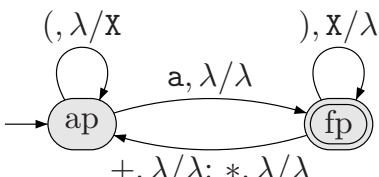


Um exemplo de AP

Conjunto EA das expressões aritméticas:

a) a $\in$ EA;

b) se $x, y \in$ EA, então $(x) \in$ EA, $x{+}y \in$ EA e $x{*}y \in$ EA.



$\Sigma = \{\mathtt{a}, (,), *, +\}$

$\Gamma = \{\mathtt{X}\}$

Computação de um AP

- Pilha: uma palavra de Γ^*. Pilha vazia: λ.
- Configuração instantânea: $[e, y, p]$, sendo p a pilha.

Computação do AP quando a palavra de entrada é `(a*(a+a))`:

$$
\begin{aligned}
[ap, (\mathtt{a} * (\mathtt{a} + \mathtt{a})), \lambda] &\vdash [ap, \mathtt{a} * (\mathtt{a} + \mathtt{a})), X] \\
&\vdash [fp, *(\mathtt{a} + \mathtt{a})), X] \\
&\vdash [ap, (\mathtt{a} + \mathtt{a})), X] \\
&\vdash [ap, \mathtt{a} + \mathtt{a})), XX] \\
&\vdash [fp, +\mathtt{a})), XX] \\
&\vdash [ap, \mathtt{a})), XX] \\
&\vdash [fp,)), XX] \\
&\vdash [fp,), X] \\
&\vdash [fp, \lambda, \lambda]
\end{aligned}
$$

Outros exemplos/condições para reconhecimento

$$[ap, \mathtt{a}), \lambda] \vdash [fp,), \lambda]$$

$\Longrightarrow$ o AP para sem consumir toda a palavra de entrada.

$$\begin{aligned}[ap, (\mathtt{a}, \lambda] &\vdash [ap, \mathtt{a}, \mathtt{X}] \\ &\vdash [fp, \lambda, \mathtt{X}]\end{aligned}$$

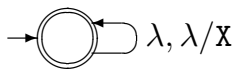
$\Longrightarrow$ o AP para em estado final com pilha não vazia.

Para uma palavra ser reconhecida:

- ela deve ser totalmente consumida;
- a máquina deve terminar em um estado final;
- a pilha deve estar vazia.

Um exemplo estranho

Seja o AP com $\Sigma = \{1\}$ e com o diagrama de estados:



Para toda palavra em $\{1\}^+$, o AP não para. Em particular:

$$[0, 1, \lambda] \vdash [0, 1, X] \vdash [0, 1, XX] \ldots$$

Perguntas:

- para λ, o AP para ou não?
- λ é reconhecida ou não?

Transições compatíveis

Definição

Seja $\delta : E \times \Sigma_\lambda \times \Gamma_\lambda \to E \times \Gamma^*$.

As transições $\delta(e_1, a_1, b_1) = [e_1', z_1]$ *e* $\delta(e_2, a_2, b_2) = [e_2', z_2]$ *são compatíveis sse* $e_1 = e_2$ *e*

$$(a_1 = a_2 \text{ ou } a_1 = \lambda \text{ ou } a_2 = \lambda) \text{ e } (b_1 = b_2 \text{ ou } b_1 = \lambda \text{ ou } b_2 = \lambda)$$

Ou ainda: $\delta(e_1, a_1, b_1) = [e_1', z_1]$ e $\delta(e_2, a_2, b_2) = [e_2', z_2]$ são incompatíveis sse $e_1 \neq e_2$ ou

$$(a_1 \neq a_2 \text{ e } a_1 \neq \lambda \text{ e } a_2 \neq \lambda) \text{ ou } (b_1 \neq b_2 \text{ e } b_1 \neq \lambda \text{ e } b_2 \neq \lambda)$$

O que é AP determinístico

Definição

Um autômato de pilha determinístico (APD) é uma sêxtupla $(E, \Sigma, \Gamma, \delta, i, F)$*, em que*

- E *é um conjunto finito de um ou mais estados*
- Σ *é o alfabeto de entrada*
- Γ *é o alfabeto de pilha*
- δ *é uma função parcial de* $E \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\})$ *para* $E \times \Gamma^*$*, sem transições compatíveis*
- $i \subseteq E$ *é o estado inicial*
- $F \subseteq E$ *é o conjunto de estados finais*

A linguagem reconhecida por um APD

Computação

$[e, ay, bz] \vdash [e', y, xz] \leftrightarrow \delta(e, a, b) = [e', x]$

$\overset{*}{\vdash}$ é o fecho reflexivo e transitivo de $\vdash$.

Definição

Seja $M = (E, \Sigma, \Gamma, \delta, i, F)$. *A linguagem reconhecida por* M *é:*

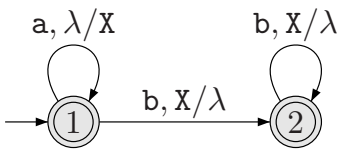
$$L(M) = \{w \in \Sigma^* \mid [i, w, \lambda] \overset{*}{\vdash} [e, \lambda, \lambda] \text{ para algum } e \in F\}.$$

Exemplo

$\{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbf{N}\}$ é reconhecida por $(\{1,2\},\{\mathtt{a},\mathtt{b}\},\{\mathtt{X}\},\delta,1,\{1,2\})$, em que δ é dada por:

1. $\delta(1,\mathtt{a},\lambda) = [1,\mathtt{X}]$;
2. $\delta(1,\mathtt{b},\mathtt{X}) = [2,\lambda]$;
3. $\delta(2,\mathtt{b},\mathtt{X}) = [2,\lambda]$.

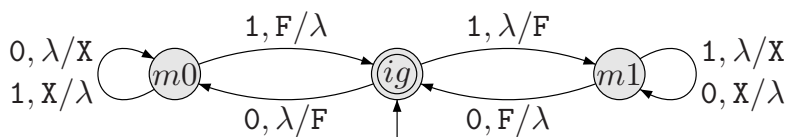
Diagrama de estados:



Um outro exemplo/versão 1

Um APD para $\{w \in \{0,1\}^* \mid n_0(w) = n_1(w)\}$

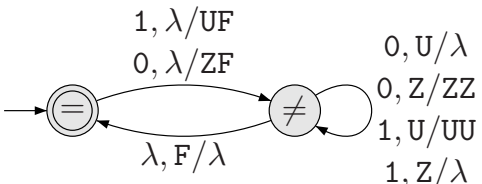
- estados indicam que símbolo ocorre mais:
  - $m0$: existem mais 0s do que 1s
  - $m1$: existem mais 1s do que 0s



Outro exemplo/versão 2

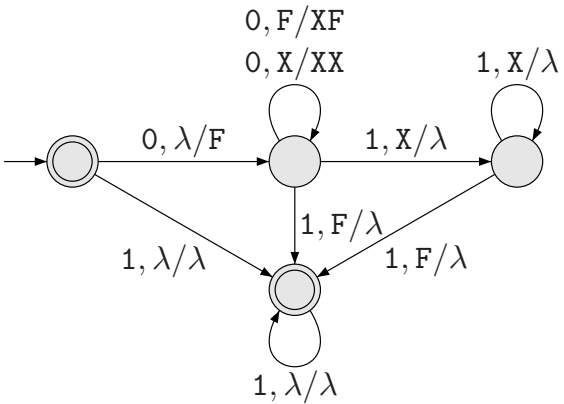
Um APD para $\{w \in \{0,1\}^* \mid n_0(w) = n_1(w)\}$

- pilha indica que símbolo ocorre mais:
  - Z: existem mais 0s do que 1s
  - U: existem mais 1s do que 0s



Mais um APD com marcação de fundo de pilha

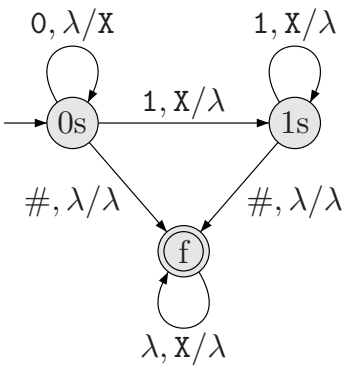
Um APD para $\{0^m 1^n \mid m \leq n\}$



APD com símbolo de final de palavra

APD que reconhece $\{0^m1^n\# \mid m \geq n\}$:

- $\{0^m1^n \mid m \geq n\}$ não é reconhecível por APD!



O que é AP não determinístico

Definição

Um autômato de pilha não determinístico (APN) é uma sêxtupla $(E, \Sigma, \Gamma, \delta, I, F)$*, em que*

- E, Σ, Γ *e* F *são como em APDs*
- δ*, a função de transição, é uma função parcial de* $E \times \Sigma_\lambda \times \Gamma_\lambda$ *para* D*, sendo* D *constituído dos subconjuntos finitos de* $E \times \Gamma^*$
- I*, um subconjunto de* E*, é o conjunto de estados iniciais*

A linguagem reconhecida por um APN

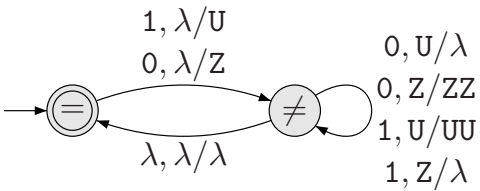
Definição

Seja um APN $M = (E, \Sigma, \Gamma, \delta, I, F)$. *A linguagem reconhecida por* M *é:*

$$L(M) = \{w \in \Sigma^* \mid [i, w, \lambda] \overset{*}{\vdash} [e, \lambda, \lambda] \text{ para } i \in I \text{ e } e \in F\}$$

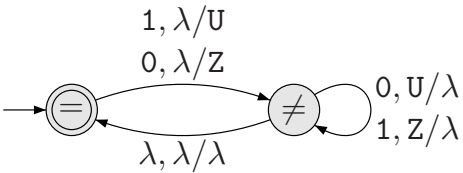
Exemplo de APN

APN que reconhece $\{w \in \{0,1\}^* \mid n_0(w) = n_1(w)\}$:

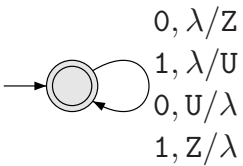


Exemplo de APN

Mais APNs que reconhecem $\{w \in \{0,1\}^* \mid n_0(w) = n_1(w)\}$:



(a) O segundo



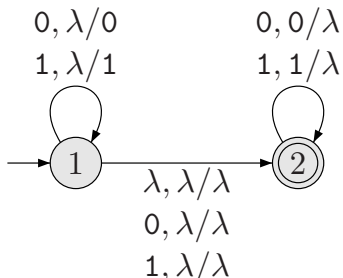
(b) O terceiro

Exemplo não tratável por APD

APN que reconhece a linguagem $\{w \in \{0,1\}^* \mid w = w^R\}$

Em uma computação de sucesso para w:

- se $|w|$ for par, será percorrida a transição de 1 para 2 sob λ
- se $|w|$ for ímpar e o símbolo do meio for a, será percorrida a transição de 1 para 2 sob a



Reconhecimento por estado final

Definição

Seja um APN $M = (E, \Sigma, \Gamma, \delta, I, F)$. *A linguagem reconhecida por* M *por estado final é:*

$$L_F(M) = \{w \in \Sigma^* \mid [i, w, \lambda] \overset{*}{\vdash} [e, \lambda, y] \text{ para } i \in I, e \in F \text{ e } y \in \Gamma^*\}.$$

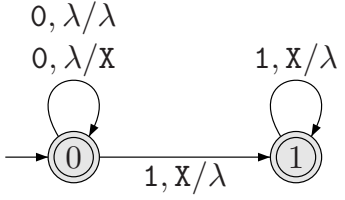
Reconhecimento por estado final

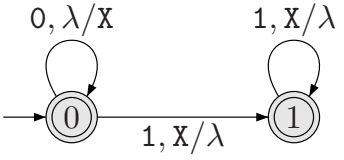
Definição

Seja um APN $M = (E, \Sigma, \Gamma, \delta, I, F)$*. A linguagem reconhecida por* M *por estado final é:*

$$L_F(M) = \{w \in \Sigma^* \mid [i, w, \lambda] \overset{*}{\vdash} [e, \lambda, y] \text{ para } i \in I, e \in F \text{ e } y \in \Gamma^*\}.$$

Exemplo: APNs para $L = \{0^m 1^n \mid m \geq n\}$:





Reconhecimento por pilha vazia

Definição

Seja um APN $M = (E, \Sigma, \Gamma, \delta, I)$. *A linguagem reconhecida por M por pilha vazia é:*

$$L_V(M) = \{w \in \Sigma^* \mid [i, w, \lambda] \overset{*}{\vdash} [e, \lambda, \lambda] \text{ para algum } i \in I \text{ e } e \in E\}.$$

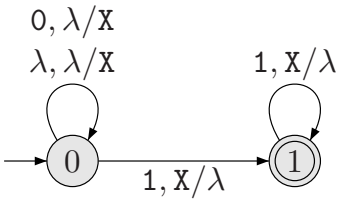
Reconhecimento por pilha vazia

Definição

Seja um APN $M = (E, \Sigma, \Gamma, \delta, I)$. *A linguagem reconhecida por M por pilha vazia é:*

$$L_V(M) = \{w \in \Sigma^* \mid [i, w, \lambda] \overset{*}{\vdash} [e, \lambda, \lambda] \text{ para algum } i \in I \text{ e } e \in E\}.$$

Exemplo: reconhecimento de $\{0^m 1^n \mid m \leq n\}$ por pilha vazia:



Equivalência de métodos de reconhecimento

Teorema

Seja L uma linguagem. As seguinte afirmativas são equivalentes:

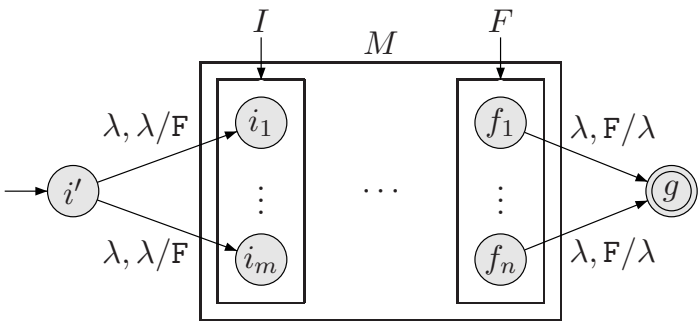
a) *L pode ser reconhecida por pilha vazia e estado final*

b) *L pode ser reconhecida por estado final*

c) *$L \cup \{\lambda\}$ pode ser reconhecida por pilha vazia*

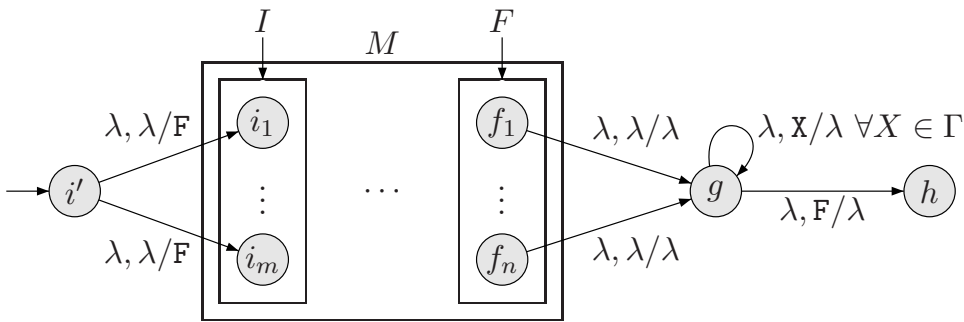
Equivalência de métodos de reconhecimento

(a) pilha vazia e estado final $\rightarrow$ (b) estado final:



Equivalência de métodos de reconhecimento

(b) estado final $\rightarrow$ (c) pilha vazia:



Equivalência de métodos de reconhecimento

Teorema

Seja L uma linguagem. As seguinte afirmativas são equivalentes:

a) *L pode ser reconhecida por pilha vazia e estado final*

b) *L pode ser reconhecida por estado final*

c) *$L \cup \{\lambda\}$ pode ser reconhecida por pilha vazia*

(c) pilha vazia $\rightarrow$ (a) pilha vazia e estado final

Se $M = (E, \Sigma, \Gamma, \delta, I)$, então $M' = (E, \Sigma, \Gamma, \delta, I, E)$.

Linguagens Formais e Autômatos

Semana 11: Autômatos de Pilha

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Linguagens Formais e Autômatos

Semana 12: Gramáticas Livre do Contexto - parte 1

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

O que é gramática livre do contexto

Definição

Uma gramática livre do contexto (GLC) é uma gramática (V, Σ, R, P)*, em que cada regra tem a forma* $X \to w$*, em que* $X \in V$ *e* $w \in (V \cup \Sigma)^*$.

O que é gramática livre do contexto

Definição

Uma gramática livre do contexto (GLC) é uma gramática (V, Σ, R, P)*, em que cada regra tem a forma* $X \to w$*, em que* $X \in V$ *e* $w \in (V \cup \Sigma)^*$*.*

Exemplo

$\{0^n 1^n \mid n \in \mathbf{N}\}$ *é gerada por* $G = (\{P\}, \{0, 1\}, R, P)$*, em que* R *consta de:*

$$P \to 0P1 \mid \lambda$$

Outro exemplo de GLC

Exemplo

GLC que gera $\{w \in \{0,1\}^* \mid w = w^R\}$*:*

$G = (\{P\}, \{0,1\}, R, P)$*, tendo* R *as 5 regras:*

$$P \rightarrow 0P0 \mid 1P1 \mid 0 \mid 1 \mid \lambda$$

Outro exemplo de GLC

Exemplo

GLC que gera $\{w \in \{0,1\}^* \mid w$ *tem um número igual de* 0*s e* 1*s*$\}$*:*

$(\{P\}, \{0,1\}, R, P)$*, tendo* R *as 3 regras:*

$$P \rightarrow 0P1P \mid 1P0P \mid \lambda$$

GLC para expressões aritméticas

Exemplo

$(\{E, T, F\}, \{\mathtt{a}, +, *, (,)\}, R, E)$, *em que* R *consta de:*

$$
\begin{aligned}
E &\rightarrow E{+}T \mid T \\
T &\rightarrow T{*}F \mid F \\
F &\rightarrow (E) \mid \mathtt{a}
\end{aligned}
$$

O que é linguagem livre do contexto

Definição

Uma linguagem é dita ser uma linguagem livre do contexto sse existe uma gramática livre do contexto que a gera.

O que é linguagem livre do contexto

Definição

Uma linguagem é dita ser uma linguagem livre do contexto sse existe uma gramática livre do contexto que a gera.

$\Rightarrow$ Como capturar a essência de uma derivação, o que não depende da ordem de aplicação das regras?

O conceito de árvore de derivação

Definição

Seja uma GLC $G = (V, \Sigma, R, P)$. Uma árvore de derivação (AD) de uma forma sentencial de G é uma árvore ordenada construída recursivamente como segue:

a) *uma árvore com apenas um vértice de rótulo P é uma AD;*

b) *se $X \in V$ é rótulo de uma folha f de uma AD A, então:*

- *i. se $X \to \lambda \in R$, então a árvore obtida acrescentando-se a A mais um vértice v com rótulo λ e uma aresta $\{f, v\}$ é uma AD;*
- *ii. se $X \to x_1 x_2 \ldots x_n \in R$, então a árvore obtida acrescentando-se a A mais n vértices $v_1, v_2, \ldots, v_n$ com rótulos $x_1, x_2, \ldots, x_n$, nessa ordem, e n arestas $\{f, v_1\}, \{f, v_2\}, \ldots, \{f, v_n\}$, é uma AD.*

Exemplo de construção de uma AD para a*(a+a)

$$E \rightarrow E+T \mid T$$
$$T \rightarrow T*F \mid F$$
$$F \rightarrow (E) \mid \mathtt{a}$$

A derivação

$$E \Rightarrow T \qquad \text{(regra } E \rightarrow T\text{)}$$

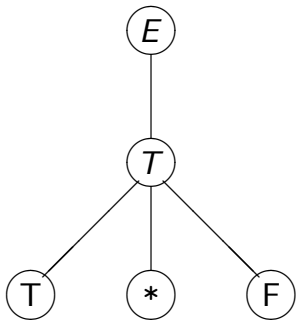
leva à AD:

Exemplo de construção de uma AD (continuação)

A derivação evolue para:

$$\begin{aligned} E &\Rightarrow T && \text{(regra } E \to T\text{)} \\ &\Rightarrow T*F && \text{(regra } T \to T*F\text{)} \end{aligned}$$

e a AD correspondente para:



Exemplo de construção de uma AD (continuação)

Tem-se duas opções para continuar a derivação:

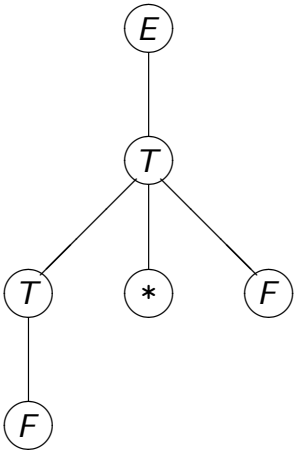
$$\begin{aligned} E &\Rightarrow T && (\text{regra } E \to T) \\ &\Rightarrow T*F && (\text{regra } T \to T*F) \\ &\Rightarrow F*F && (\text{regra } T \to F) \end{aligned}$$

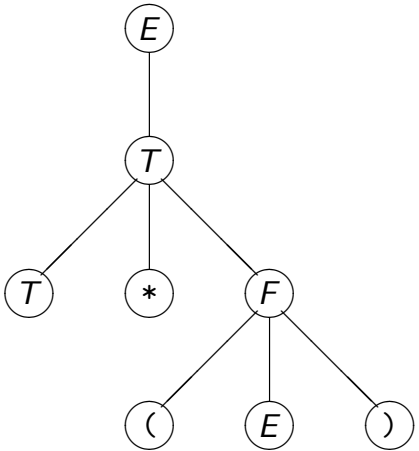
ou então:

$$\begin{aligned} E &\Rightarrow T && (\text{regra } E \to T) \\ &\Rightarrow T*F && (\text{regra } T \to T*F) \\ &\Rightarrow T*(E) && (\text{regra } F \to (E)). \end{aligned}$$

Exemplo de construção de uma AD (continuação)

As duas opções:





Exemplo de construção de uma AD/conclusão

$\Rightarrow$ Após uma derivação de 11 passos tem-se uma AD para $\mathtt{a} * (\mathtt{a} + \mathtt{a})$

Observações:

- Número de passos da derivação: número de vértices internos da AD.
- A estrutura da AD é normalmente utilizada para associar significado:

  Mais de uma AD para $w \Rightarrow$ mais de um significado para w.

Ambiguidade

Definição

Uma GLC é uma gramática ambígua quando existe mais de uma AD para alguma sentença que ela gera.

Ambiguidade

Definição

Uma GLC é uma gramática ambígua quando existe mais de uma AD para alguma sentença que ela gera.

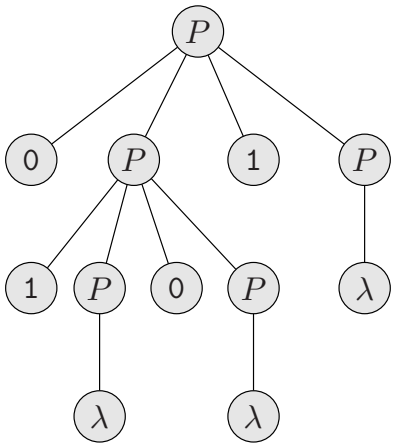
Exemplo

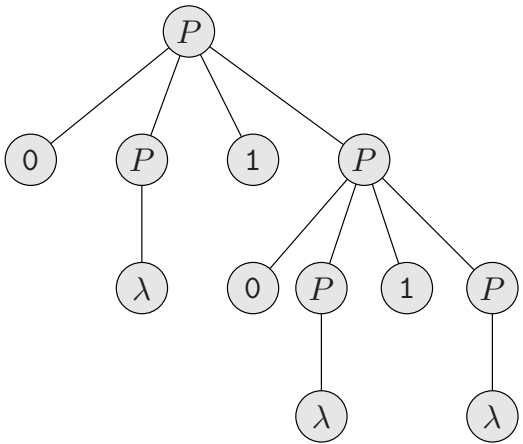
Uma GLC ambígua:

$$P \rightarrow 0P1P \mid 1P0P \mid \lambda$$

Duas árvores de derivação para 0101

$P \rightarrow 0P1P \mid 1P0P \mid \lambda$





Outra gramática ambígua

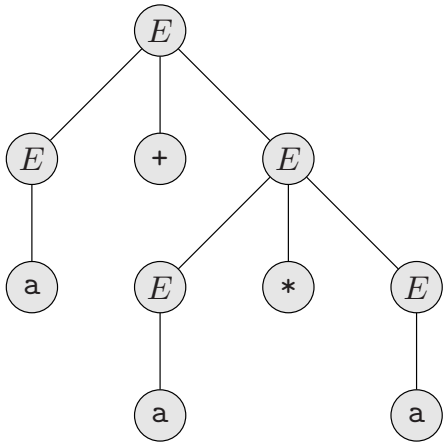
Exemplo

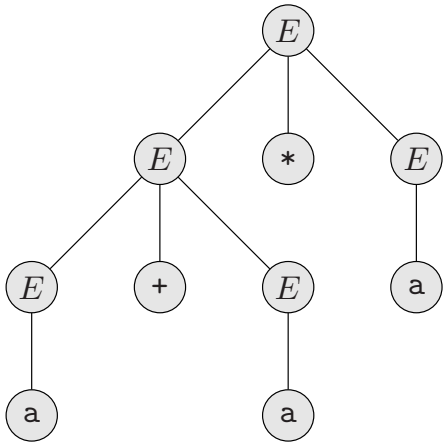
Uma GLC ambígua para expressões aritméticas:

$$E \rightarrow E + E \mid E * E \mid (E) \mid \mathtt{a}$$

Duas ADs para a+a*a

$E \rightarrow E + E \mid E * E \mid (E) \mid \mathtt{a}$





Derivações mais à esquerda e mais à direita

Definição

Uma derivação é dita mais à esquerda (DME) se em cada passo é expandida a variável mais à esquerda.

Derivações mais à esquerda e mais à direita

Definição

Uma derivação é dita mais à esquerda (DME) se em cada passo é expandida a variável mais à esquerda.

Definição

Uma derivação é dita mais à direita (DMD) se em cada passo é expandida a variável mais à direita.

Derivações mais à esquerda e mais à direita

Definição

Uma derivação é dita mais à esquerda (DME) se em cada passo é expandida a variável mais à esquerda.

Definição

Uma derivação é dita mais à direita (DMD) se em cada passo é expandida a variável mais à direita.

$\Rightarrow$ Existe uma única DME e uma única DMD correspondentes a uma AD e vice-versa.

Ambiguidade e DME e DMD

Como existe uma única DME e uma única DMD correspondentes a uma AD e vice-versa:

- uma GLC é ambígua sse existe mais de uma DME para alguma sentença que ela gere;
- uma GLC é ambígua sse existe mais de uma DMD para alguma sentença que ela gere.

Linguagens inerentemente ambíguas

Definição

Linguagem inerentemente ambígua: LLC para a qual toda GLC é ambígua.

Linguagens inerentemente ambíguas

Definição

Linguagem inerentemente ambígua: LLC para a qual toda GLC é ambígua.

Exemplo

$\{a^m b^n c^k \mid m = n \text{ ou } n = k\}$ *é inerentemente ambígua.*

Linguagens inerentemente ambíguas

Definição

Linguagem inerentemente ambígua: LLC para a qual toda GLC é ambígua.

Exemplo

$\{a^m b^n c^k \mid m = n \text{ ou } n = k\}$ *é inerentemente ambígua.*

- A detecção e remoção de ambiguidade em GLCs é muito importante.
- O problema de determinar se uma GLC é ambígua é indecidível.

Analisadores sintáticos obtidos de GLCs

Dois tipos de analisadores sintáticos:

- Bottom-up, à medida que lê a palavra:
  - constrói a AD da fronteira para a raiz (aplica as regras de forma invertida)
  - a derivação respectiva é uma DMD (obtida de trás para a frente)
- Top-down, à medida que lê a palavra:
  - constrói a AD da raiz em direção à fronteira
  - a derivação respectiva é uma DME

(detalhes em qualquer livro sobre construção de compiladores).

A necessidade de manipulação de GLCs

- Algumas gramáticas podem ser mais adequadas que outras, dependendo do contexto para o qual elas foram projetadas.
- Existem algumas formas normais que são apropriadas em diversas situações.
- A seguir, técnicas de manipulação de GLCs, assim como duas formas normais importantes.

Linguagens Formais e Autômatos

Semana 12: Gramáticas Livre do Contexto - parte 1

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Linguagens Formais e Autômatos

Semana 13: Gramáticas Livres do Contexto - parte 2

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Variáveis inúteis

Definição

Seja uma GLC $G = (V, \Sigma, R, P)$*. Uma variável* $X \in V$ *é dita ser uma variável útil sse existem* $u, v \in (V \cup \Sigma)^*$ *e* $w \in \Sigma^*$ *tais que:*

$$P \stackrel{*}{\Rightarrow} uXv \stackrel{*}{\Rightarrow} w.$$

Variáveis inúteis

Definição

Seja uma GLC $G = (V, \Sigma, R, P)$. *Uma variável* $X \in V$ *é dita ser uma variável útil sse existem* $u, v \in (V \cup \Sigma)^*$ *e* $w \in \Sigma^*$ *tais que:*

$$P \overset{*}{\Rightarrow} uXv \overset{*}{\Rightarrow} w.$$

Exemplo

$P \rightarrow AB \mid \mathtt{a}$

$B \rightarrow \mathtt{b}$

$C \rightarrow \mathtt{c}$

Que variáveis dessa GLC são inúteis?

Exemplos de variáveis inúteis

$P \to AB \mid \mathtt{a}$

$B \to \mathtt{b}$

$C \to \mathtt{c}$

- C é inútil: não existem u e v tais que $P \stackrel{*}{\Rightarrow} uCv$;
- A é inútil: não existe $w \in \Sigma^*$ tal que $A \stackrel{*}{\Rightarrow} w$;
- B é inútil: $P \stackrel{*}{\Rightarrow} uBv$ apenas para $u = A$ e $v = \lambda$, e não existe $w \in \Sigma^*$ tal que $AB \stackrel{*}{\Rightarrow} w$.

Exemplos de variáveis inúteis

$P \rightarrow AB \mid \mathtt{a}$

$B \rightarrow \mathtt{b}$

$C \rightarrow \mathtt{c}$

- C é inútil: não existem u e v tais que $P \stackrel{*}{\Rightarrow} uCv$;
- A é inútil: não existe $w \in \Sigma^*$ tal que $A \stackrel{*}{\Rightarrow} w$;
- B é inútil: $P \stackrel{*}{\Rightarrow} uBv$ apenas para $u = A$ e $v = \lambda$, e não existe $w \in \Sigma^*$ tal que $AB \stackrel{*}{\Rightarrow} w$.

GLC equivalente sem símbolos inúteis:

$P \rightarrow \mathtt{a}$.

Eliminação de variáveis inúteis

Seja $G = (V, \Sigma, R, P)$ tal que $L(G) \neq \emptyset$.
Construção de uma GLC G'' equivalente a G, sem variáveis inúteis:

a) Obtenha $G' = (V', \Sigma, R', P)$, em que:

- $V' = \{X \in V \mid X \overset{*}{\Rightarrow}_G w$ para algum $w \in \Sigma^*\}$, e
- $R' = \{r \in R \mid r$ não contém símbolo de $V - V'\}$.

b) Obtenha $G'' = (V'', \Sigma, R'', P)$, em que:

- $V'' = \{X \in V' \mid P \overset{*}{\Rightarrow}_{G'} uXv$ para algum $u, v \in (V' \cup \Sigma)^*\}$, e
- $R'' = \{r \in R' \mid r$ não contém símbolo de $V' - V''\}$.

Determinando variáveis que produzem sentenças

Algoritmo que determina $\{X \in V \mid X \stackrel{*}{\Rightarrow} w \text{ para algum } w \in \Sigma^*\}$:

Entrada: uma GLC $G = (V, \Sigma, R, P)$.
Saída: $\mathcal{I}_1 = \{X \in V \mid X \stackrel{*}{\Rightarrow} w \text{ para algum } w \in \Sigma^*\}$.
$\mathcal{I}_1 \leftarrow \emptyset$;
repita
    $\mathcal{N} \leftarrow \{X \notin \mathcal{I}_1 \mid X \rightarrow z \in R \text{ e } z \in (\mathcal{I}_1 \cup \Sigma)^*\}$;
    $\mathcal{I}_1 \leftarrow \mathcal{I}_1 \cup \mathcal{N}$
até $\mathcal{N} = \emptyset$;
retorne $\mathcal{I}_1$.

Determinando variáveis alcançáveis a partir de P

Algoritmo que determina
$\{X \in V \mid P \stackrel{*}{\Rightarrow} uXv$ para algum $u, v \in (V \cup \Sigma)^*\}$:

Entrada: uma GLC $G = (V, \Sigma, R, P)$.
Saída: $\mathcal{I}_2 = \{X \in V \mid P \stackrel{*}{\Rightarrow} uXv$ para algum $u, v \in (V \cup \Sigma)^*\}$.
$\mathcal{I}_2 \leftarrow \emptyset$; $\mathcal{N} \leftarrow \{P\}$;
repita
    $\mathcal{I}_2 \leftarrow \mathcal{I}_2 \cup \mathcal{N}$;
    $\mathcal{N} \leftarrow \{Y \notin \mathcal{I}_2 \mid X \rightarrow uYv$ para algum $X \in \mathcal{N}$ e $u, v \in (V \cup \Sigma)^*\}$
até $\mathcal{N} = \emptyset$;
retorne $\mathcal{I}_2$.

Exemplo/eliminação de variáveis inúteis

$$
\begin{aligned}
A &\to ABC \mid AEF \mid BD \\
B &\to B0 \mid 0 \\
C &\to 0C \mid EB \\
D &\to 1D \mid 1 \\
E &\to BE \\
F &\to 1F1 \mid 1
\end{aligned}
$$

$$
\begin{aligned}
V' = \{B, D, F, A\} \Rightarrow \quad A &\to BD \\
B &\to B0 \mid 0 \\
D &\to 1D \mid 1 \\
F &\to 1F1 \mid 1
\end{aligned}
$$

$$
\begin{aligned}
V'' = \{A, B, D\} \Rightarrow \quad A &\to BD \\
B &\to B0 \mid 0 \\
D &\to 1D \mid 1
\end{aligned}
$$

Eliminação de uma regra

$G = (V, \Sigma, R, P)$, $X \to w \in R$ em que $\underline{X \neq P}$

Para eliminar $X \to w$, colocar em R':

1. cada $Y \to z \in R - \{X \to w\}$ tal que $X \notin \mathit{vars}(z)$; e
2. cada $Y \to x_0\gamma_1x_1\gamma_2x_2 \ldots \gamma_nx_n$, em que cada γ_j pode ser X ou w, para cada $Y \to x_0Xx_1Xx_2 \ldots Xx_n \in R$ com $n > 0$ ocorrências de X e $X \notin \mathit{vars}(x_i)$ para $0 \leq i \leq n$. *Exceção*: no caso em que $Y = X$, apenas $X \to x_0Xx_1Xx_2 \ldots Xx_n$ **não** pertence a R'.

Exemplo/eliminação de uma regra

G:

$$P \rightarrow ABA$$
$$A \rightarrow \mathtt{a}A \mid \mathtt{a}$$
$$B \rightarrow \mathtt{b}B\mathtt{c} \mid \lambda$$

G' obtida eliminando-se a regra $A \rightarrow \mathtt{a}$:

Exemplo/eliminação de uma regra

G:

$$P \rightarrow ABA$$
$$A \rightarrow \mathtt{a}A \mid \mathtt{a}$$
$$B \rightarrow \mathtt{b}B\mathtt{c} \mid \lambda$$

G' obtida eliminando-se a regra $A \rightarrow \mathtt{a}$:

$$P \rightarrow ABA \mid AB\mathtt{a} \mid \mathtt{a}BA \mid \mathtt{a}B\mathtt{a}$$
$$A \rightarrow \mathtt{a}A \mid \mathtt{aa}$$
$$B \rightarrow \mathtt{b}B\mathtt{c} \mid \lambda$$

$\Rightarrow$ Como são as derivações de aa em G e em G'?

Formas Normais para GLCs

Que modificações mínimas a regras de gramáticas regulares propiciam gerar qualquer LLC?

Formas Normais para GLCs

Que modificações mínimas a regras de gramáticas regulares propiciam gerar qualquer LLC?

1. Generalizar o formato $X \to aY$ permitinto variável no lugar do terminal $a \Rightarrow$ Forma normal de Chomsky

Formas Normais para GLCs

Que modificações mínimas a regras de gramáticas regulares propiciam gerar qualquer LLC?

1. Generalizar o formato $X \rightarrow aY$ permitinto variável no lugar do terminal $a \Rightarrow$ Forma normal de Chomsky
2. Preservar geração de novo terminal a cada passo, exigindo que o lado direito de uma regra comece com um terminal $\Rightarrow$ Forma normal de Greibach

Formas sentenciais não descrescentes

As formas normais não admitem regras λ. Com isto:

$$\text{se } x \Rightarrow y, \text{ então } |x| \leq |y|$$

Formas sentenciais não descrescentes

As formas normais não admitem regras λ. Com isto:

$$\text{se } x \Rightarrow y, \text{ então } |x| \leq |y|$$

Uma GLC G' em forma normal, obtida de uma GLC G, será tal que $L(G') = L(G) - \{\lambda\}$.

Formas sentenciais não descrescentes

As formas normais não admitem regras λ. Com isto:

$$\text{se } x \Rightarrow y, \text{ então } |x| \leq |y|$$

Uma GLC G' em forma normal, obtida de uma GLC G, será tal que $L(G') = L(G) - \{\lambda\}$.

Se $\lambda \in L(G)$, pode-se acrescentar uma regra apenas para gerar a palavra λ:

- introduzir um novo símbolo de partida P'
- acrescentar as $P' \to \lambda$ e $P' \to w$ para cada w tal que exista a regra $P \to w$ em G'

O conceito de variáveis anuláveis

Como obter, a partir de uma GLC G, uma GLC G' sem regras λ tal que $L(G') = L(G) - \{\lambda\}$?

O conceito de variáveis anuláveis

Como obter, a partir de uma GLC G, uma GLC G' sem regras λ tal que $L(G') = L(G) - \{\lambda\}$?

Definição

Uma variável X é dita ser anulável em uma GLC G sse $X \overset{*}{\Rightarrow}_G \lambda$.

O conceito de variáveis anuláveis

Como obter, a partir de uma GLC G, uma GLC G' sem regras λ tal que $L(G') = L(G) - \{\lambda\}$?

Definição

Uma variável X é dita ser anulável em uma GLC G sse $X \overset{*}{\Rightarrow}_G \lambda$.

Definição

O conjunto das variáveis anuláveis de uma GLC $G = (V, \Sigma, R, P)$, VA_G, é definido recursivamente assim:

- $X \in VA_G$, *se* $X \to \lambda \in R$
- *se* $X \to w \in R$ *e* $w \in VA_G^*$, *então* $X \in VA_G$

Algoritmo para determinar variáveis anuláveis

Cálculo de VA_G

Entrada: uma GLC $G = (V, \Sigma, R, P)$;
Saída: $VA_G = \{X \in V \mid X \stackrel{*}{\Rightarrow} \lambda\}$.
$VA_G \leftarrow \emptyset$;
repita
    $N \leftarrow \{X \in V - VA_G \mid X \rightarrow w \in R \text{ e } w \in VA_G^*\}$;
    $VA_G \leftarrow VA_G \cup N$
até $N = \emptyset$;
retorne VA_G.

Eliminação de regras λ

Seja $G = (V, \Sigma, R, P)$. R' de $G' = (V, \Sigma, R', P)$ tal que $L(G') = L(G) - \{\lambda\}$ é constituído de:

1. cada regra $Y \to z \in R$ tal que $z \neq \lambda$ e $vars(z) \cap VA_G = \emptyset$
2. cada regra $Y \to x_0\gamma_1 x_1 \gamma_2 x_2 \ldots \gamma_n x_n$, em que cada γ_j pode ser X_j ou λ, para $Y \to x_0 X_1 x_1 X_2 x_2 \ldots X_n x_n \in R$, sendo $n > 0$, $X_i \in VA_G$ para $1 \leq i \leq n$, e cada x_i sem variáveis anuláveis. *Exceção*: regra λ **não** pertence a R'.

Eliminação de regras λ

Seja $G = (V, \Sigma, R, P)$. R' de $G' = (V, \Sigma, R', P)$ tal que $L(G') = L(G) - \{\lambda\}$ é constituído de:

1. cada regra $Y \to z \in R$ tal que $z \neq \lambda$ e $vars(z) \cap VA_G = \emptyset$
2. cada regra $Y \to x_0\gamma_1 x_1 \gamma_2 x_2 \dots \gamma_n x_n$, em que cada γ_j pode ser X_j ou λ, para $Y \to x_0 X_1 x_1 X_2 x_2 \dots X_n x_n \in R$, sendo $n > 0$, $X_i \in VA_G$ para $1 \leq i \leq n$, e cada x_i sem variáveis anuláveis.
   Exceção: regra λ **não** pertence a R'.

$\Rightarrow$ Se $P \in VA_G$ então $\lambda \in L(G)$
$L(G) = L(G') \cup \{\lambda\}$

Exemplo/eliminação de regras λ

Seja G:

$$P \rightarrow APB \mid C$$
$$A \rightarrow A\mathtt{aa}A \mid \lambda$$
$$B \rightarrow BB\mathtt{b} \mid C$$
$$C \rightarrow \mathtt{c}C \mid \lambda$$

Exemplo/eliminação de regras λ

Seja G:

$$P \to APB \mid C$$
$$A \to A\mathtt{aa}A \mid \lambda$$
$$B \to BB\mathtt{b} \mid C$$
$$C \to \mathtt{c}C \mid \lambda$$

$VA_G = \{A, C, P, B\}$. G':

$$P \to APB \mid AP \mid AB \mid PB \mid A \mid B \mid C$$
$$A \to A\mathtt{aa}A \mid \mathtt{aa}A \mid A\mathtt{aa} \mid \mathtt{aa}$$
$$B \to BB\mathtt{b} \mid B\mathtt{b} \mid \mathtt{b} \mid C$$
$$C \to \mathtt{c}C \mid \mathtt{c}$$

Regras unitárias e variáveis encadeadas

Definição

Regra unitária: regra da forma $X \to Y$ em que X e Y são variáveis.

Algoritmo para determinar variáveis encadeadas

Cálculo de $enc(X)$

Entrada: uma GLC $G = (V, \Sigma, R, P)$ e uma variável $X \in V$.
Saída: $enc(X)$.
$E \leftarrow \{X\}$;
repita
    $N \leftarrow \{Z \in V - E \mid Y \rightarrow Z \in R \text{ para algum } Y \in E\}$;
    $E \leftarrow E \cup N$
até $\mathcal{N} = \emptyset$
retorne E.

Eliminando regras unitárias

Uma GLC equivalente a $G = (V, \Sigma, R, P)$, sem regras unitárias, é $G' = (V, \Sigma, R', P)$ em que

$R' = \{X \to w \mid w \notin V$ e existe $Y \in enc(X)$ tal que $Y \to w \in R\}$.

Exemplo/eliminação de regras unitárias

GLC para expressões aritméticas:

$$E \rightarrow E+T \mid T$$
$$T \rightarrow T*F \mid F$$
$$F \rightarrow (E) \mid \mathtt{a}$$

Exemplo/eliminação de regras unitárias

GLC para expressões aritméticas:

$$E \rightarrow E+T \mid T$$
$$T \rightarrow T*F \mid F$$
$$F \rightarrow (E) \mid \mathtt{a}$$

Os conjuntos $enc(X)$ para cada variável X são:

- $enc(E) = \{E, T, F\}$;
- $enc(T) = \{T, F\}$;
- $enc(F) = \{F\}$.

Exemplo/eliminação de regras unitárias

GLC para expressões aritméticas:

$$E \rightarrow E{+}T \mid T$$
$$T \rightarrow T{*}F \mid F$$
$$F \rightarrow (E) \mid \mathtt{a}$$

Os conjuntos $enc(X)$ para cada variável X são:

- $enc(E) = \{E, T, F\}$;
- $enc(T) = \{T, F\}$;
- $enc(F) = \{F\}$.

GLC equivalente, sem regras unitárias:

$$E \rightarrow E{+}T \mid T{*}F \mid (E) \mid \mathtt{a}$$
$$T \rightarrow T{*}F \mid (E) \mid \mathtt{a}$$
$$F \rightarrow (E) \mid \mathtt{a}$$

Interação entre os métodos de eliminação

a) Ao se eliminar regras λ podem aparecer regras unitárias.
Exemplo: GLC com as regras $A \to BC$ e $B \to \lambda$.

Interação entre os métodos de eliminação

a) Ao se eliminar regras λ podem aparecer regras unitárias. Exemplo: GLC com as regras $A \to BC$ e $B \to \lambda$.

b) Ao se eliminar regras unitárias não podem aparecer regras λ, visto que novas regras só contêm o lado direito de regras já existentes (que não podem ser λ).

Interação entre os métodos de eliminação

a) Ao se eliminar regras λ podem aparecer regras unitárias. Exemplo: GLC com as regras $A \to BC$ e $B \to \lambda$.

b) Ao se eliminar regras unitárias não podem aparecer regras λ, visto que novas regras só contêm o lado direito de regras já existentes (que não podem ser λ).

c) Ao se eliminar regras λ podem aparecer variáveis inúteis. Exemplo: o do item (a), caso $B \to \lambda$ seja a única regra B.

Interação entre os métodos de eliminação

a) Ao se eliminar regras λ podem aparecer regras unitárias. Exemplo: GLC com as regras $A \to BC$ e $B \to \lambda$.

b) Ao se eliminar regras unitárias não podem aparecer regras λ, visto que novas regras só contêm o lado direito de regras já existentes (que não podem ser λ).

c) Ao se eliminar regras λ podem aparecer variáveis inúteis. Exemplo: o do item (a), caso $B \to \lambda$ seja a única regra B.

d) Ao se eliminar regras unitárias podem aparecer variáveis inúteis. Exemplo: GLC que contém $A \to B$ e B não aparece do lado direito de nenhuma outra regra (B torna-se inútil).

Interação entre os métodos de eliminação

a) Ao se eliminar regras λ podem aparecer regras unitárias. Exemplo: GLC com as regras $A \to BC$ e $B \to \lambda$.

b) Ao se eliminar regras unitárias não podem aparecer regras λ, visto que novas regras só contêm o lado direito de regras já existentes (que não podem ser λ).

c) Ao se eliminar regras λ podem aparecer variáveis inúteis. Exemplo: o do item (a), caso $B \to \lambda$ seja a única regra B.

d) Ao se eliminar regras unitárias podem aparecer variáveis inúteis. Exemplo: GLC que contém $A \to B$ e B não aparece do lado direito de nenhuma outra regra (B torna-se inútil).

e) Ao se eliminar variáveis inúteis, não podem aparecer novas regras, inclusive regras λ ou unitárias.

Garantido consistência das eliminações

A seguinte sequência de eliminações para $G = (V, \Sigma, R, P)$:

1. eliminar regras λ
2. eliminar regras unitárias
3. eliminar símbolos inúteis

produz uma GLC para $L(G) - \{\lambda\}$ cujas regras são das formas:

- $X \to a$ para $a \in \Sigma$
- $X \to w$ para $|w| \geq 2$

Gramática na forma normal de Chomsky

Definição

Uma GLC $G = (V, \Sigma, R, P)$ é dita estar na forma normal de Chomsky (FNC) se não contém variáveis inúteis e cada uma de suas regras está em uma das formas:

- $X \to YZ$ *para* $Y, Z \in V$
- $X \to a$ *para* $a \in \Sigma$

Transformação para a forma normal de Chomsky

Para obter uma GLC na FNC que gere $L(G) - \{\lambda\}$ a partir de G:

1. eliminar regras λ
2. eliminar regras unitárias
3. eliminar variáveis inúteis

Transformação para a forma normal de Chomsky

Para obter uma GLC na FNC que gere $L(G) - \{\lambda\}$ a partir de G:

1. eliminar regras λ
2. eliminar regras unitárias
3. eliminar variáveis inúteis
4. modificar cada regra $X \rightarrow w$ tal que $|w| \geq 2$, se necessário, de forma que ela fique contendo apenas variáveis

Transformação para a forma normal de Chomsky

Para obter uma GLC na FNC que gere $L(G) - \{\lambda\}$ a partir de G:

1. eliminar regras λ
2. eliminar regras unitárias
3. eliminar variáveis inúteis
4. modificar cada regra $X \to w$ tal que $|w| \geq 2$, se necessário, de forma que ela fique contendo apenas variáveis
5. substituir cada regra $X \to Y_1 Y_2 \ldots Y_n$, $n \geq 3$, em que cada Y_i é uma variável, pelo conjunto das regras: $X \to Y_1 Z_1$, $Z_1 \to Y_2 Z_2$, $\ldots$, $Z_{n-2} \to Y_{n-1} Y_n$, em que Z_1, Z_2, $\ldots$, Z_{n-2} são variáveis novas

Exemplo/obtenção de gramática na FNC

Seja a GLC G:

$$L \rightarrow (S)$$
$$S \rightarrow SE \mid \lambda$$
$$E \rightarrow \mathtt{a} \mid L$$

Exemplo/obtenção de gramática na FNC

Seja a GLC G:

$$L \rightarrow (S)$$
$$S \rightarrow SE \mid \lambda$$
$$E \rightarrow \mathtt{a} \mid L$$

Após a eliminação de regras λ:

$$L \rightarrow (S) \mid ()$$
$$S \rightarrow SE \mid E$$
$$E \rightarrow \mathtt{a} \mid L$$

Exemplo/obtenção de gramática na FNC

Seja a GLC G:

$$L \rightarrow (S)$$
$$S \rightarrow SE \mid \lambda$$
$$E \rightarrow \mathtt{a} \mid L$$

Após a eliminação de regras λ:

$$L \rightarrow (S) \mid ()$$
$$S \rightarrow SE \mid E$$
$$E \rightarrow \mathtt{a} \mid L$$

Como $enc(L) = \{L\}$, $enc(S) = \{S, E, L\}$ e $enc(E) = \{E, L\}$:

$$L \rightarrow (S) \mid ()$$
$$S \rightarrow SE \mid \mathtt{a} \mid (S) \mid ()$$
$$E \rightarrow \mathtt{a} \mid (S) \mid ()$$

Exemplo/obtenção de gramática na FNC(continuação)

$$L \rightarrow (S) \mid ()$$
$$S \rightarrow SE \mid \mathtt{a} \mid (S) \mid ()$$
$$E \rightarrow \mathtt{a} \mid (S) \mid ()$$

Finalmente:

$$L \rightarrow AX \mid AB$$
$$S \rightarrow SE \mid \mathtt{a} \mid AX \mid AB$$
$$E \rightarrow \mathtt{a} \mid AX \mid AB$$
$$X \rightarrow SB$$
$$A \rightarrow ($$
$$B \rightarrow)$$

Eliminação de regras recursivas à esquerda

Sejam, a seguir, todas as regras X de uma GLC G:

$$X \rightarrow Xy_1 \mid Xy_2 \mid \ldots \mid Xy_n \mid w_1 \mid w_2 \mid \ldots \mid w_k$$

$n > 0$, $k > 0$, em que nenhum w_i começa com X.

Eliminação de regras recursivas à esquerda

Sejam, a seguir, todas as regras X de uma GLC G:

$$X \rightarrow Xy_1 \mid Xy_2 \mid \ldots \mid Xy_n \mid w_1 \mid w_2 \mid \ldots \mid w_k$$

$n > 0$, $k > 0$, em que nenhum w_i começa com X.

Utilizando-se recursão à direita, em vez de recursão à esquerda:

$$X \rightarrow w_1 Z \mid w_2 Z \mid \ldots \mid w_k Z$$

$$Z \rightarrow y_1 Z \mid y_2 Z \mid \ldots \mid y_n Z \mid \lambda$$

em que Z é uma variável nova.

Eliminação de regras recursivas à esquerda

Sejam, a seguir, todas as regras X de uma GLC G:

$$X \rightarrow Xy_1 \mid Xy_2 \mid \ldots \mid Xy_n \mid w_1 \mid w_2 \mid \ldots \mid w_k$$

$n > 0$, $k > 0$, em que nenhum w_i começa com X.

Utilizando-se recursão à direita, em vez de recursão à esquerda:

$$X \rightarrow w_1 Z \mid w_2 Z \mid \ldots \mid w_k Z$$
$$Z \rightarrow y_1 Z \mid y_2 Z \mid \ldots \mid y_n Z \mid \lambda$$

em que Z é uma variável nova.

Eliminando-se a regra λ:

$$X \rightarrow w_1 Z \mid w_2 Z \mid \ldots \mid w_k Z \mid w_1 \mid w_2 \mid \ldots \mid w_k$$
$$Z \rightarrow y_1 Z \mid y_2 Z \mid \ldots \mid y_n Z \mid y_1 \mid y_2 \mid \ldots \mid y_n$$

Exemplo/eliminação de regras recursivas à esquerda

G: $E \rightarrow E{+}E \mid E{*}E \mid (E) \mid \mathtt{a}$

Exemplo/eliminação de regras recursivas à esquerda

G: $E \rightarrow E{+}E \mid E{*}E \mid (E) \mid \mathtt{a}$

Eliminando-se recursão à esquerda:

$$E \rightarrow (E)Z \mid \mathtt{a}Z \mid (E) \mid \mathtt{a}$$
$$Z \rightarrow {+}EZ \mid {*}EZ \mid {+}E \mid {*}E$$

Eliminação de variável no lado direito de regra

Seja $G = (V, \Sigma, R, P)$ tal que $X \rightarrow uYv \in R$, $Y \in V$ e $Y \neq X$.

Sejam $Y \rightarrow w_1 \mid w_2 \mid \ldots \mid w_n$ todas as regras Y em R.

Eliminação de variável no lado direito de regra

Seja $G = (V, \Sigma, R, P)$ tal que $X \rightarrow uYv \in R$, $Y \in V$ e $Y \neq X$.

Sejam $Y \rightarrow w_1 \mid w_2 \mid \ldots \mid w_n$ todas as regras Y em R.

R pode ser substituído por

$$(R - \{X \rightarrow uYv\}) \cup \{X \rightarrow uw_1v \mid uw_2v \mid \ldots \mid uw_nv\}.$$

Gramática na forma normal de Greibach

Definição

Uma GLC $G = (V, \Sigma, R, P)$ é dita estar na forma normal de Greibach (FNG) se não tem variáveis inúteis e todas as suas regras têm a forma $X \to ay$ para $a \in \Sigma$ e $y \in V^*$.

Gramática na forma normal de Greibach

Definição

Uma GLC $G = (V, \Sigma, R, P)$ é dita estar na forma normal de Greibach (FNG) se não tem variáveis inúteis e todas as suas regras têm a forma $X \to ay$ para $a \in \Sigma$ e $y \in V^*$.

- Uma forma sentencial em uma DME de uma gramática na FNG é da forma xy, em que $x \in \Sigma^+$ e $y \in V^*$.
- A cada passo de uma DME, concatena-se um terminal a mais ao prefixo x.
- O tamanho de uma derivação de uma palavra $w \in \Sigma^+$ é $|w|$.

Transformação para a forma normal de Greibach

Para obter uma GLC na FNG que gere $L(G) - \{\lambda\}$ a partir de uma GLC G:

1. eliminar regras λ
2. eliminar regras unitárias
3. eliminar variáveis inúteis

Transformação para a forma normal de Greibach

Para obter uma GLC na FNG que gere $L(G) - \{\lambda\}$ a partir de uma GLC G:

1. eliminar regras λ
2. eliminar regras unitárias
3. eliminar variáveis inúteis
4. numerar as variáveis a partir de 1, com a variável de partida recebendo o número 1

Transformação para a forma normal de Greibach

Para obter uma GLC na FNG que gere $L(G) - \{\lambda\}$ a partir de uma GLC G:

1. eliminar regras λ
2. eliminar regras unitárias
3. eliminar variáveis inúteis
4. numerar as variáveis a partir de 1, com a variável de partida recebendo o número 1
5. Para cada variável X, na ordem da numeração:
   - se existe $X \to Yw$ em que número de Y é menor que o de X, substituir Y
   - se existe $X \to Xw$, eliminar a recursão à esquerda

Transformação para a forma normal de Greibach

Para obter uma GLC na FNG que gere $L(G) - \{\lambda\}$ a partir de uma GLC G:

1. eliminar regras λ
2. eliminar regras unitárias
3. eliminar variáveis inúteis
4. numerar as variáveis a partir de 1, com a variável de partida recebendo o número 1
5. Para cada variável X, na ordem da numeração:
   - se existe $X \to Yw$ em que número de Y é menor que o de X, substituir Y
   - se existe $X \to Xw$, eliminar a recursão à esquerda
6. para cada variável X, em ordem decrescente da numeração: se há regra $X \to Yw$, em que Y é variável, substituir Y

Transformação para a forma normal de Greibach

Para obter uma GLC na FNG que gere $L(G) - \{\lambda\}$ a partir de uma GLC G:

1. eliminar regras λ
2. eliminar regras unitárias
3. eliminar variáveis inúteis
4. numerar as variáveis a partir de 1, com a variável de partida recebendo o número 1
5. Para cada variável X, na ordem da numeração:
   - se existe $X \to Yw$ em que número de Y é menor que o de X, substituir Y
   - se existe $X \to Xw$, eliminar a recursão à esquerda
6. para cada variável X, em ordem decrescente da numeração: se há regra $X \to Yw$, em que Y é variável, substituir Y
7. para cada variável nova Z, criada eliminando-se recursão à esquerda: se há regra $Z \to Yw$, em que Y é variável, substituir Y

Transformação para a forma normal de Greibach

Para obter uma GLC na FNG que gere $L(G) - \{\lambda\}$ a partir de uma GLC G:

1. eliminar regras λ
2. eliminar regras unitárias
3. eliminar variáveis inúteis
4. numerar as variáveis a partir de 1, com a variável de partida recebendo o número 1
5. Para cada variável X, na ordem da numeração:
   - se existe $X \to Yw$ em que número de Y é menor que o de X, substituir Y
   - se existe $X \to Xw$, eliminar a recursão à esquerda
6. para cada variável X, em ordem decrescente da numeração: se há regra $X \to Yw$, em que Y é variável, substituir Y
7. para cada variável nova Z, criada eliminando-se recursão à esquerda: se há regra $Z \to Yw$, em que Y é variável, substituir Y
8. em cada regra $X \to aw$, substituir terminais em w por variáveis

Exemplo/transformação para a FNG

Seja a GLC:

$$E \rightarrow E+T \mid T$$
$$T \rightarrow T*F \mid F$$
$$F \rightarrow (E) \mid \mathtt{a}$$

Exemplo/transformação para a FNG

Seja a GLC:

$$E \to E{+}T \mid T$$
$$T \to T{*}F \mid F$$
$$F \to (E) \mid \mathtt{a}$$

Após os 3 primeiros passos:

$$E \to E{+}T \mid T{*}F \mid (E) \mid \mathtt{a}$$
$$T \to T{*}F \mid (E) \mid \mathtt{a}$$
$$F \to (E) \mid \mathtt{a}$$

Exemplo/transformação para a FNG

Seja a GLC:

$$E \to E{+}T \mid T$$
$$T \to T{*}F \mid F$$
$$F \to (E) \mid \mathtt{a}$$

Após os 3 primeiros passos:

$$E \to E{+}T \mid T{*}F \mid (E) \mid \mathtt{a}$$
$$T \to T{*}F \mid (E) \mid \mathtt{a}$$
$$F \to (E) \mid \mathtt{a}$$

Após numerar na ordem E, T, F, no passo 4, no passo 5 elimina-se recursão à esquerda nas regras E:

$$E \to T{*}FZ_1 \mid (E)Z_1 \mid \mathtt{a}Z_1 \mid T * F \mid (E) \mid \mathtt{a}$$
$$Z_1 \to +TZ_1 \mid +T$$

Exemplo/transformação para a FNG

Seja a GLC:

$$E \to E{+}T \mid T$$
$$T \to T{*}F \mid F$$
$$F \to (E) \mid \mathtt{a}$$

Após os 3 primeiros passos:

$$E \to E{+}T \mid T{*}F \mid (E) \mid \mathtt{a}$$
$$T \to T{*}F \mid (E) \mid \mathtt{a}$$
$$F \to (E) \mid \mathtt{a}$$

Após numerar na ordem E, T, F, no passo 4, no passo 5 elimina-se recursão à esquerda nas regras E:

$$E \to T{*}FZ_1 \mid (E)Z_1 \mid \mathtt{a}Z_1 \mid T * F \mid (E) \mid \mathtt{a}$$
$$Z_1 \to {+}TZ_1 \mid {+}T$$

Ainda no passo 5 elimina-se recursão à esquerda nas regras T:

$$T \to (E)Z_2 \mid \mathtt{a}Z_2 \mid (E) \mid \mathtt{a}$$
$$Z_2 \to {*}FZ_2 \mid {*}F$$

Exemplo/transformação para a FNG (continuação)

No passo 6, elimina-se T em $E \to T*FZ_1$ e $E \to T*F$.

$$E \to (E)Z_2*FZ_1 \mid \mathtt{a}Z_2*FZ_1 \mid (E)*FZ_1 \mid \mathtt{a}*FZ_1$$
$$E \to (E)Z_2*F \mid \mathtt{a}Z_2*F \mid (E)*F \mid \mathtt{a}*F$$
$$E \to (E)Z_1 \mid \mathtt{a}Z_1 \mid (E) \mid \mathtt{a}$$
$$T \to (E)Z_2 \mid \mathtt{a}Z_2 \mid (E) \mid \mathtt{a}$$
$$F \to (E) \mid \mathtt{a}$$
$$Z_1 \to +TZ_1 \mid +T$$
$$Z_2 \to *FZ_2 \mid *F$$

Exemplo/transformação para a FNG (continuação)

No passo 6, elimina-se T em $E \to T*FZ_1$ e $E \to T*F$.

$$E \to (E)Z_2*FZ_1 \mid \mathtt{a}Z_2*FZ_1 \mid (E)*FZ_1 \mid \mathtt{a}*FZ_1$$
$$E \to (E)Z_2*F \mid \mathtt{a}Z_2*F \mid (E)*F \mid \mathtt{a}*F$$
$$E \to (E)Z_1 \mid \mathtt{a}Z_1 \mid (E) \mid \mathtt{a}$$
$$T \to (E)Z_2 \mid \mathtt{a}Z_2 \mid (E) \mid \mathtt{a}$$
$$F \to (E) \mid \mathtt{a}$$
$$Z_1 \to +TZ_1 \mid +T$$
$$Z_2 \to *FZ_2 \mid *F$$

Não há o que fazer no passo 7. No último passo obtém-se:

$$E \to (EPZ_2VFZ_1 \mid \mathtt{a}Z_2VFZ_1 \mid (EPVFZ_1 \mid \mathtt{a}VFZ_1$$
$$E \to (EPZ_2VF \mid \mathtt{a}Z_2VF \mid (EPVF \mid \mathtt{a}VF$$
$$E \to (EPZ_1 \mid \mathtt{a}Z_1 \mid (EP \mid \mathtt{a}$$
$$T \to (EPZ_2 \mid \mathtt{a}Z_2 \mid (EP \mid \mathtt{a}$$
$$F \to (EP \mid \mathtt{a}$$
$$Z_1 \to +TZ_1 \mid +T$$
$$Z_2 \to *FZ_2 \mid *F$$
$$V \to *$$
$$P \to)$$

Linguagens Formais e Autômatos

Semana 13: Gramáticas Livres do Contexto - parte 2

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Linguagens Formais e Autômatos

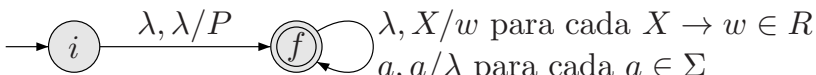
Semana 14: Autômatos de pilha e o Lema do Bombeamento para LLC

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Obtenção de AP a partir de GLC

AP que reconhece a linguagem gerada por (V, Σ, R, P):

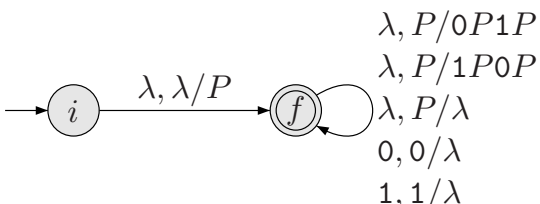


Obtenção de AP a partir de GLC/Exemplo

$G = (\{P\}, \{0, 1\}, R, P)$, em que R consta de:

$$P \to 0P1P \mid 1P0P \mid \lambda$$

Um AP que reconhece $L(G)$:



Outro AP a partir da GLC

$G = (\{P\}, \{0, 1\}, R, P)$, em que R consta de:

$$P \to 0P1P \mid 1P0P \mid \lambda$$

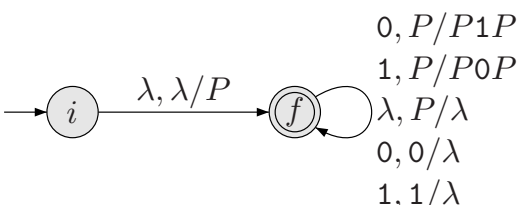
Regra $X \to ay \Rightarrow$ transição $[f, y] \in \delta(f, a, X)$

Outro AP a partir da GLC

$G = (\{P\}, \{0, 1\}, R, P)$, em que R consta de:

$$P \to 0P1P \mid 1P0P \mid \lambda$$

Regra $X \to ay \Rightarrow$ transição $[f, y] \in \delta(f, a, X)$



O lema do bombeamento para LLCs

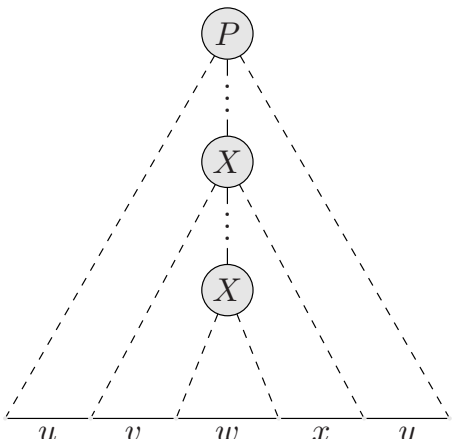
Lema do bombeamento

Seja L uma LLC. Então existe $k > 0$ tal que para qualquer $z \in L$ com $|z| \geq k$ existem u, v, w, x e y tais que:

- $z = uvwxy$;
- $|vwx| \leq k$;
- $vx \neq \lambda$; e
- $uv^i wx^i y \in L$ para todo $i \geq 0$.

O lema do bombeamento para LLCs

Existe $k > 0$ tal que qualquer palavra $z \in L$ com $|z| \geq k$ terá uma AD da forma:



O lema do bombeamento para LLCs (continuação)

- Como G não tem regras λ, pelo menos um dentre v e x é diferente de λ: $vx \neq \lambda$.
- A repetição do rótulo X em uma subAD com raiz de rótulo X ocorre, no mais tardar, quando a subpalavra gerada correspondente à subAD, vwx, tem k símbolos: $|vwx| \leq k$.
- Pela estrutura da AD, vê-se que:
  - $P \stackrel{*}{\Rightarrow} uXy$;
  - $X \stackrel{*}{\Rightarrow} vXx$; e
  - $X \stackrel{*}{\Rightarrow} w$.

  Tem-se, então, que $P \stackrel{*}{\Rightarrow} uv^i wx^i y$, $i \geq 0$. Assim, $uv^i wx^i y \in L$ para todo $i \geq 0$.

Exemplo de uso do lema do bombeamento

Teorema

$L = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \in \mathbf{N}\}$ *não é LLC.*

Demonstração

Suponha que L seja uma LLC. Seja k a constante do LB e $z = \mathtt{a}^k\mathtt{b}^k\mathtt{c}^k$. *Como* $|z| > k$, *sejam u, v, w, x e y tais que* $z = uvwxy$, $|vwx| \leq k$, *e* $vx \neq \lambda$. *Considera- se dois casos:*

- *vx contém algum* a. *Como* $|vwx| \leq k$, *vx não contém* cs. *Portanto,* uv^2wx^2y *contém mais* as *que* cs. *Assim,* $uv^2wx^2y \notin L$.
- *vx não contém* a. *Como* $vx \neq \lambda$, uv^2wx^2y *contém menos* as *que* bs *e/ou* cs. *Dessa forma,* $uv^2wx^2y \notin L$.

Logo, em qualquer caso $uv^2wx^2y \notin L$, *contrariando o LB. Portanto, L não é LLC.*

Exemplo de uso do lema do bombeamento

Teorema

$L = \{0^n \mid n \text{ é primo}\}$.

Demonstração

Suponha que L seja uma LLC. Seja k a constante do LB, e seja $z = 0^n$, *em que n é um número primo maior que k. Como* $|z| > k$, *para provar que L não é livre do contexto, basta então supor que* $z = uvwxy$, $|vwx| \leq k$ *e* $vx \neq \lambda$, *e encontrar um i tal que* $uv^iwx^iy \notin L$, *contrariando o LB. Pelas informações anteriores, tem-se que* $uv^iwx^iy = 0^{n+(i-1)(|vx|)}$ *(pois* $z = 0^n$*). Assim, i deve ser tal que* $n + (i-1)|vx|$ *não seja um número primo. Ora, para isso, basta fazer* $i = n+1$, *obtendo-se* $n+(i-1)|vx| = n+n|vx| = n(1+|vx|)$, *que não é primo (pois* $|vx| > 0$*). Desse modo,* $uv^{n+1}wx^{n+1}y \notin L$, *contradizendo o LB. Logo, L não é LLC.*

Algumas propriedades de fechamento

Teorema

A classe das LLCs é fechada sob:

- *união,*
- *concatenação,*
- *fecho de Kleene.*

Demonstração

Trivial, usando GLCs.

Não fechamento das LLCs

A classe das LLCs não é fechada sob:

- Interseção.
  - $L_1 = \{a^n b^n c^k \mid n, k \geq 0\}$ é LLC
  - $L_2 = \{a^n b^k c^k \mid n, k \geq 0\}$ é LLC
  - $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ não é LLC

Não fechamento das LLCs

A classe das LLCs não é fechada sob:

- Interseção.
  - $L_1 = \{a^n b^n c^k \mid n, k \geq 0\}$ é LLC
  - $L_2 = \{a^n b^k c^k \mid n, k \geq 0\}$ é LLC
  - $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ não é LLC
- Complementação.
  - $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

Um teorema importante

Teorema

Seja L uma LLC e R uma linguagem regular. Então $L \cap R$ é uma LLC.

Demonstração

Produto de APN e AFD...

Um teorema importante

Teorema

Seja L uma LLC e R uma linguagem regular. Então $L \cap R$ é uma LLC.

Demonstração

Produto de APN e AFD...

Seja $L = \{w \in \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^* \mid n_{\mathtt{a}}(w) = n_{\mathtt{b}}(w) = n_{\mathtt{c}}(w)\}$.

Suponha que L seja uma LLC. Então, como $a^*b^*c^*$ é regular, $L \cap a^*b^*c^*$ é LLC. Mas, $L \cap a^*b^*c^* = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \in \mathbf{N}\}$, que não é LLC. Logo, L não é LLC.

Problemas decidíveis e indecidíveis para LLcs

Problemas decidíveis:

- Determinar se $w \in L$, para qualquer LLC L e palavra w.
- Determinar se $L = \emptyset$, para qualquer LLC L.

Problemas indecidíveis:

- Determinar se G é ambígua, para qualquer GLC G.
- Determinar se $L = \Sigma^*$, para qualquer LLC L.
- Verificar se $L_1 \cap L_2 = \emptyset$, para quaisquer LLCs L_1 e L_2.
- Determinar se $L_1 \subseteq L_2$, para quaisquer LLCs L_1 e L_2.
- Determinar se $L_1 = L_2$, para quaisquer LLCs L_1 e L_2.

Linguagens Formais e Autômatos

Semana 14: Autômatos de pilha e o Lema do Bombeamento para LLC

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Linguagens Formais e Autômatos

Semana 15: Revisão e exercícios

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia

Orientações para realização de provas

Tiago Januario

Departamento de Ciência da Computação
Universidade Federal da Bahia