Resumos TCOM

MIEIC

18 de fevereiro de 2021

Conteúdo

1 Introdução

Este documento contém resumos dos conteúdos lecionados em TCOM no ano letivo 2018/2019. Este documento contém apenas resumos dos conteúdos abordados na unidade curricular e não substitui o estudo mais aprofundado destes conteúdos.

2 Provas

Métodos de Prova [if H then C (H $\rightarrow$ C)]

- Por Contradição [H and not C implies falsehood]
- Por Contra-exemplo: mostrar um exemplo que prova que a proposição é falsa
- Por Contra-positivo [if not C then not H]: provar um é provar o outro
- Por Indução (provar S(n)):
  - Caso base (Provar S(i), sendo i o 1º valor, normalmente 0 ou 1)
  - Passo indutivo (Assumindo a hipótese, prova-se S(n+1))
  - Sendo n um número geral, a propriedade aplica-se a todos os n

3 Deterministic Finite Automata (DFA)

Alfabeto ($\sum$) é um conjunto finito de símbolos não vazios.

String é uma sequência finita de símbolos selecionados do alfabeto.

Linguagem L sobre um alfabeto ($\sum$) é um subconjunto de $\sum^*$ ($L \subseteq \sum^*$)

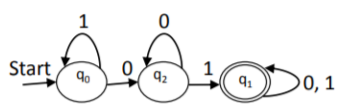
- $\sum^* = \sum^0 \bigcup \sum^+ = \sum^0 \bigcup \sum^1 \bigcup \sum^2 \bigcup ...$
- Qualquer problema pode ser convertido numa linguagem e vice-versa.

DFA é determinista: num estado, para cada input, existe apenas uma possível transição.

DFA: $A = (Q, \sum, \delta, q_0, F)$

- Q é um conjunto de estados.
- $\sum$ é o alfabeto.
- δ é a função de transição, de estados e inputs para estados.
  - Exemplo: $p = \delta(q, a)$
- $q_0 \in Q$ é o estado inicial.
- $F \subseteq Q$ é o conjunto de estados finais/de aceitação.

Diagrama de transição e respetiva tabela:



	0	1
$\rightarrow q_0$	q_2	q_0
$*q_1$	q_1	q_1
q_2	q_2	q_1

Se uma linguagem L é L(A) para um DFA A, então é uma linguagem regular.

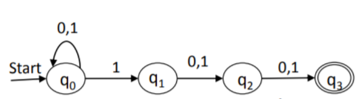
4 Non-deterministic Finite Automata (NFA)

FA não determinista:

- Pode estar em mais do que um estado ao mesmo tempo.
- A partir de um estado, com um input, pode ir para vários estados.
- No final, basta que um dos etados alcançados seja o estado final.

$NFA = (Q, \sum, \delta, q_0, F)$

- Igual ao DFA, exceto que a função de transição δ retorna um sub-conjunto de Q, em vez de um único estado.
- Tabela de transição usa conjuntos de estados.



	0	1
$\rightarrow q_0$	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_3\}$	$\{q_3\}$
$*q_3$	$\varnothing$	$\varnothing$

$$NFA \quad A = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_3\})$$

$$\delta(q_0, 0) = \{q_0\}$$
$$\delta(q_0, 1) = \{q_0, q_1\}$$
$$\delta(q_1, 0) = \{q_2\}$$
$$\delta(q_1, 1) = \{q_2\}$$
$$\delta(q_2, 0) = \{q_3\}$$
$$\delta(q_2, 1) = \{q_3\}$$
$$\delta(q_3, 0) = \emptyset$$
$$\delta(q_3, 1) = \emptyset$$

- Para converter um NFA num DFA usa-se a técnica de construção de subconjuntos: se o NFA tem n estados, o DFA terá, no máximo, 2^n estados, incluindo o estado morto ($\emptyset$).

Estado morto (Dead State) é um estado de não aceitação com auto-transições para todos os símbolos do alfabeto. É usado para capturar erros num DFA.

5 ϵ-NFAs

ϵ-NFA: NFA com transições ϵ

ϵ-NFA E $= (Q, \sum, \delta, q_0, F)$

- Maior diferença é que a função de transição δ lida com ϵ.
  - $\delta(q, a)$: estado $qinQ$ e $ain \sum \bigcup \{\epsilon\}$
- ϵ representa transições espontâneas.
- Para saber quais os estados que conseguimos alcançar a partir de um estado q com ϵ, calculamos o ϵ-close(q). Exemplos:
  - ϵ-close(q_0) $= \{q_0, q_1\}$
  - ϵ-close(q_3) $= \{q_3, q_5\}$

Para um dado ϵ-NFA existe sempre um DFA equivalente.

6 Regular Expressions

Uma alternativa equivalente aos NFAs (incluindo ϵ-NFAs) e DFAs.

Operações sobre linguagens:

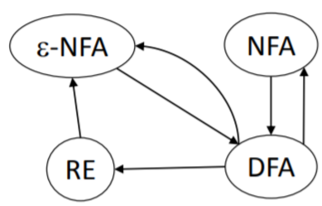
- União ($\bigcup$): $L = \{001, 10\}, M = \{\epsilon, 001\}, LUM = \{\epsilon, 001, 10\}$
- Concatenação (.): $LM = L.M = \{001, 10, 001001, 10001\}$
- Fecho (*): $L = \{0, 1\}, L^* alinguagemdasstringsbinrias$

ϵ e $\emptyset$ são expressões regulares ($L(\epsilon) = \{\epsilon\}$ e $L(\emptyset) = \emptyset$)

Operadores das Expressões Regulares:

- * (zero ou mais)
- + (um ou mais)
- . (concatenação - pode ser omitido)
- Precedência dos operadores: * -> . -> + (podem ser usados parêntesis para forçar uma ordem)

Todas as linguagens definidas por FAs podem ser definidas por Expressões Regulares e vice-versa:



Dois métodos para converter um DFA numa RE:

- Construção de Caminhos ($R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)}(R_{kk}^{(k-1)})R_{kj}^{(k-1)}$)
- Eliminação de Estados

Duas REs são equivalentes se definem a mesma linguagem.

Regras/Leis algébricas para REs:

- Identidade: $\emptyset + L = L + \emptyset = L$; $\epsilon L = L\epsilon = L$
- Absorção: $L\emptyset = \emptyset L = \emptyset$
- Distributiva: L(M + N) = LM + LN
- Idempotência: L + L = L
- $(L^*)^* = L^*, \emptyset^* = \epsilon; \epsilon^* = \epsilon; L^+ = LL^* = L^*L; L^* = L^+ + \epsilon$

7 Regular Languages' Properties

Linguagens regulares são aquelas que podem ser representadas por DFAs, NFAs, ϵ-NFAs ou REs.

Pumping Lemma:

- Dada uma linguagem regular infinita L, existe pelo menos um n (pumping length), para cada string $w \in L$ com comprimento $|w| \geq n$, podemos escrever $w = xyz$ com $|xy| \leq n$ e $|y| \geq 1 (y \neq \epsilon)$, tal que: $xy^k z \in L, k = 0, 1, 2, ...$
- Ou seja, podemos encontrar uma string y, que pode ser repetida ou removida, produzindo strings que pertencem à linguagem.
- Linguagens finitas são linguagens regulares.
- O Pumping Lemma pode ser usado para mostrar que uma linguagem infinita não é regular. Mas não pode ser usado para mostrar que a linguagem é regular.
- O Pumping Lemma é uma condição necessária, mas não suficiente para mostrar que uma linguagem é regular.

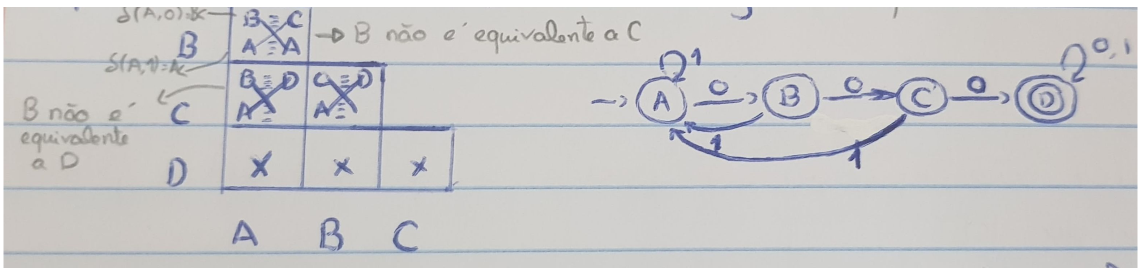
Como provar que uma linguagem A é não regular?

- Por contradição:
  - Assume-se que A é regular.
  - Como A é regular, tem um pumping length (p).
  - Todas as strings maiores que p podem ser bombeadas.
  - Encontra-se uma string s de A tal que $|s| \geq p$.
  - Divide-se s em xyz.
  - Mostra-se que $xy^i z \in A$ para algum i.
  - Depois, consideram-se todas as formas de s ser dividida em xyz.
  - Prova-se que nenhuma das divisões satisfaz as três condições da bombagem ao mesmo tempo.
  - Logo, s não pode ser bombeada: **Contradição**.
- Condições da Bombagem:
  - 1: $xy^i z \in A$, para todo o $i \geq 0$
  - 2: $|y| > 0$
  - 3: $|xy| \leq p$ (pumping length)

Operações com Linguagens Regulares:

- União, Interseção, Complemento, Diferença.
- Reverso (L^R).
- Closure (*) e Concatenação.
- Homomorfismo e Homomorfismo Inverso ($L(h(r)) = h(L(r))$).

Equivalência de estados e minimização (exemplo):



- D não pode ser equivalente a outro estado não final.

8 Context-free Grammar (CFG)

Notação que permite especificar linguagens não regulares (algumas).

CFG G = (V, T, P, S) -> tuplo das CFGs

- V: variáveis da linguagem
- T: terminais, ou seja, símbolos usados nas strings (alfabeto)
- P: produções ou regras da linguagem
- S: símbolo (variável) inicial

Derivação: a partir de uma string, aplicam-se as "regras"da linguagem

- Leftmost ($\underset{lm}{\overset{*}{\Longrightarrow}}$): substituem-se primeiro as variáveis à esquerda.
- Rightmost ($\underset{rm}{\overset{*}{\Longrightarrow}}$): substituem-se primeiro as variáveis à direita.

Syntax trees (árvores de análise)

- Estrutura de dados mais usada para representar o programa de input num compilador
- Cada nó representa uma variável da gramática, um terminal ou ϵ

Eliminação de ambiguidade: podem-se adicionar novas variáveis para distinguir níveis de prioridade e regras de associação.

Uma CFL é ambígua se todas as gramáticas para L são ambíguas.

9 Pushdown Automata (PDA)

O PDA (pushdown automata) é um ϵ-NFA com uma stack de símbolos

- Adiciona a possibilidade de memorizar uma quantidade infinita de informação.
- O PDA só tem acesso ao topo da stack (LIFO).
- Como funciona?
  - A unidade de controlo lê e consome os símbolos do input.
  - Transição para um novo estado baseado no estado atual, símbolo do input e símbolo no topo da stack.
  - Transições espontâneas com ϵ.
  - Topo da stack substituído por símbolos.

PDA P = $(Q, \sum, \Gamma, \delta, q_0, Z_0, F)$

- Q: conjunto dos estados.
- $\sum$: alfabeto.
- Γ: alfabeto finito da stack.
- δ: função de transição.
- q_0: estado inicial.
- Z_0: símbolo incial da stack.
- F: conjunto dos estados finais (se for um PDA de estado final).

Transições representadas cmo a, X/α

- a: símbolo do input consumido.
- X: topo da stack (que será retirado - pop).
- α: string a colocar na stack (push) - valor mais à esquerda ficará no topo da stack.

Descrição instantânea (q, ω, γ) - (ID):

- q: estado.
- ω: input restante.
- γ: conteúdo da stack.

As CFLs definidas por uma CFG são as linguagens aceites por um PDA por empty stack e também aceites por um PDA por final state.

Conversão de PDAs em CFGs:

- O evento principal do processamento de um PDA é retirar um símbolo da stack enquanto se consome o input.
- Adicionam-se variáveis à gramática para cada:
  - eliminação de um símbolo da stack X.
  - transiçã de p para q eliminando X, representado pelo símbolo composto [pXq]

Determinismo; $|\delta(q, s, t)| + |\delta(q, \epsilon, t)| \leq 1$

- q: estado.
- s: input.
- t: topo da stack.

10 Context-free Languages (CFL)

Chomsky Normal Form (CNF): simplificação de CFGs

- Eliminação de símbolos não usados.
- Eliminação de produções ϵ.
- Eliminação de produções unitárias.
- Variáveis dos terminais quando estão juntos são isoladas.
- Substitui-se os terminais por estas variáveis.
- Substituem-se corpos longos.

Verificar se uma string pertence a uma CFL

- **Cocke-Younger-Kasami (CYK) Algorithm**
  - x_{ij}: representa o conjunto de variáveis que produz a string i-j.
  - $O(N^3)$ usando programação dinâmica, preenche-se a tabela ($a_1a_2a_3a_4$ é a string de input):

x_{14}			
x_{13}	x_{24}		
x_{12}	x_{23}	x_{34}	
x_{11}	x_{22}	x_{33}	x_{44}
a_1	a_2	a_3	a_4

$$X_{13} = X_{12}X_{22} \cup X_{11}X_{23}$$

Pumping Lemma para CFLs

- Assume-se que L é uma CFL.
- Então, existe uma constante n com a qual, para todo o z em L com $|z| \geq n$, conseguimos escrever z = uvwxy:
  - $|vwx| \leq n$
  - $vx \neq \epsilon$ (pelo menos uma, v ou x, não pode ser a string vazia)
  - Para todo $i \geq 0$, uv^iwx^iy

Interseção de uma LR com uma CFL resulta numa CFL.

11 Turing Machine (TM)

Controlador de estados finitos (número de estados finitos).

Fita com comprimento infinito que consiste em células (cada célula pode conter um símbolo).

Input: string finita, que consiste em símbolos do alfabeto do input, colocada no início da fita (todas as outras células são marcadas com B).

Símbolos na fita: alfabeto do input + blank (B) + outros símbolos necessários.

TM T = $(Q, \sum, \Gamma, \delta, q_0, B, F)$

- Q: estados da TM.
- $\sum$: alfabeto do input.
- Γ: alfabeto da fita.
- δ: função de transição.
- q_0: estado inicial.
- B: blank.
- F: conjunto dos estados finais.

Transições entre estados ($0/X \rightarrow$):

- 0: símbolodo input.
- X: símbolo colocado na fita.
- $\rightarrow$: direção da leitura.

$\delta(q, X) = (p, Y, D)$

- q e p: estados inicial e final da transição.
- X e Y: símbolos da fita (o que está na fita e o que fica na fita, respetivamente).
- D: direção da leitura.